Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften (Dr. rer. pol.)
von der Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)
genehmigte Dissertation.

Description Logic Rules

M.Sc. Markus Krötzsch

Tag der mündlichen Prüfung: 19. Februar 2010
Referent: Prof. Dr. Rudi Studer
1. Korreferent: Prof. Dr. Peter H. Schmitt
2. Korreferent: Prof. Dr. Pascal Hitzler

Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB)
Fakultät Wirtschaftswissenschaften
Karlsruher Institut für Technologie (KIT)

Karlsruhe März 2010

Preface

Formal models of domain-specific knowledge abound in science and technology. It is desirable that such models can be managed, exchanged, and interpreted in computer systems, and the term “ontology” was coined to refer to the respective modelling artefacts.

A prominent application field for ontologies is the Semantic Web where the *Web Ontology Language* OWL is the predominant modelling language. The formal semantics of OWL is largely based on the *description logic* (DL) family of knowledge representation formalisms that are well-suited for terminological modelling. Rule-based knowledge representation languages, in contrast, have a stronger focus on modelling relationships between instances. Both perspectives are relevant in applications but the combination of rules and DLs turns out to be difficult, since vital computational properties such as decidability are lost easily.

The subject of this work is to advance the development of hybrid DL rule languages based on first-order Horn rules. Reasoning for SWRL – the combination of DLs with (first-order) *datalog* – is known to be undecidable, and we identify *DL Rules* as a novel class of decidable SWRL fragments that is closely related to DLs. New decidability results for DLs with role constructors let us include simple role conjunction and concept products into DL Rules. DL Rules are further extended with *DL-safe variables* to arrive at *DL+safe rules*. The latter generalise DL Rules and the known approaches of DL-safe rules and role-safe recursive CARIN.

This leads to expressive DL rule languages with high computational complexities, motivating the study of more restricted languages. We introduce *Horn DLs* to generalise the known DL Horn-$\mathcal{SHIQ}$, and show that many of these DLs exhibit high reasoning complexities in spite of their low data complexity. *DLP* has been proposed as a logic in the “expressive intersection” of DLs and datalog. We question the meaning of this description, and develop formal design criteria for DLP that let us specify the largest datalog-expressible fragment of description logics.

Combining these insights, we arrive at a new tractable DL rule language ELP which extends both DLP and the light-weight DL $\mathcal{EL}^{++}$, although the union of these languages is intractable. ELP incorporates DL Rules and a form of DL+safe rules, and we present a reasoning procedure based on a direct reduction to datalog that preserves the structure of rules. This also lets us derive a new datalog-based inferencing procedure for the DL $\mathcal{SROEL}(\sqcap_s, \times)$ which extends $\mathcal{EL}^{++}$.

This work advances the understanding of the relationship of rules and description logics, leading to concrete new knowledge representation formalisms of practical relevance. DL+safe rules constitute one of the broadest classes of decidable SWRL fragments known today. ELP provides a tractable DL rule language that generalises the novel light-weight ontology languages OWL RL and OWL EL as standardised by W3C, and that has been adopted as the basis for the WSML-DL

v2.0 dialect of the *Web Service Modeling Language*. Our work also suggests new rule-based implementation methods for supporting these languages based on a single inferencing algorithm.

Acknowledgements Working at AIFB gave me the opportunity to co-author two textbooks, a W3C standard, and a piece of software that is actually used – and to still engage in the research that led to this work. The unique environment that made this possible has largely been shaped and maintained by the efforts of Prof. Dr. Rudi Studer, and I wish to sincerely thank him.

I am grateful to Prof. Dr. Peter H. Schmitt for an inspiringly thorough review of this work that included many insightful observations and helpful hints.

The research reported herein had not been possible without my supervisors and co-authors Prof. Dr. Pascal Hitzler and Dr. Sebastian Rudolph. I warmly thank Pascal for support and inspiration throughout my studies, and Sebastian for extensive afternoon discussions, insightful comments, and audacious puns.

I wish to thank all colleagues and former colleagues at "Semantic Karlsruhe" for providing such an inspiring and enjoyable atmosphere. Moreover, I am indebted to numerous people with whom I have had the opportunity to discuss my research over the years, including the members of *ReaSIG* in Karlsruhe, and the *DL-istas* in Oxford and Manchester. I further thank all the *good* anonymous reviewers for their diligence and dedication – you know who you are.

Apologies for not listing everybody by name! Many of you have provided me with valuable feedback and invaluable advice. It is a pleasure to work with you.

Special thanks are due to Prof. Dr. Carsten Lutz, Dr. Bijan Parsia, and Prof. Dr. Ulrike Sattler for helpful words of advice on research in general, and on "getting your PhD done" in particular.

Much of my time outside research has been spent on developing Semantic MediaWiki, and I wish to thank everyone who has supported me in this task, especially all contributors, bug reporters, and support providers. Again, I must apologise for not listing all your names! Yaron Koren did an amazing job in keeping the project going while I was completely immersed in this work.

I will not thank Denny Vrandečić for his incurable optimism – which he cannot change anyway – but for letting me crush into his office to engage him in afternoon-filling discussions about research, the universe, and everything, as well as for the work he took from me while I was writing my thesis.

Thanks are due to my family and friends for understanding and supporting my commitment to my work. I thank my parents for too many things to list here.

All of my work is dependent upon the unconditional love and support from my wife Anja.

Markus Krötzsch
Karlsruhe, March 2010

Contents

List of Figures

Chapter 1

Introduction

Ontological modelling is relevant in a number of disciplines – prominent application areas include medicine, the life sciences, and the Semantic Web –, and various *ontology languages* have been devised as a suitable conceptual basis. Examples include CycL [Cyc02], LOOM [MB87], KIF [GF92], KRSS [PSS93], F-Logic [KLW95], Common Logic [ISO07], but also domain-specific languages such as OBO [DR06]. A prominent and highly influential representative of such languages is the *Web Ontology Language* OWL which became a W3C standard in 2004 [PSHH04] and which has been updated and extended in 2009 [OWL09]. The formal semantics of OWL is largely based on *description logics* as an expressive knowledge representation formalism with a particular emphasis on terminological, i.e. schema-level, modelling. Rule languages, in contrast, provide an alternative paradigm for modelling knowledge[1] with a stronger focus on instances and relations between them. The combination of both approaches is desirable but difficult, and – based on a more precise notion of "rule language" – it will be the main objective of this work.

The following sections provide a wider perspective and motivation for this work. Section 1.1 gives a short discussion of ontological modelling in the context of various historical developments, and discusses its relation to the Semantic Web. An intuitive introduction of description logics and their history is then provided in Section 1.2. In Section 1.3, we give an overview of popular uses of the term "rule," and outline which meaning the term will have within the remainder of this work. Section 1.4 explicates the aims and objectives of this work, and Section 1.5 offers some guidance for reading it.

[1]We generally use the term "knowledge" in the technical sense of "knowledge representation and reasoning" and especially we do not presuppose or endorse any philosophical theory of knowledge.

1.1 Ontologies and the Semantic Web

In computer science, an ontology is a description of knowledge about a domain of interest, the core of which is a machine-processable specification with a formally defined meaning.[2] Approaches to knowledge representation and reasoning, and especially the formalisms that are discussed within this work, provide the formal underpinnings for the creation and usage of ontologies in this sense. Application areas of ontologies include geoscience [Goo05, RP05, SWE, FMC$^+$09], bioinformatics [Gen00, SAR$^+$07, GGPS03], medicine [RGG$^+$94, SCC97, dCHS$^+$04, GZB06], electrical engineering [UD07, UG07], service science [SGA07], and – maybe most prominently – the Semantic Web [BLHL01].

The modern usage of ontologies marks the convergence of two strands of scientific and technological development: the description of the world in terms of abstract models, and the automated calculation with formally specified knowledge. Scientific modelling, the former of the two aspects, can be traced back to ancient philosophy, and indeed started with fundamental questions that initiated the philosophical field of *Ontology* [Sow00]. Yet, the advent of rigorous scientific models started only in the 18th century with the systematic study of natural phenomena. Classical models include, e.g., the biological classification of the *Linnean taxonomy*, the *International Classification of Diseases* (ICD), or the *Dewey Decimal Classification* (DDC) for library organisation. These examples also highlight a development toward using models for communication – the ICD was initially created for enabling international exchange of mortality statistics – and for organisation and search – an important goal of DDC is to allow users to *find* a book in a library. Both aspects have gained further importance in modern information technologies.

Today, formal models abound in science and technology, and standards have been devised for their specification. A typical example from computer science is the *Unified Modelling Language* UML.[3] Models thus also have become computational artefacts that are stored and processed in computer systems, and the requirement for more "intelligent" automatic evaluation of models was a natural consequence. In many cases, "intelligent evaluation" has been interpreted as the capability to draw logical inferences from the given information, which is where knowledge representation and reasoning comes to the fore as the second main component of ontology-based applications.

The idea of formal inferencing as a means for simulating and augmenting human reasoning has a long history which involves Aristotle's *syllogisms*, Ramon

[2]The term is derived from the philosophical discipline of *Ontology* – the study of existence and being – since a basic purpose of ontologies in computer science is to describe the existing entities and their inter-relation.

[3]`http://www.uml.org/`

Llull's "*Ars generalis ultima,*" and the visionary ideas of Gottfried Leibniz; see [Sow00] for details. Yet, significant progress toward that goal happened only in the late 19th century with the systematic development of formal logic. Although the seminal results of Gödel [Göd31] and Turing [Tur37] revealed principal boundaries both of logical deduction and of practical computation, the development of electronic computers renewed the interest in knowledge representation and (automated) reasoning, and the field of Artificial Intelligence (AI) provided the environment for extended research activities in that area; see, e.g., [RN03] for an introduction.

It was soon discovered that computational complexity is a major limiting factor for automated deduction, destroying the hope that the rapid growth of computing power would suffice to solve all practically relevant reasoning problems as long as they were at least decidable. Continued research revealed the fundamental conflict between maximising the expressive power of a knowledge representation formalism on the one side, and minimising the computational complexity of the relevant reasoning problems for this formalism on the other. This basic trade-off between expressiveness and computational feasibility has consequences for the design of modelling languages, and thus relates knowledge representation and reasoning to formal modelling.

Ontological modelling – though not always with that particular name – has been done in various contexts and applications. The *expert systems* of the 1980s were mostly based on rule languages for modelling knowledge, whereas *Cyc* became known as a major effort for creating a huge and complex ontology based on a more expressive knowledge representation language [LG90]. Notable modelling efforts have also been made in life sciences and medicine, leading to ontologies of significant practical impact. Prominent clinical and health care ontologies include GALEN (around 25,000 atomic concepts [RGG+94]), SNOMED-CT (around 300,000 atomic concepts [JS08]), and the NCI Thesaurus of the US National Cancer Institute (around 25,000 concepts [dCHS+04]). However, the most prominent use of ontologies to date relates to a more recent activity of establishing a *Semantic Web*.

The Semantic Web has been conceived as an extension of the World Wide Web that allows computers to intelligently search, combine, and process Web content based on the meaning that this content has to humans [BLHL01, SBLH06]. In the absence of human-level artificial intelligence, this can only be accomplished if the intended meaning (i.e. the *semantics*) of Web resources is explicitly specified in a format that is processable by computers. For this it is not enough to store data in a machine-processable syntax – every HTML page on the Web is machine-processable in a sense – but it is also required that this data is endowed with a formal *semantics* that clearly specifies which conclusions should be drawn from the collected information.[4] Clearly, this would be an impossible endeavour when aiming at all human knowledge found on the Web, given that it is often hard enough for humans to even agree on the contents of a certain document, not to mention formalising it in a way that is meaningful to computers. In reality, of course, the purpose of the Semantic Web is rather to enable machines to access *more* information that hitherto required human time and attention to be used. While this is a reasonable goal from a practical viewpoint, it also means that "Semantic Web" does not refer to a concrete extension of the World Wide Web, but rather to an ideal toward which the Web evolves over time. At the same time, any progress in this field can similarly be useful in applications that are not closely related to the Web.

Realising the above-mentioned goals makes it necessary to address a number of difficult challenges that are not addressed by classical Web technologies. This is where topics of formal modelling and automated deduction come into play. Expressing human knowledge in a formally specified language is a classical modelling task. The rich experiences gathered within this domain throughout history are an important guide in identifying relevant modelling structures up to the present day. The most recently developed Semantic Web language *OWL 2* (see below), for instance, has been influenced by feature requests from modelling use cases in life sciences. Moreover, semantic technologies can draw from modelling methodologies, software applications, and corresponding user-interface paradigms that have been developed for supporting humans in the task of constructing models.

How knowledge is to be modelled also depends, of course, on the intended usage of the constructed model. On the Semantic Web, one would like computer programs to draw conclusions from given information, so that aspects of formal knowledge representation and reasoning become relevant. In the first place, the insights gathered in this field help in understanding the fundamental difficulties and limits that one has to be aware of when constructing reasoning systems. On the practical side, semantic technologies can build on algorithms and tools that were developed for solving relevant inferencing problems.

The above discussion views the development of the Semantic Web as an approach of incorporating knowledge modelling and automatic deduction into the Web. Conversely, it is also true that semantic technologies introduce aspects and features of Web applications into the domain of formal modelling and knowledge representation. Most basically, the Web introduces a notion of distributed,

[4]Note that, indeed, the term "semantics" occurs with two distinct interpretations in the previous two sentences. In the first sense, it refers to the meaning that texts in a human language have: this is the usage common in linguistics. In the second sense, it refers to the formal interpretation of a computer language: this is the usage common in computer science. Both notions of the term are found in discussions of the Semantic Web.

heterogeneous, yet inter-linked information that is novel to the other disciplines. Whereas Web data is indeed independently published and maintained in many sources, it is still universally accessible based on global addressing schemes and standardised protocols. More specifically, the Web emphasises the importance of clearly specified, standardised languages that can be used to exchange data across software boundaries. Although there are some examples of earlier standardisation activities around knowledge representation formalisms,[5] the Semantic Web clearly has increased the practical importance of standardisation in this area. These activities have also facilitated tool interoperability and information exchange in application areas beyond the Web.

As of today, the most prominent standards for semantic technologies are the *Resource Description Framework* RDF [MM04, KC04, Bec04, Hay04], enabling the exchange of factual data, the *SPARQL* language for querying such data [PS08, BB08, CFT08], and the *Web Ontology Language* OWL for modelling complex schematic knowledge [OWL09]. As the name suggests, OWL is most relevant for ontological modelling, although some of its modelling features were already introduced by RDF Schema [BG04]. This work is closely related to the knowledge representation formalism that provides the formal underpinning for a significant part of the OWL standard – *description logics* (DLs) – which will be introduced in more detail in Section 1.2.

The OWL standard has first been published in 2004, and an updated and extended version has recently been released under the name *OWL 2*.[6] The new standard is fully compatible with the old one, i.e. with "OWL 1," but it provides a number of additional features both on the technical and on the logical level. We will not introduce the syntactic details and formal intricacies of OWL 2 herein, see [HKP⁺09, HKR09, HKRS08] for a detailed introduction. This work relates to the so-called *direct semantics* of OWL 2 which is based on the description logic $\mathcal{SROIQ}$, and it is generally more convenient to use the syntax of DL or first-order logic for our purposes.

A particular aspect that is worth special emphasis, however, is the inclusion of tractable sub-languages – so-called *profiles* – into OWL 2 [MCH⁺09]. The three profiles that are provided are called OWL EL, OWL RL, and OWL QL. Their purpose is to provide "maximal" sub-languages of OWL 2 for which standard reasoning problems can be solved in polynomial time. It should be noted that the union of any two of the languages does no longer have this property. The fact that these profiles have been introduced in OWL 2 witnesses the increased demand for tractable formalisms, and it illustrates the practical impact that research on

[5]The most prominent example is the logic programming language *Prolog* that is covered by the ISO/IEC 13211 standard, cf. [DEDC96].

[6]See [OWL09] for an overview; the main technical specifications are [MPSP09, MPSC09, SHKG09, PSM09, Sch09b, MCH⁺09]

the worst-case complexity of important reasoning problems has within this field. Establishing complexity results for new and extended knowledge representation languages will also be a major topic of this work.

1.2 Description Logics

Description logics (DLs) are among the most important formalisms for ontological modelling today, which is also due to their central rôle for the semantics of the Web Ontology Language OWL. DLs have developed as a family of related knowledge representation languages ranging from light-weight formalisms for which common inference tasks can be solved in polynomial time to highly expressive logics for which reasoning is undecidable. A major design goal for description logics, however, typically is to retain decidability of standard inferencing tasks such as checking knowledge base satisfiability. Another common feature of the overwhelming majority of today's description logics is that they can be considered as fragments of first-order logic (with equality),[7] although a different syntax is commonly used for DLs.

Theories of a DL are usually called *knowledge bases*, which specifically avoids any informal connotations that the general term "ontology" often has, as discussed in Section 1.1. DL knowledge bases describe models that are based on *individual elements*, *classes* of which elements can be instances, and *binary relationships* between the elements. These three types of semantic entities are syntactically denoted by means of *individual* names, *concept* names, and *role* names, which essentially correspond to constants, and unary and binary predicates in first-order logic.[8] Some DLs have been extended with datatypes, thus introducing notions of sorted logic, but these approaches will not be considered within this work.

Basic statements that can be formulated with this vocabulary include:

- *assertions* such as $\mathtt{City}(\mathtt{ulm})$ ("The element denoted by $\mathtt{ulm}$ is in the class denoted by $\mathtt{city}$" i.e. "Ulm is a city"), or $\mathtt{locatedIn}(\mathtt{dresden}, \mathtt{germany})$ ("Dresden is located in Germany"),
- *concept inclusions* such as $\mathtt{Capital} \sqsubseteq \mathtt{City}$ ("capitals are cities"), and
- *role inclusions* such as $\mathtt{captialOf} \sqsubseteq \mathtt{locatedIn}$ ("a capital of some country is always located within this country").

[7]Exceptions include, e.g., DLs that include operators for specifying transitive closure that are rarely considered today.

[8]Some application areas use other terms, and especially OWL uses the terms "class" and "property" to refer to concepts and roles. In this work,"class" always refers to the semantic entity that a concept describes, i.e. to a set of individuals within a model.

Here we adopt the convention of capitalising concept names. In addition, DLs provide many operators for combining concept names into complex concept expressions, the semantics of which is derived from the semantics of the individual components. Basic operators include the Boolean constructors $\sqcap$ (intersection), $\sqcup$ (union), and $\neg$ (negation). *Role restrictions* further allow us to describe classes based on binary relationships of individual elements. For example, the concept $\exists\,\mathtt{citizenOf}.\mathtt{EUCountry}$ describes the class of all things that are citizens of some EU country, while $\forall\,\mathtt{citizenOf}.\mathtt{EUCountry}$ refers to those things that are citizens of nothing but EU countries (including, as usual in first-order logic, the things that are not citizens of anything). Combining these expressive features, it can be stated that people who have nothing but EU citizenships are either EU citizens or have no citizenship at all:

$$\mathtt{Person} \sqcap \forall\,\mathtt{citizenOf}.\mathtt{EUCountry} \sqsubseteq \mathtt{EUCitizen} \sqcup \neg\exists\,\mathtt{citizenOf}.\top.$$

Here, the operator $\top$ denotes the class of all elements, so $\neg\exists\,\mathtt{citizenOf}.\top$ refers to things without any citizenship. Further constructors are introduced in Chapter 3. DLs typically provide much less features for creating complex role expressions than for creating complex concept expressions. A basic example are *inverse roles*, as in the concept expression $\exists\,\mathtt{citizenOf}^{-}.\mathtt{Person}$ that describes the class of all things that have some citizen who is a person. More advanced role constructors are less common, but will be relevant for various parts of this work; see Chapter 5 for a detailed discussion. A construct that is available in many modern DLs, and in particular in (all profiles of) OWL 2, are so-called *complex role inclusions* that allow us to state that, whenever two individuals are connected with a chain of relations, they must also be directly related by some other relation. For example, we can formulate that the brother of someone's father is her uncle:

$$\mathtt{hasFather} \circ \mathtt{hasBrother} \sqsubseteq \mathtt{hasUncle}.$$

Expressions of this kind significantly increase the modelling power of DLs, and can easily lead to higher reasoning complexities or even to undecidability. At the same time, complex role inclusions provide an important basis for some of the approaches of modelling rules in description logics that are discussed in this work.

Historically, description logics developed out of *semantic networks* [Qui68] and *frame logics* [Min74] in the mid-1980s. The knowledge representation language KL-ONE [BS85] and the *frame logic* $\mathcal{FL}$ [BL84] are often considered to be the first description logics. However, it was soon discovered that KL-ONE leads to undecidable inferencing problems [SS89], and that fundamental reasoning tasks tend to be computationally intractable even in very simple DLs like $\mathcal{ALC}$ [SSS91]; see [DLNS96, BCM^{+}07] for an overview of related results.

Yet, the actual implementation of inferencing engines has been a major goal of DL research since its early days, and numerous systems have been proposed. The initial implementation for KL-ONE was soon succeeded by various early DL reasoners such as Loom [MB87], Krypton [BPL85], Nikl [KBR86], Back [QK90], Classic [BBMR89], and Kris [BH91]. Many of these early systems were not only very efficient and scalable but also, unfortunately, incomplete. Namely, the structural inferencing algorithms that they applied are insufficient for discovering all logical inferences in all but the most basic DLs. Later implementations overcame this problem by employing *tableaux algorithms*. Examples of modern systems that are based on this idea include FaCT++ [TH06], Pellet [SPG$^+$07], and RacerPro [HM01]. In spite of the high worst-case complexities of the underlying reasoning problems, it turned out that many practical problems can be solved by using such highly optimised and well-engineered implementations. More recently, alternative approaches have been proposed to address common problems in tableau-based systems, such as the relatively poor handling of large amounts of instance data. Examples include resolution-based algorithms as in KAON2 [MS06], the hypertableau system HermiT [MSH07, MSH08], approaches based on type elimination [RKH08d, RKH08c], and recent "consequence-based" approaches [Kaz09a].

Moreover, a number of light-weight description logics have been studied in recent years to address the emerging requirements for reasoning with very large ontologies. Notable approaches include the description logics $\mathcal{EL}^{++}$ [BBL05], DL-Lite [CGL$^+$07], and DLP [GHVD03] which provide the formal background for the OWL 2 profiles OWL EL, OWL QL, and OWL RL, respectively. Both $\mathcal{EL}^{++}$ and DLP are studied and extended within this work as part of the general struggle for more expressive yet tractable knowledge representation languages. DLP – Description Logic Programs – are of additional interest since they have been proposed as a language within the "intersection" of description logics and rule languages. We will see that the actual relationship between DL and rules is significantly more complicated – the term "intersection" is rather not adequate here –, but DLP still provides an inspiration for our studies.

1.3 What is a Rule?

Rule-based modelling has a long tradition in knowledge representation and reasoning, and a plethora of different rule formalisms have been proposed. What these formalisms have in common is not so much their formal background – of which some rule languages have very little – but rather a common metaphor for modelling knowledge. In the broadest sense, a rule could be any statement which says that a certain *conclusion* must be valid whenever a certain *premise* is satisfied, i.e. any statement that could be read as a sentence of the form "if . . . then

. . .”[9] Typical representatives are rules in logic programming, association rules in databases, or production rules as they occur in various business rules systems. In this work, we will confine ourselves to concrete kinds of first-order Horn logic rules that will be defined more accurately. Yet it is worth noting that the term “rule” as such refers rather to a knowledge modelling paradigm than to a particular formalism or language. And it is also this paradigm that makes rules attractive in many applications, since users sometimes find it more natural to formulate knowledge in terms of rules than in terms of other kinds of ontological axioms.

But the difference between rules and ontologies is not merely pedagogical. In the cases we consider, rules can often help to express knowledge that cannot be formulated in description logics. At the same time, there are also various features of DL that rule languages do not provide, so a natural question to ask is how the strengths of DL and of rules can be combined. It turns out that this is indeed possible, but that the added power often also comes at the price of higher complexity and more difficult implementation.

It has been noted that rules of any type should consist at least of a premise and a conclusion, with the intuitive meaning that in any situation where the premise applies the conclusion must also hold. Such a general description comprises some, if not all, DL axioms. Consider, e.g., the “rule” that, if a person is the author of a book then she is a (member of the class) book author. This can surely be expressed in DL: using the syntax introduced in Section 1.2, we can write

$$\mathtt{Person} \sqcap \exists \mathtt{authorOf}.\mathtt{Book} \sqsubseteq \mathtt{Bookauthor}.$$

It has already been mentioned that DLs can usually be considered as fragments of first-order predicate logic. Indeed, it turns out that we can equivalently write the above statement as a predicate logic formula (see Section 3.2 for formal details):

$$\forall x.(\mathtt{Person}(x) \land \exists y.(\mathtt{authorOf}(x, y) \land \mathtt{Book}(y)) \to \mathtt{Bookauthor}(x)).$$

Using standard semantic equivalences of first-order logic, we thus obtain:

$$\forall x \forall y.(\mathtt{Person}(x) \land \mathtt{authorOf}(x, y) \land \mathtt{Book}(y) \to \mathtt{Bookauthor}(x)).$$

This formula is a logical implication with universally quantified variables, hence it comes close to our vague idea of a “rule.” The universal quantifiers express the fact that the implication is applicable to all individuals that satisfy the premise. But defining “first-order logic rules” to be arbitrary first-order logic implications would not say much since every first-order logic formula can be rewritten to fit

[9]Instead of the terms “premise” and “conclusion” it is also common to speak of “precondition” and “postcondition,” “body” and “head,” or “precedent” and “antecedent” of a rule. We use these terms interchangeably.

that syntactic form. One therefore typically restricts to so-called *Horn rules*: implications with conjunctions of atomic formulae as their body and head. Using the term "rule" as a synonym for "first-order Horn implication" has become common practice in connection with the Semantic Web, as witnessed by formalisms such as the *Semantic Web Rule Language* [HPSB+04], *Description Logic Rules* [KRH08a], *DL-safe rules* [MSS05], and the *Rule Interchange Format* (RIF-Core [BHK+09]), most of which will also be discussed in more detail within this work.

While a main focus of this work are (extensions of) the rule languages mentioned above, it should be noted that there are a number of rather different interpretations of the term "rule" outside of first-order logic. Among the most popular rule formalisms in computer science is certainly logic programming [Llo88], which is closely associated with the Prolog programming language and its various derivatives and extensions [DEDC96, CM03]. At first glance, Prolog rules appear to be very similar to first-order logic implications that merely use a slightly different syntax, putting the precondition to the right of the rule. The example above would read as follows in Prolog:

$$\mathtt{Bookauthor}(X) \;\mathtt{:-}\; \mathtt{Person}(X), \mathtt{authorOf}(X, Y), \mathtt{Book}(Y).$$

Basic Prolog indeed has the same expressiveness as first-order Horn logic, and can equivalently be interpreted under a first-order logic semantics. But there are many extensions of Prolog that introduce features beyond first-order logic, such as operational plug-ins (e.g., for arithmetic functions) and *non-monotonic* inferences which derive new results from the fact that something else can *not* be derived. Logic programming in this form, as the name suggests, has been conceived as a way of specifying and controlling powerful computations, and not as an ontology language for direct interchange on the Web. Two ontologies from different sources can usually be merged simply by taking the union of their axioms (meaningful or not), whereas two independent Prolog programs can hardly be combined without carefully checking manually that the result is still a program that can be successfully executed by the employed logic programming engine. The use of logic programming in combination with ontologies can still be quite useful, but most of the research that has been conducted in this field is beyond the scope of this work (see Section 4.3 for an overview).

A related rule formalism that has also been proposed as an ontology language is *F-Logic* [KLW95]. While F-Logic incorporates a Prolog-like rule syntax that is evaluated under a non-monotonic semantics in current systems, its core is the *frame* syntax for defining classes and instances from which it derives its name. F-Logic is closely related to the upcoming Rule Interchange Format, especially to the *Basic Logic Dialect* RIF-BLD [BK09]. The latter does not include non-monotonic features, and can be evaluated under a first-order logic semantics that allows for a combination of RIF-BLD with OWL and the rule-based extensions considered within this work (see also [dB09]). We are more interested in (onto)logical expressiveness, and will not discuss the technical details of this combination within this work.

Yet another kind of rules that is very relevant in practice is known as *production rules*, such as *Event Condition Action Rules* or *business rules*. Rule languages of this type apply a more *operational* interpretation of rules, i.e. they view rules as program statements that can be *executed* actively. For ontology languages like OWL, the semantics of an ontology is not affected by the order in which ontological axioms are considered. In contrast, for rules with an operational semantics it can be crucial to know which rule is executed first, and part of the semantics of production rules is concerned with the question of precedence between rules. A popular evaluation strategy for production rule systems is known as the *Rete Algorithm* [For82]. Many different kinds of production rule engines are used in practice and many rule engines implement their own customised semantic interpretations of rules that do not follow a shared published semantics. As such, production rules again are hard to interchange between different systems, and the ongoing work on the W3C Rule Interchange Format is among the first efforts to allow for the kind of interoperability that a common semantic standard can offer [dSMPH09]. Yet it is currently unclear how production rule engines should best be combined with ontology-based systems, and we shall not pursue this endeavour in the remainder of this work.

Besides the interpretation of “rule” in these diverse approaches, the term can also have an even more general meaning in the context of knowledge representation. In particular, a “deduction rule” or “rule of inference” is sometimes understood as an instruction of how to derive additional conclusions from a logical theory. In this sense, the rule is not part of the encoded knowledge, but rather a component of algorithms that are used to process this knowledge. It can be argued that the deduction rules of virtually any calculus could be expressed as logical rules of some suitable logic. But this logic is typically required to be very expressive, making it difficult or impossible to implement general-purpose reasoners that can process the logical theory that was derived from a set of deduction rules. Since we are interested in semantic technologies that represent knowledge in a machine-processable way, the topic of this work is rules in the earlier sense, i.e. axioms for representing ontological knowledge in the form of a rule.

1.4 Aims and Objectives

The discussion in Section 1.3 illustrates that rule-based formalisms are highly relevant in various application areas of formal or semi-formal knowledge modelling.

In spite of the rather wide interpretation of the term “rule” that is common in various areas, many of these approaches – especially the ones that are related to logic programming and deductive databases – also provide a clearly defined formal semantics with well-understood relationships to first- and higher-order logic. It therefore seems natural to apply selected rule-based approaches to ontological modelling tasks as discussed in Section 1.1, e.g. in the context of the Semantic Web.

This simple conclusion, however, disregards the fact that a large part of today’s ontological models are based on description logics as introduced in Section 1.2. There are various reasons why DLs have become a predominant modelling formalism in many areas, including their strong focus on terminological, i.e. schema-level, modelling. Rule languages, in contrast, are typically superior for modelling relationships between instances, and more scalable when handling large data sets. Much research has been conducted in recent years to reconcile both approaches,[10] yet many basic questions remain open even when restricting to rules with a first-order semantics.

The principal objective of this work therefore is to advance the development of hybrid knowledge representation formalisms that combine aspects of rules and description logics. The two main motivations underlying this goal are apparent from the above discussion:

1. Extending the expressiveness and practical applicability of DL-based ontology languages by incorporating features of rule-based formalisms
2. Increasing the interoperability between rule languages and description logics

It has been mentioned before that there is often a trade-off between expressiveness and practical applicability, and we therefore must aim for a suitable balance between the two. Indeed, the combination of function-free first-order Horn logic – a simple rule language known as (monotonic) *datalog* [AHV94] – with description logics has been proposed as (the logical core of) the *Semantic Web Rule Language* (SWRL) [HPSB$^+$04], but reasoning in SWRL already turns out to be undecidable.

To address these challenges within this work, we pursue three related, and often intertwined, strands of research which define concrete goals for the remainder of this work:

Discovering and extending decidable fragments of SWRL While reasoning in the unrestricted combination of DL and datalog is generally undecidable, SWRL still defines a fragment of first-order logic that is useful as a framework for studying rule extensions of description logics. A concrete research

[10]See Section 4.3 for a general overview.

question then is: Which non-trivial fragments of SWRL allow for decidable reasoning, and what is the worst-case complexity of reasoning in these cases?

Identifying and characterising rule fragments of DLs A further approach that is dual to the first one in a certain sense is to study the commonalities of description logics and rules. Related research questions in this case are: How can DLs be restricted so as to recover certain positive characteristics of first-order Horn logic? How does this restriction affect reasoning complexities? Is it possible to characterise the "intersection" of DL and datalog as a fragment of first-order logic? These questions relate to *Horn DLs* and *Description Logic Programs* (DLP).

Developing tractable hybrid knowledge representation languages Recent applications of ontologies face an ever increasing amount of data which has inspired research on *tractable* knowledge representation formalisms for which reasoning can be achieved in polynomial time. Given the additional focus on instance data that rules provide, the search for tractable yet expressive formalisms is of special importance in this context.

A summary of our contributions in each of these areas is given in Chapter 10.

Studying worst-case complexities in the context of this work allows us to compare *hardness* – in a computational sense – of standard inference tasks to hardness of well-known description logics, and thus helps to understand the theoretical expressivity of our approaches in relation to other knowledge representation languages. To some extent, complexity measures can also hint at the feasibility of implementing efficient reasoning algorithms in practice, though worst-case complexity is generally too coarse a measure to obtain conclusive results in this respect.

1.5 Guide to the Reader

An overview of the chapters of this work and their mutual dependencies is given below. Many chapters provide extensive informal discussions to augment the rigorous formal parts. Nevertheless, intuitive explanations are generally in danger of over-simplification and ambiguity, and the reader is thus advised to refer to the according definitions, theorems, or – for material beyond the scope of this work – to the given literature for precise authoritative statements. We also explicitly point out if a section is largely introductory in nature, so that experts might want to skip it and refer back to it if needed. A comprehensive index is provided to support this style of reading.

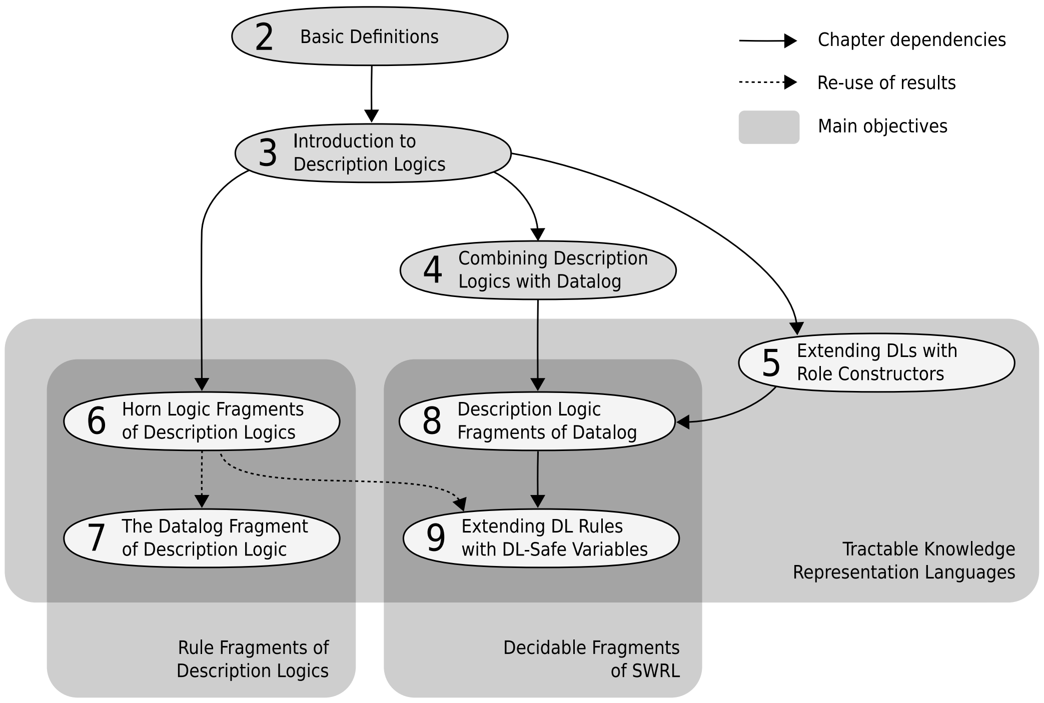
2 Basic Definitions
3 Introduction to Description Logics
4 Combining Description Logics with Datalog
5 Extending DLs with Role Constructors
6 Horn Logic Fragments of Description Logics
7 The Datalog Fragment of Description Logic
8 Description Logic Fragments of Datalog
9 Extending DL Rules with DL-Safe Variables
Chapter dependencies
Re-use of results
Main objectives
Tractable Knowledge Representation Languages
Rule Fragments of Description Logics
Decidable Fragments of SWRL

Figure 1.1: Dependencies between chapters and relation to main objectives

This work contains proofs. Readers who are only interested in the results can safely skip these parts by continuing with the narrative beyond the subsequent □ symbol. Moreover, a number of complex proofs have been split into separate lemmata which can also be skipped as parts of the proof. Statements that are marked as *theorem* or *proposition*, in contrast, are considered to be interesting as results in their own right.

The dependencies between the individual chapters, and the relationship to the main objectives as explained in Section 1.4 is illustrated in Fig. 1.1. The synopsis of the chapters is as follows:

Chapter 2 This chapter briefly reviews first-order logic and makes some remarks on complexity theory that can safely be skipped by knowledgeable readers. However, Section 2.2 introduces *emulation* as a new notion that conveniently describes semantic correspondences encountered throughout this work.

Chapter 3 This chapter formally introduces DLs by presenting the description logic $\mathcal{SROIQ}$ (and our notation for it) as a basis for large parts of this work. We also clarify the relationship of DLs to first-order logic and other logics, and give an overview of DL nomenclature.

Chapter 4 This chapter introduces *datalog* as a first-order rule language and defines its combination with $\mathcal{SROIQ}$ that we will call *SWRL* throughout this work. Moreover, an extended summary of related works is provided in Section 4.3 and 4.4.

Chapter 5 The topic of this chapter are extensions of description logics with *role constructors*, which also play an important rôle for aligning the expressiveness of rules and DLs. New results are derived for highly expressive DLs, but also for the tractable description logic $\mathcal{SROEL}(\sqcap_s, \times)$ for which reasoning is reduced to inferencing in datalog.

Chapter 6 This chapter provides a general definition of Horn description logics based on existing work for Horn-$\mathcal{SHIQ}$, and establishes a number of complexity results for Horn DLs. The related proofs – a PSpace tableaux procedure and various reductions of halting problems for (alternating) Turing machines – are among the technically most interesting arguments in this work. We also discuss the light-weight Horn DL $\mathcal{RL}$ which is closely related to OWL 2 RL [MCH$^+$09].

Chapter 7 This chapter characterises the largest datalog-expressible fragment of $\mathcal{SROIQ}$, thus extending the existing *DLP* formalism [GHVD03]. An extended discussion is provided to arrive at a suitable definition of "largest" and "datalog-expressible." Establishing either property for the defined language $\mathcal{DLP}$ requires intricate proofs that utilise model-theoretic properties that distinguish datalog from Horn logic *with* function symbols.

Chapter 8 *Description Logic Rules* are defined and studied within this chapter. DL Rules provide an interesting family of decidable SWRL fragments that can be expressed in description logics by means of computationally simple yet not necessarily obvious encodings. This new approach is generalised to a large class of DLs, including DLs with the additional role operators of Chapter 5.

Chapter 9 DL Rules are applied within this chapter to arrive at a generalisation of DL-safe rules [MSS05] which we call *DL+safe rules*. We study the complexity of this extended formalism and introduce the tractable hybrid knowledge representation language ELP.

Chapter 10 This final chapter concludes by summarising and discussing the obtained results, and by providing an outlook to future work.

We point out that there is a clear distinction between chapters that provide introductory or preliminary information – Chapters 1, 2, 3, 4, and 10 –, and chapters that contain novel results – Chapters 5, 6, 7, 8, 9. Each chapter starts with a more detailed overview of its contents, and chapters with novel results provide a concluding summary and a discussion of related works.

Chapter 2

Basic Definitions

This chapter mostly introduces basic definitions and results that are required in later parts of this work, but it also introduces a novel notion of *emulation* that we will use frequently for describing a particular kind of semantic correspondence between logical theories or knowledge bases.

We begin by recalling first-order logic with equality in Section 2.1, discuss important types of logical correspondences in Section 2.2, and conclude with some brief remarks on complexity theory in Section 2.3.

2.1 First-Order Logic with Equality

In this section, we give a brief introduction to first-order logic with equality (denoted as $\mathbf{FOL}_{\approx}$) which constitutes the overarching semantic framework for the knowledge representation formalisms that are studied within this work. Our main goal is to provide a concise reference for basic notions and notations that are used in later chapters. Readers without prior knowledge on first-order logic may wish to consider a more extended introductory text, e.g. the textbook [Fit96].

Definition 2.1.1 A *signature* $\langle \mathbf{I}, \mathbf{F}, \mathbf{P}, \mathbf{V} \rangle$ of first-order logic with equality ($\mathbf{FOL}_{\approx}$) consists of a set of *individual names* (or *constant symbols* or simply *constants*) $\mathbf{I}$, a set of *function symbols* $\mathbf{F}$, a set of *predicate names* (or *predicate symbols* or just *predicates*) $\mathbf{P}$, and a set of *variable names* $\mathbf{V}$, all of which are mutually disjoint and finite. The function $\mathsf{ar} : \mathbf{F} \cup \mathbf{P} \to \mathbb{N}$ associates a natural number $\mathsf{ar}(p)$ with each function or predicate symbol $p \in \mathbf{F} \cup \mathbf{P}$ that defines the (unique) *arity* of p.

Based on a $\mathbf{FOL}_{\approx}$ signature $\langle \mathbf{I}, \mathbf{F}, \mathbf{P}, \mathbf{V} \rangle$, we define the following notions. The set of *terms* is defined to be the smallest set such that

- if $t \in \mathbf{I} \cup \mathbf{V}$, then t is a term, and

– if $f \in \mathbf{F}$ with $\mathsf{ar}(f) = n$, and if $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is also a term.

Terms are used as arguments for predicates to form atomic formulae. An *atom* is an expression of the form $P(t_1, \ldots, t_n)$ with $P \in \mathbf{P}$ and $\mathsf{ar}(P) = n$, or an expression of the form $t \approx s$, where $t_1, \ldots, t_n, t, s$ are terms. The set of $\mathbf{FOL}_\approx$ *formulae* is defined to be the smallest set that contains all atoms, and such that:

– $\top$ and $\bot$ are formulae,

– if φ is a formula, then so is $\neg\varphi$ (negation),

– if φ and ψ are formulae, then so are $(\varphi \wedge \psi)$ (conjunction), $(\varphi \vee \psi)$ (disjunction), and $(\varphi \rightarrow \psi)$ (implication),

– if φ is a formula, and $x \in \mathbf{V}$, then $\forall x.\varphi$ (universal quantification) and $\exists x.\varphi$ (existential quantification) are formulae.

A *literal* is an atom or the negation of an atom.

A *subformula* is a substring of a formula that is again a formula. An occurrence of a variable x in a formula φ is *bound* if it is contained in a subformula of the form $\mathsf{Q}x.\psi$ of φ with $\mathsf{Q} \in \{\exists, \forall\}$. A *sentence* (or *closed formula*) is a formula that contains only bound occurrences of variables. A *theory* of $\mathbf{FOL}_\approx$ is a set of sentences. $\Diamond$

We explicitly introduce $\top$ and $\bot$ to represent *true* and *false* syntactically. As usual, parentheses will be omitted when no confusion is likely. Moreover, we will often not mention the signature explicitly if irrelevant or clear from the context. Note that we assume variables to be part of the signature, and that we generally assume signatures to be finite. This is relevant when studying the worst-case complexity of related reasoning problems, since Turing machines – the primary vehicles for complexity considerations – require finite alphabets for representing inputs. This does not imply that we cannot introduce additional symbols as needed, and in particular we assume that the underlying signature is extended whenever new symbols are required in a syntactic construction. The semantics of first-order logic is defined as follows.

Definition 2.1.2 A $\mathbf{FOL}_\approx$ *interpretation* $\mathcal{I}$ is a tuple $\langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$, consisting of a non-empty *interpretation domain* $\Delta^\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$. The domain is a set of *individuals* that defines the (abstract) world within which all symbols are interpreted. Symbols of the signature are interpreted as follows:

– If $a \in \mathbf{I}$ is an individual name, then $a^\mathcal{I} \in \Delta^\mathcal{I}$.

– If $f \in \mathbf{F}$ is a function symbol of arity $\mathsf{ar}(f) = n$, then $f^\mathcal{I}$ is a function from $(\Delta^\mathcal{I})^n$ to $\Delta^\mathcal{I}$.

- If $P \in \mathbf{P}$ is a predicate of arity $\mathsf{ar}(P) = n$, then $P^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$.

Here, $(\Delta^{\mathcal{I}})^n$ denotes the set of n-tuples of elements of $\Delta^{\mathcal{I}}$. A *variable assignment* $\mathcal{Z}$ for $\mathcal{I}$ is a mapping $\mathcal{Z} : \mathbf{V} \to \Delta^{\mathcal{I}}$. Given an element $\delta \in \Delta^{\mathcal{I}}$ and a variable $x \in \mathbf{V}$, we write $\mathcal{Z}\{x \mapsto \delta\}$ to denote the variable assignment that assigns x to δ, and that agrees with $\mathcal{Z}$ on all other variables.

Given an interpretation $\mathcal{I}$ and a variable assignment $\mathcal{Z}$ for $\mathcal{I}$, the interpretation $t^{\mathcal{I},\mathcal{Z}}$ of a term t is inductively defined as follows:

- If $t \in \mathbf{I}$ then $t^{\mathcal{I},\mathcal{Z}} := t^{\mathcal{I}}$.
- If $t \in \mathbf{V}$ then $t^{\mathcal{I},\mathcal{Z}} := \mathcal{Z}(t)$.
- If $t = f(t_1, \dots, t_n)$ then $t^{\mathcal{I},\mathcal{Z}} := f^{\mathcal{I}}(t_1^{\mathcal{I},\mathcal{Z}}, \dots, t_n^{\mathcal{I},\mathcal{Z}})$.

The *truth value* $\varphi^{\mathcal{I},\mathcal{Z}}$ of a formula φ is defined as follows:

- Set $\top^{\mathcal{I},\mathcal{Z}} :=$ *true* and $\bot^{\mathcal{I},\mathcal{Z}} :=$ *false*.
- For $\varphi = P(t_1, \dots, t_n)$, set $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if $\langle t_1^{\mathcal{I},\mathcal{Z}}, \dots, t_n^{\mathcal{I},\mathcal{Z}} \rangle \in P^{\mathcal{I}}$, and $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For $\varphi = t_1 \approx t_2$, set $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if $t_1^{\mathcal{I},\mathcal{Z}} = t_2^{\mathcal{I},\mathcal{Z}}$, and $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For $\varphi = \neg\psi$, define $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if $\psi^{\mathcal{I},\mathcal{Z}} =$ *false*, and $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For $\varphi = (\psi_1 \wedge \psi_2)$, define $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if $\psi_i^{\mathcal{I},\mathcal{Z}} =$ *true* for all $i \in \{1, 2\}$, and $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For $\varphi = (\psi_1 \vee \psi_2)$, define $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if $\psi_i^{\mathcal{I},\mathcal{Z}} =$ *true* for some $i \in \{1, 2\}$, and $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For $\varphi = (\psi_1 \to \psi_2)$, define $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if $\psi_1^{\mathcal{I},\mathcal{Z}} =$ *false* or $\psi_2^{\mathcal{I},\mathcal{Z}} =$ *true*, and $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For $\varphi = \exists x.\psi$, define $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if there is some $\delta \in \Delta^{\mathcal{I}}$ such that $\psi^{\mathcal{I},\mathcal{Z}\{x\mapsto\delta\}} =$ *true*, and set $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For $\varphi = \forall x.\psi$, define $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *true* if, for all $\delta \in \Delta^{\mathcal{I}}$, we find that $\psi^{\mathcal{I},\mathcal{Z}\{x\mapsto\delta\}} =$ *true*, and set $\varphi^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.

The truth value of sentences does not depend on any variable assignment, so we can omit assignments in this case. A sentence φ is *satisfied* (or *modelled*) by $\mathcal{I}$ if $\varphi^{\mathcal{I}} =$ *true*, and a theory T is *satisfied* (or *modelled*) by $\mathcal{I}$ if $\mathcal{I}$ satisfies all elements of T. We write $\mathcal{I} \models \varphi$ and $\mathcal{I} \models T$ in these cases, and say that $\mathcal{I}$ is a *model* of φ and T, respectively. ◇

This model theory leads to the well-known notions of logical consistency and entailment:

Definition 2.1.3 Consider theories T and T'.

- T is *consistent* (or *satisfiable*) if it has a model and *inconsistent* (or *unsatisfiable*) otherwise,
- T *entails* T', written $T \models T'$, if all models of T are also models of T'.

This terminology is extended to formulae by treating them as singleton theories. A theory or formula that is entailed is also called a *logical consequence*. ◇

The inclusion of equality in $\mathbf{FOL}_{\approx}$ has semantic effects, but does not significantly increase expressiveness. A related discussion can be found in Section 4.1.3.

2.2 Semantic Correspondences between Logical Theories

An important motive for basing knowledge representation languages on formal logic is the increased level of semantic interoperability that this enables. Indeed, a formal semantics effectively provides a declarative, implementation-independent specification of the conclusions that can be drawn from a given logical theory, thus acting as a standard for tool developers and practitioners. Ideally, logical theories can thus be used in different tools and in combination with different other theories, while still preserving their intended meaning. Moreover, even if two theories are not identical, it is possible that they are equally suitable for a given purpose. In this section, we formalise conditions that describe various levels of semantic correspondence between two theories, and we discuss when these correspondences can be relevant in practice.

The most well-known notion of semantic correspondence is semantic equivalence: two theories of first-order logic are *semantically equivalent* (or simply *equivalent*) if they have the same models. This very strong condition also implies that equivalent theories have exactly the same logical consequences, and thus represent exactly the same knowledge in terms of formal knowledge representation. Semantic equivalence in first-order logic is also a modular property in the following sense. Given a theory T with a subtheory $T_1 \subseteq T$ such that T_1 is equivalent to T_2, we find that T is equivalent to $(T \setminus T_1) \cup T_2$. A typical application of semantic equivalence are syntactic transformations on logical theories, e.g. when replacing $(p \to q)$ by $(\neg p \vee q)$. It is common to extend the notion of equivalence to (sub)formulae, and we can thus state that the latter two formulae are semantically equivalent.

A much weaker form of correspondence is equisatisfiability: two theories are *equisatisfiable* if they are either both satisfiable or both unsatisfiable. Obviously, equivalent theories are also equisatisfiable, while the converse is not true. Indeed, equisatisfiability provides only a very loose correspondence between theories, and it certainly does not preserve logical consequences. For example, every logical theory is equisatisfiable to $\{\}$ (the empty theory) or to $\{p \wedge \neg p\}$ (an inconsistent theory). Equisatisfiability thus is not useful for exchanging formally encoded knowledge, but rather for devising algorithms for satisfiability checking. If an inference engine is only interested in a theory's satisfiability then it is viable to apply satisfiability-preserving transformations to simplify the problem, even if semantic equivalence is not preserved. Common inference tasks such as query answering or entailment checking can often be reduced to satisfiability checking, so that equisatisfiability plays an important rôle in many inferencing algorithms.

Equivalence and equisatisfiability constitute the two main types of correspondences that are typically considered in formal logic. This classification of semantic correspondences, however, is arguably too coarse for capturing various levels of semantic similarity. In particular, many syntactical transformations introduce auxiliary signature symbols that are not used in any of the considered theories – we will typically call such symbols *fresh*. As a classical example, the *Skolemisation* of the formula $\exists x.P(c, x)$ is the formula $P(c, s_c)$, where s_c is a fresh (Skolem) constant. It is well-known that the original formula and the Skolemised version are equisatisfiable, but the same could be said for the empty theory. A more accurate description of the situation would be to say that both theories are "semantically equivalent up to the interpretation of s_c" – this is the idea underlying the next definition that is closely related to the well-known concept of a *conservative extension*.

Definition 2.2.1 Given $\mathbf{FOL}_{\approx}$ theories T and T' with signatures $\mathscr{S}$ and $\mathscr{S}'$, then T' *semantically emulates* T if

(1) $\mathscr{S}'$ extends $\mathscr{S}$, i.e. the sets of constants, functions, predicates, and variables of $\mathscr{S}'$ are (not necessarily proper) supersets of the respective sets of $\mathscr{S}$,

(2) every model of T' becomes a model of T when restricted to the interpretations of symbols from $\mathscr{S}$, and

(3) for every model $\mathcal{J}$ of T there is a model $\mathcal{I}$ of T' that has the same domain as $\mathcal{J}$, and that coincides with $\mathcal{J}$ on all symbols of $\mathscr{S}$. ◇

Note that, in contrast to equivalence and equisatisfiability, semantic emulation is not a symmetric relation, since one of the theories introduces additional "internal" symbols to its signature. It would be possible to establish more general notions that are based on arbitrary incomplete mappings between two signatures, but we found the basic definition above to be adequate to cover a large amount of semantic correspondences that occur within this work. It is usually not necessary to mention the signatures of T and T' explicitly, since it is always possible to find minimal signatures for T and T' that satisfy condition (1) of Definition 2.2.1.

Our notion of semantic emulation closely relates to the well-known concept of *semantic conservative extensions*: one could indeed say that T' semantically emulates T iff T' is semantically conservative over T. We use another terminology herein since it is more naturally extended to related concepts below, and since it avoids confusion with a stricter version of conservative extension that assumes a theory to be a (syntactic) superset of the theory it extends conservatively [LWW07].

Given a situation as in Definition 2.2.1, we find that a first-order formula φ over $\mathscr{S}$ is a logical consequence of T if and only if it is a logical consequence of T'. This illustrates how strong this form of correspondence is, and it hints at the practical relevance of this condition for knowledge representation: whenever a theory T' semantically emulates a theory T, we find that T' and T encode the same information *about the symbols* in T, and in particular that T' cannot be distinguished from T in any application that restricts to those symbols. In a sense, T' thus really "simulates" the behaviour of T in arbitrary contexts, but possibly by means of rather different syntactic structures.[1] If the required "interface" is restricted not only to a particular set of symbols but also to a particular logic, then the following definition may seem more natural.

Definition 2.2.2 Let T and T' be two $\mathbf{FOL}_\approx$ theories, let $\mathscr{S}$ be the signature over which T is defined, and let $\mathcal{L}$ be some fragment of $\mathbf{FOL}_\approx$. We say that T' *$\mathcal{L}$-emulates* T if for every $\mathcal{L}$ formula φ over $\mathscr{S}$, we find that $T' \cup \{\varphi\}$ and $T \cup \{\varphi\}$ are equisatisfiable. ◇

In particular, this provides us with a notion of $\mathbf{FOL}_\approx$-emulation that describes a situation where two theories behave equivalent in the context of any first-order theory over the given signature, thus coinciding with the well-known notion of *conservative extension*. To avoid confusion, formal results will always be explicit about the intended type of emulation, although we will sometimes speak of "emulation" to refer to semantic emulation in informal discussions. It is not hard to see that semantic emulation implies $\mathbf{FOL}_\approx$-emulation.

Proposition 2.2.3 *For any fragment $\mathcal{L}$ of first-order logic with equality and theories T and T', if T' semantically emulates T then T' $\mathcal{L}$-emulates T.*

Proof. It suffices to show the claim for the case that $\mathcal{L}$ is $\mathbf{FOL}_\approx$. Consider two theories T' and T such that T' semantically emulates T. We need to show that T' $\mathbf{FOL}_\approx$-emulates T. A simple induction on the structure of $\mathbf{FOL}_\approx$ formulae

[1] We generally avoid the term "simulation" here since it is already common in the context of model-theoretic relationships in modal logic [BvBW06].

can be used to show that the validity of a $\mathbf{FOL}_{\approx}$ formula φ w.r.t. any first-order interpretation is independent of the interpretation of the signature elements not occurring in φ (†). To show the claim, suppose the conditions of Definition 2.2.1 hold but T does not $\mathbf{FOL}_{\approx}$-emulate T'. Hence, there is a $\mathbf{FOL}_{\approx}$ formula φ over $\mathscr{S}$ such that $T \cup \{\varphi\}$ and $T' \cup \{\varphi\}$ are not equisatisfiable. However, if $T \cup \{\varphi\}$ has some model $\mathcal{I}$, then we can apply condition (3) of Definition 2.2.1 to obtain an extended model $\mathcal{I}'$ such that $\mathcal{I}' \models T'$. But since φ contains only symbols that are interpreted in the same way by $\mathcal{I}$ and $\mathcal{I}'$, we obtain $\mathcal{I}' \models \varphi$ from (†). Conversely, if $T' \cup \{\varphi\}$ has a model $\mathcal{J}$, then condition (2) implies that the restriction $\mathcal{I}$ of $\mathcal{J}$ to the signature of T is such that $\mathcal{J} \models T$. As before, (†) implies $\mathcal{J} \models \varphi$. □

For completeness, we also show that semantic emulation is strictly stronger that $\mathbf{FOL}_{\approx}$-emulation in general. Establishing this result requires some form of existential statements, and indeed semantic emulation and $\mathbf{FOL}_{\approx}$-emulation coincide on universal formulae that do not include function symbols [Sch09a].

Proposition 2.2.4 *There are signatures $\Sigma_0 \subseteq \Sigma_1$ and sets T_i of sentences over Σ_i such that T_1 $\mathbf{FOL}_{\approx}$-emulates T_0, and T_1 does not semantically emulate T_0.*

Proof. Let Σ_0 be a signature containing a binary predicate R, nullary function symbol 0, and a unary function symbol f. Let Σ_1 denote the extension of Σ_0 that additionally contains a nullary function symbol ω and a unary predicate symbol B.

Now let T_0 denote the set of the following sentences:

(1) $\forall x.R(x, f(x))$
(2) $\forall x.\forall y.\forall z.R(x,y) \wedge R(y,z) \rightarrow R(x,z)$
(3) $\forall x.\neg R(x,x)$

Let T_1 denote the set of sentences with $T_0 \subseteq T_1$, and containing the following additional sentences:

(4) $B(0)$
(5) $\forall x.B(x) \rightarrow B(f(x))$
(6) $\neg B(\omega)$

For the first part of the claim, consider an arbitrary first-order sentence φ over Σ_0. The claim is established by showing that $T_0 \cup \{\varphi\}$ is satisfiable iff $T_1 \cup \{\varphi\}$ is. The "if" direction is immediate from $T_0 \subseteq T_1$. For the "only if" direction, we show that every model of $T_0 \cup \{\varphi\}$ can be extended to a model of $T_1 \cup \{\varphi\}$.

Let S denote the infinite set of Σ_1 sentences $S := \{R(f^i(0), \omega) \mid i \geq 0\}$ where f^i denotes the i-fold application of f (with $f^0(0) = 0$). Then $T_0 \cup \{\varphi\} \cup S$ is satisfiable by models over Σ_1. To see this, note that $T_0 \cup \{\varphi\}$ is satisfiable over Σ_1 (by the Coincidence Lemma) and that the interpretation of ω is arbitrary for the according models. Therefore, for any finite set $F \subseteq S$, there is a model $\mathcal{M}_F = (M, I)$ of

$T_0 \cup \{\varphi\}$ with $\omega^I = (f^{k+1}(0))^I$ where $k = \max_i(R(f^i(0), \omega) \in F)$. But then $\mathcal{M}_F$ is also a model of $T_0 \cup \{\varphi\} \cup F$. By compactness of first-order logic [CK90], we conclude that $T_0 \cup \{\varphi\} \cup S$ is also satisfiable.

Thus, let $\mathcal{N} = (N, J)$ be a model of $T_0 \cup \{\varphi\} \cup S$. Since B does not occur in $T_0 \cup \{\varphi\} \cup S$, we can select $\mathcal{N}$ such that $B^J = \{(f^i(0))^I \mid i \geq 0\}$. We claim that $\mathcal{N}$ is a model of $T_1 \cup \{\varphi\}$. By construction, it satisfies T_0, φ, and the formulae (4) and (5) of T_1. To see that it also satisfies formula (6), it suffices to note that $\omega^J \neq (f^i(0))^J$ for all $i \geq 0$, which can be shown by a simple induction over i using the fact that $\mathcal{N} \models S$. We thus constructed a model of $T_1 \cup \{\varphi\}$ as required.

For the second part of the claim, let $\mathcal{M} = (M, I)$ be the structure with $M = \{i \mid i \geq 0\}$ and $f^I(i) = i + 1$ and $R^I = \{(i, j) \mid i < j\}$. It is easy to see that $\mathcal{M}$ is a model of T_0. However, $\mathcal{M}$ cannot be expanded to a model of T_1 since for (4) and (5) together imply that $B^I = M$ so that (6) cannot be satisfied. □

In many cases that are considered herein, it is possible to establish semantic emulation between two theories. There are, however, also interesting examples of transformation procedures that establish $\mathcal{L}$-emulation for some logical fragment $\mathcal{L}$ that is significantly smaller than $\mathbf{FOL}_{\approx}$. For example, a typical result is that two theories entail the same *ground facts*, i.e. atomic formulae without variable symbols, even though they may not be semantically equivalent. This correspondence extends to arbitrary Boolean combinations of ground facts, i.e. to all formulae of variable-free first-order logic $\mathbf{FOL}_{\approx}^{\text{ground}}$. In this work, examples of transformations that establish $\mathbf{FOL}_{\approx}^{\text{ground}}$-emulation can be found in Section 5.4 and in Section 8.5.

2.3 Computational Complexity

Giving an introduction to computational complexity is beyond the scope of this work, and interested readers are referred to [Pap94] for an extensive textbook treatment. In this section, we merely point out some basic assumptions, and introduce the main complexity classes that appear in later chapters.

Within this work, complexity is always considered as a characteristic of a class of *decision problems* (as opposed, e.g., to counting problems), which in our case will typically relate to an inference task. The complexity of a class of problems is measured in terms of the amount of certain resources that are required to solve problems of that class based on a certain abstract computational model. The classical model of computation used in this context is the *Turing machine* – we will encounter deterministic, non-deterministic, and alternating specimen in this work – and the most common types of resources are time (the number of computation steps needed) and space (the number of memory cells that are used).

Any single problem is trivially solved by a suitable Turing machine without using any resources, by simply returning the answer to that problem as a constant output. Hence, one normally considers infinite classes of problems and general approaches for solving them. In this case, the required amount of resources typically depends on the size of the input problem. When speaking of the *size of a logical theory* we simply refer to the minimal number of symbols that is required to write this theory in the alphabet provided by its signature, the additional logical operators, and auxiliary symbols such as parentheses. Description logics (see Chapter 3) also include numbers; unless otherwise noted, we assume them to be written in binary notation when calculating the size of a theory.[2]

The complexity classes considered in this work mainly are P, NP, PSpace, ExpTime, NExpTime, and N2ExpTime. It is known that these classes subsume each other in the given order, e.g. all problems in P are also in NP, while it is unknown whether or not any of these (direct) inclusions is strict, although this is commonly conjectured. It is known, however, that P $\subsetneq$ ExpTime, NP $\subsetneq$ NExpTime, and NExpTime $\subsetneq$ N2ExpTime. In any case, experience shows that problems of higher complexity classes are often significantly harder to implement efficiently in practice.

Roughly speaking, a class of problems is *hard* for another class of problems if any problem of the second class can be solved by reducing it to a problem of the first class, and where this reduction is “significantly easier” than solving the problem directly. Since the overwhelming majority of complexities that are studied within this work are above NP, we will mostly consider polynomial-time reductions for showing hardness. To establish hardness for P, reductions must be restricted to those running in LogSpace, but this will rarely be required and usually be easy to verify.

Further formal definitions, such as the specification of relevant Turing machines, are provided within the respective sections.

[2]The use of unary encoding of numbers increases the size of the input exponentially, and hence may have significant effect on complexity measures; however, most results that we will use have by now been established for binary coding of numbers.

Chapter 3

Introduction to Description Logics

The basic expressive features of description logics have already been introduced in Section 1.2. In this chapter, we provide a more formal introduction to the field, focussing on the very expressive DL $\mathcal{SROIQ}$ that provides a basis for many of our subsequent investigations. While this chapter provides a sufficient background for understanding the remaining parts of this work, there are also a number of more extensive treatments of description logics available that the reader may want to consult for a more easy-paced introduction. In particular, [BCM$^+$07] provides introductory and advanced material on many aspects of DL research, while a textbook introduction to description logics in the context of Semantic Web technologies can be found in [HKR09]. The latter also explains the exact relationship between DL and OWL (2) that is not detailed here.

Section 3.1 begins this chapter by introducing the syntax and semantics of $\mathcal{SROIQ}$, and by discussing simplifications and normal forms that are relevant within this work. The relationship of $\mathcal{SROIQ}$ to various other logics, especially to first-order logic with equality, is explicated in Section 3.2. Based on these considerations, we can then derive a number of other DLs and their names as explained in Section 3.3.

3.1 The Description Logic $\mathcal{SROIQ}$

We now formally define the syntax and semantics of the widely used description logic $\mathcal{SROIQ}$ that is the basis for many investigations within this work. $\mathcal{SROIQ}$ requires a number of additional structural restrictions to ensure that standard reasoning problems remain decidable. Since those restrictions are not relevant in all DLs that are considered in this work, we first define a more general description logic $\mathcal{SROIQ}^{\text{free}}$ to which no such restrictions apply.

3.1.1 Syntax

$\mathcal{SROIQ}^{\text{free}}$ and all other DLs considered herein are based on three disjoint sets of *individual names* **I**, *concept names* **A**, and *role names* **N**. Throughout this work, we assume that these basic sets are finite, and consider them to be part of the given knowledge base when speaking about the "size of a knowledge base." We further assume **N** to be the union of two disjoint sets of *simple roles* $\mathbf{N}_\mathrm{s}$ and *non-simple roles* $\mathbf{N}_\mathrm{n}$. Later on, the use of simple roles in conclusions of logical axioms will be restricted to ensure, intuitively speaking, that relationships of these roles are not implied by *chains* of other role relationships. In exchange, simple roles might be used in $\mathcal{SROIQ}$ axioms where non-simple roles might lead to undecidability.

The approach we take here assumes an *a priori* declaration of simple and non-simple role names. A common alternative approach is to derive a maximal set of simple roles from the structure of a given DL knowledge base. This *a posteriori* approach of determining the sets $\mathbf{N}_\mathrm{n}$ or $\mathbf{N}_\mathrm{s}$ is more adequate in practical applications where it is often not viable to declare simplicity of roles in advance. Especially if ontologies are dynamic, simplicity of roles may need to be changed over time to suit the overall structure of axioms. For the investigation of theoretical properties, however, pre-supposing complete knowledge about the names of simple and non-simple roles can simplify many definitions significantly.

Definition 3.1.1 Consider a DL signature $\mathscr{S} = \langle \mathbf{I}, \mathbf{A}, \mathbf{N}\rangle$ with $\mathbf{N} = \mathbf{N}_\mathrm{s} \cup \mathbf{N}_\mathrm{n}$. The set **R** of $\mathcal{SROIQ}^{\text{free}}$ *role expressions* (or simply *roles*) for $\mathscr{S}$ is defined by the following grammar:

$$\mathbf{R} ::= U \mid \mathbf{N} \mid \mathbf{N}^-$$

where U is called the *universal role*. The set $\mathbf{R}_\mathrm{s} \subseteq \mathbf{R}$ of all *simple role expressions* is defined to contain all role expressions that contain no non-simple role names. The set $\mathbf{R}_\mathrm{n}$ of *non-simple role expressions* is $\mathbf{R}_\mathrm{n} := \mathbf{R} \setminus \mathbf{R}_\mathrm{s}$. A bijective function $\mathrm{Inv} : \mathbf{R} \to \mathbf{R}$ is defined by setting $\mathrm{Inv}(R) := R^-$, $\mathrm{Inv}(R^-) := R$, and $\mathrm{Inv}(U) := U$ for all $R \in \mathbf{N}$.

The set **C** of $\mathcal{SROIQ}^{\text{free}}$ *concept expressions* (or simply *concepts*) for $\mathscr{S}$ is defined by the following grammar:

$$\mathbf{C} ::= \top \mid \bot \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \neg\mathbf{C} \mid (\mathbf{C} \sqcap \mathbf{C}) \mid (\mathbf{C} \sqcup \mathbf{C}) \mid \forall\mathbf{R}.\mathbf{C} \mid \exists\mathbf{R}.\mathbf{C} \mid \geqslant n\,\mathbf{R}.\mathbf{C} \mid \leqslant n\,\mathbf{R}.\mathbf{C}$$

where n is a non-negative integer. ◇

Concepts are used to model classes while roles represent binary relationships. In some application areas of description logics, especially in relation to the Web Ontology Language OWL, "class" is used as a synonym for "concept." We will reserve the former notion for talking about syntactic constructs, and use the latter for semantic considerations only. For example, a *subconcept* is a substring of a concept expression that is again a concept, while a *subclass* is a class that is semantically subsumed by another class, i.e. that describes a subset of instances. Similarly, it is also common to use the term “property” as a synonym for “role” in some contexts, but we will not make use of this terminology in this work.

Parentheses are typically omitted if the exact structure of a given concept expression is clear or irrelevant. Also, we will commonly assume a signature and according sets of concept and role expressions to be given using the notation of Definition 3.1.1, mentioning it explicitly only to distinguish multiple signatures if necessary. Using these conventions, role and concept expressions can be combined into axioms:

Definition 3.1.2 A $\mathcal{SROIQ}^{\text{free}}$ *RBox axiom* is an expression of one of the following forms:

- $R_1 \circ \ldots \circ R_n \sqsubseteq R$ where $R_1, \ldots, R_n, R \in \mathbf{R}$ and where $R \notin \mathbf{R}_{\text{n}}$ only if $n = 1$ and $R_1 \in \mathbf{R}_{\text{s}}$,
- $\mathsf{Ref}(R)$ (reflexivity), $\mathsf{Tra}(R)$ (transitivity), $\mathsf{Irr}(R)$ (irreflexivity), $\mathsf{Dis}(R, R')$ (role disjointness), $\mathsf{Sym}(R)$ (symmetry), $\mathsf{Asy}(R)$ (asymmetry), where $R, R' \in \mathbf{R}$.

A $\mathcal{SROIQ}^{\text{free}}$ *TBox axiom* is an expression of the form $C \sqsubseteq D$ or $C \equiv D$ with $C, D \in \mathbf{C}$. A $\mathcal{SROIQ}^{\text{free}}$ *ABox axiom* is an expression of the form $C(a)$, $R(a, b)$, or $a \approx b$ where $C \in \mathbf{C}$, $R \in \mathbf{R}$, and $a, b \in \mathbf{I}$. ◇

RBox axioms of the form $R_1 \circ \ldots \circ R_n \sqsubseteq R$ are also known as *role inclusion axioms* (RIAs), and a RIA is said to be *complex* if $n > 1$. Expressions such as $\mathsf{Ref}(R)$ are called *role characteristics*. Note that, in our formulation, the universal role U is introduced as a constant (or nullary operator) on roles, and not as a “special” role name. In effect, $U \in \mathbf{R} \setminus \mathbf{N}_{\text{n}}$ and in particular $U \in \mathbf{R}_{\text{s}}$. Treating U as a simple role deviates from earlier works on $\mathcal{SROIQ}$, but it can be shown that U can typically be allowed in axioms that are often restricted to simple roles (cf. Definition 3.1.4) without leading to undecidability or increased worst-case complexity of reasoning; see Chapter 5 for details. TBox axioms are also known as *terminological axioms* or *schema axioms*, and expressions of the form $C \sqsubseteq D$ are known as *generalised concept inclusions* (GCIs). ABox axioms are also called *assertional axioms*, where axioms $C(a)$ are *concept assertions*, axioms $R(a, b)$ are *role assertions*, and axioms $a \approx b$ are *equality assertions*.

Many of the above types of axioms can be expressed in terms of other axioms, so that substantial syntactic simplifications are possible in many DLs. Relevant abbreviations are discussed in Section 3.1.3 below. Logical theories in description logic are called *knowledge bases*:

Definition 3.1.3 A $\mathcal{SROIQ}^{\text{free}}$ *RBox* (*TBox*, *ABox*) is a set of $\mathcal{SROIQ}^{\text{free}}$ RBox axioms (TBox axioms, ABox axioms). A $\mathcal{SROIQ}^{\text{free}}$ knowledge base is the union of a (possibly empty) $\mathcal{SROIQ}^{\text{free}}$ RBox, TBox, and ABox. ◇

The above definitions still disregard some additional restrictions that are relevant for ensuring decidability of common reasoning tasks. The next definition therefore introduces $\mathcal{SROIQ}$ as a decidable sublanguage of $\mathcal{SROIQ}^{\text{free}}$.

Definition 3.1.4 A $\mathcal{SROIQ}$ role expression is a $\mathcal{SROIQ}^{\text{free}}$ role expression. A $\mathcal{SROIQ}$ concept expression C is a $\mathcal{SROIQ}^{\text{free}}$ concept expression such that all subconcepts D of C that are of the form $\exists S.\mathsf{Self}$, $\geqslant n\, S.E$, or $\leqslant n\, S.E$ are such that $S \in \mathbf{R}_{\text{s}}$ is simple.

A $\mathcal{SROIQ}^{\text{free}}$ RBox is *regular* if there is a strict (irreflexive) total order $\prec$ on $\mathbf{R}$ such that

- for $R \notin \{S, \text{Inv}(S)\}$, we find $S \prec R$ iff $\text{Inv}(S) \prec R$, and
- every RIA is of one of the forms:

$$R \circ R \sqsubseteq R, \qquad \text{Inv}(R) \sqsubseteq R,$$

$$R_1 \circ \ldots \circ R_n \sqsubseteq R, \quad R \circ R_1 \circ \ldots \circ R_n \sqsubseteq R, \quad R_1 \circ \ldots \circ R_n \circ R \sqsubseteq R$$

  such that $R, R_1, \ldots, R_n \in \mathbf{R}$, and $R_i \prec R$ for $i = 1, \ldots, n$.

A $\mathcal{SROIQ}$ RBox is a regular $\mathcal{SROIQ}^{\text{free}}$ RBox that contains role characteristics of the forms $\mathsf{Irr}(S)$, $\mathsf{Dis}(S, T)$, and $\mathsf{Asy}(S)$ only for simple role names $S, T \in \mathbf{N}_{\text{s}}$. A $\mathcal{SROIQ}$ TBox (ABox) is a $\mathcal{SROIQ}^{\text{free}}$ TBox (ABox) that contains only $\mathcal{SROIQ}$ concept expressions. A $\mathcal{SROIQ}$ knowledge base is the union of a $\mathcal{SROIQ}$ RBox, TBox, and ABox. ◇

A $\mathcal{SROIQ}$ (RBox, TBox, or ABox) axiom is an axiom that occurs within some $\mathcal{SROIQ}$ knowledge base (in the RBox, TBox, or ABox). Note that some $\mathcal{SROIQ}^{\text{free}}$ role inclusion axioms like, e.g., $R \circ S \circ R \sqsubseteq R$ cannot be part of any regular RBox.

3.1.2 Semantics and Inferencing

The semantics of description logics is typically specified by providing a model theory from which notions like logical consistency and entailment can be derived in the usual way. These notions are again specified for the most general case of $\mathcal{SROIQ}^{\text{free}}$ but they can readily be applied to $\mathcal{SROIQ}$ as well. The basis for this approach is the definition of a DL interpretation:

Definition 3.1.5 An *interpretation* $\mathcal{I}$ for a $\mathcal{SROIQ}^{\text{free}}$ signature $\mathscr{S} = \langle \mathbf{I}, \mathbf{A}, \mathbf{N} \rangle$ is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a mapping with the following properties:

Name	Syntax	Semantics
inverse role	R^-	$\{\langle x,y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle y,x\rangle \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
nominals	$\{a\}$	$\{a^\mathcal{I}\}$
univ. restriction	$\forall R.C$	$\{x \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in R^\mathcal{I}$ implies $y \in C^\mathcal{I}\}$
exist. restriction	$\exists R.C$	$\{x \in \Delta^\mathcal{I} \mid$ for some $y \in \Delta^\mathcal{I}$, $\langle x,y\rangle \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\}$
Self concept	$\exists S.\mathsf{Self}$	$\{x \in \Delta^\mathcal{I} \mid \langle x,x\rangle \in S^\mathcal{I}\}$
qualified number	$\leqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I}$ and $y \in C^\mathcal{I}\} \leq n\}$
restriction	$\geqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I}$ and $y \in C^\mathcal{I}\} \geq n\}$

Figure 3.1: Semantics of role and concept expressions in $\mathcal{SROIQ}^{\text{free}}$ for an interpretation $\mathcal{I}$ with domain $\Delta^\mathcal{I}$

- if $a \in \mathbf{I}$ then $a^\mathcal{I} \in \Delta^\mathcal{I}$,
- if $A \in \mathbf{A}$ then $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$,
- if $R \in \mathbf{N}$ then $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.

The mapping $\cdot^\mathcal{I}$ is extended to arbitrary role and concept expressions as specified in Fig. 3.1. ◇

The set $\Delta^\mathcal{I}$ is called the *domain* of $\mathcal{I}$. We often do not mention an interpretation's signature $\mathscr{S}$ explicitly if it is irrelevant or clear from the context. We can now define when an interpretation is a model for some DL axiom.

Definition 3.1.6 Given an interpretation $\mathcal{I}$ and a $\mathcal{SROIQ}^{\text{free}}$ (RBox, TBox, or ABox) axiom α, we say that $\mathcal{I}$ *satisfies* (or *models*) α, written $\mathcal{I} \models \alpha$, if the respective conditions of Fig. 3.2 are satisfied. $\mathcal{I}$ *satisfies* (or *models*) a $\mathcal{SROIQ}^{\text{free}}$ knowledge base KB, denoted as $\mathcal{I} \models$ KB, if it satisfies all of its axioms. In these situations, we also say that $\mathcal{I}$ is a *model* of the given axiom or knowledge base. ◇

This allows us to derive standard model-theoretic notions as follows:

Definition 3.1.7 Consider $\mathcal{SROIQ}^{\text{free}}$ knowledge bases KB and KB′.

- KB is *consistent* (satisfiable) if it has a model and *inconsistent* (unsatisfiable) otherwise,

Axiom α	Condition for $\mathcal{I} \models \alpha$
$R_1 \circ \ldots \circ R_n \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
$\mathsf{Tra}(R)$	if $R^{\mathcal{I}} \circ R^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
$\mathsf{Ref}(R)$	$\langle x, x\rangle \in R^{\mathcal{I}}$ for all $x \in \Delta^{\mathcal{I}}$
$\mathsf{Irr}(S)$	$\langle x, x\rangle \notin S^{\mathcal{I}}$ for all $x \in \Delta^{\mathcal{I}}$
$\mathsf{Dis}(S,T)$	if $\langle x, y\rangle \in S^{\mathcal{I}}$ then $\langle x, y\rangle \notin T^{\mathcal{I}}$ for all $x, y \in \Delta^{\mathcal{I}}$
$\mathsf{Sym}(R)$	if $\langle x, y\rangle \in R^{\mathcal{I}}$ then $\langle y, x\rangle \in R^{\mathcal{I}}$ for all $x, y \in \Delta^{\mathcal{I}}$
$\mathsf{Asy}(S)$	if $\langle x, y\rangle \in S^{\mathcal{I}}$ then $\langle y, x\rangle \notin S^{\mathcal{I}}$ for all $x, y \in \Delta^{\mathcal{I}}$
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$R(a,b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$
$a \approx b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
$\circ$ on the right-hand side denotes standard composition of binary relations: $R^{\mathcal{I}} \circ T^{\mathcal{I}} := \{\langle x, z\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}, \langle y, z\rangle \in T^{\mathcal{I}}\}$	

Figure 3.2: Semantics of $\mathcal{SROIQ}^{\text{free}}$ axioms for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$

- KB *entails* KB′, written KB $\models$ KB′, if all models of KB are also models of KB′.

This terminology is extended to axioms by treating them as singleton knowledge bases. A knowledge base or axiom that is entailed is also called a *logical consequence*. ◇

Applying this terminology, we can state, e.g., that the axiom $\top \sqsubseteq \bot$ is inconsistent, and that the knowledge base $\{A \sqsubseteq B, B \sqsubseteq C\}$ entails the axiom $A \sqsubseteq C$. Various common properties of first-order logic are readily seen to hold for description logics as well. DLs are monotonic logics: the more axioms a knowledge base contains, the less models it has, and the more axioms are logical consequences. In other words, adding information never reduces the amount of logical consequences. A related property is the general intolerance to logical inconsistencies: a knowledge base that has no models entails all possible axioms.

When description logics are applied as an ontology modelling language, it is important to discover logical consequences. The (typically automatic) process of deriving logical consequences is called reasoning or inferencing, and a number of standard reasoning tasks play a central rôle in DLs.

Definition 3.1.8 Consider a $\mathcal{SROIQ}^{\text{free}}$ knowledge base KB. The *standard reasoning tasks* of description logics are described as follows:

- *Inconsistency checking:* Is KB inconsistent?

- *Concept subsumption:* Given concepts C, D, does $\mathrm{KB} \models C \sqsubseteq D$ hold?
- *Instance checking:* Given a concept C and individual name a, does $\mathrm{KB} \models C(a)$ hold?
- *Concept unsatisfiability:* Given a concept C, is there no model $\mathcal{I} \models \mathrm{KB}$ such that $C^{\mathcal{I}} \neq \emptyset$? ◇

Further reasoning tasks are considered as "standard" in some works. Common problems include instance retrieval (finding *all* instances of a concept) and classification (computing *all* subsumptions between concept names). We restrict our selection here to ensure that all standard reasoning tasks can be viewed as decision problems that have a common worst-case complexity for all logics studied within this work.

Proposition 3.1.9 *The standard reasoning tasks in $\mathcal{SROIQ}^{\text{free}}$ can be reduced to each other in linear time, and this is possible in any fragment of $\mathcal{SROIQ}^{\text{free}}$ that includes axioms of the form $A(a)$ and $A \sqcap C \sqsubseteq \bot$.*

Proof. We find that KB is inconsistent if the concept $\top$ is unsatisfiable. C is unsatisfiable in KB if $\mathrm{KB} \models C \sqsubseteq \bot$. Given a fresh individual name a, we obtain $\mathrm{KB} \models C \sqsubseteq D$ if $\mathrm{KB} \cup \{C(a)\} \models D(a)$. For a fresh concept name A, $\mathrm{KB} \models C(a)$ if $\mathrm{KB} \cup \{A(a), A \sqcap C \sqsubseteq \bot\}$ is inconsistent. This cyclic reduction shows that all reasoning problems can be reduced to one another. □

A number of other "non-standard" reasoning tasks have been studied in description logics. Examples include the computation of explanations for logical entailments [Kal06, HPS08], and of least common subsumer concepts that generalise given concept expressions in description logics where union of concepts is not available [Baa03, BST07]. Another practically relevant inference tasks is conjunctive query answering, as discussed in Section 4.4.

3.1.3 Simplifications and Normal Forms

Description logics have a very rich syntax that often provides many different ways of expressing equivalent statements. In this section, we introduce a number of simplifications and normal form transformations that allow us to simplify subsequent presentations. We start by considering simplification for TBox axioms and conclude with remarks on simplification of RBox and ABox axioms.

Every $\mathcal{SROIQ}^{\text{free}}$ GCI $C \sqsubseteq D$ can be expressed as $\top \sqsubseteq \neg C \sqcup D$, i.e. by stating that the concept $\neg C \sqcup D$ is universally valid. In the following, we will often tacitly assume that GCIs are expressed as universally valid concepts, and we will use concept expressions C to express axioms $\top \sqsubseteq C$. Nonetheless, we still use $\sqsubseteq$ whenever this notation appears to be more natural for a given purpose. Likewise, we consider $C \equiv D$ as an abbreviation for $\{C \sqsubseteq D, D \sqsubseteq C\}$, and omit $\equiv$ as an atomic constructor for axioms.

It is well known that many DL constructs can be considered as "syntactic sugar" in the sense that they can readily be expressed in terms of other operators. Examples are found by applying basic propositional equivalences such as $A \sqcup B \equiv \neg(\neg A \sqcap \neg B)$ or $\top \equiv A \sqcup \neg A$. These simplifications are applicable when dealing with DLs that are characterised by a set of operators which can freely be combined to form concept expressions. In this work, however, we derive more complex syntactic restrictions to arrive at DLs that are not closed under typical propositional equivalences – concrete examples are found in Chapter 6 and 7. We thus do not exclude any operators from our considerations at this stage, and will introduce simplifications based on applicable equivalences in later chapters only.

There still are some general simplifications that we can endorse for all parts of this work:

- Whenever a DL features counting quantifiers, we use $\geqslant 1\, R.C$ instead of $\exists R.C$, and $\leqslant 0\, R.\neg C$ instead of $\forall R.C$.
- We exploit commutativity and associativity of $\sqcap$, as given by the equivalences $A \sqcap B \equiv B \sqcap A$ and $A \sqcap (B \sqcap C) \equiv (A \sqcap B) \sqcap C$, to generally disregard nesting and ordering of conjuncts. For example, "a concept of the form $\exists R.A \sqcap C$ with C arbitrary" is used to refer to concept expressions $B \sqcap \exists R.A$ ($C = B$) or $B \sqcap (B' \sqcap \exists R.A)$ ($C = B \sqcap B'$). This convention introduces some non-determinism, e.g. if $B' = \exists R.A$ in the previous example, but the choice will never be essential in our arguments.
- We exploit commutativity and associativity of $\sqcup$ as in the case of $\sqcap$.

These conventions greatly reduce the amount of cases that need to be considered in definitions. All additional simplifying assumptions will be stated explicitly when applicable, usually by syntactically transforming axioms. For example, we do respect the nesting structure of $\sqcap$ and $\sqcup$, but we define a normal form transformation that exploits distributivity to normalise axioms. Namely, the *disjunctive normal form* (DNF) of a concept C is obtained by exhaustively replacing subconcepts of the form $(C \sqcup D) \sqcap E$ with $(C \sqcap E) \sqcup (D \sqcap E)$. Note that we do not distribute Boolean concept constructors over role restrictions, i.e. our DNF may still contain complex nested concepts. Moreover, this simple transformation may lead to an exponential blow-up in the size of the axiom. It is well-known that this can be prevented by decomposing nested concepts first, but we will not require this for the cases where we use the disjunctive normal form.

Another well-known normal form is the negation normal form (NNF). While this standard transformation normalises the uses of negation in concept expressions, it does often not contribute significantly to a simplified presentation. The

C	$\mathsf{NNF}(C)$	$\mathsf{pNNF}(C)$
$\top, \bot, A, \neg A, \{a\}, \neg\{a\}$, $\exists R.\mathsf{Self}, \neg\exists R.\mathsf{Self}$	C	C
$\neg\top$	$\bot$	$\bot$
$\neg\bot$	$\top$	$\top$
$\neg\neg D$	$\mathsf{NNF}(D)$	$\mathsf{pNNF}(D)$
$D_1 \sqcap D_2$	$\mathsf{NNF}(D_1) \sqcap \mathsf{NNF}(D_2)$	$\mathsf{pNNF}(D_1) \sqcap \mathsf{pNNF}(D_2)$
$D_1 \sqcup D_2$	$\mathsf{NNF}(D_1) \sqcup \mathsf{NNF}(D_2)$	$\mathsf{pNNF}(D_1) \sqcup \mathsf{pNNF}(D_2)$
$\neg(D_1 \sqcap D_2)$	$\mathsf{NNF}(\neg D_1) \sqcup \mathsf{NNF}(\neg D_2)$	$\mathsf{pNNF}(\neg D_1) \sqcup \mathsf{pNNF}(\neg D_2)$
$\neg(D_1 \sqcup D_2)$	$\mathsf{NNF}(\neg D_1) \sqcap \mathsf{NNF}(\neg D_2)$	$\mathsf{pNNF}(\neg D_1) \sqcap \mathsf{pNNF}(\neg D_2)$
$\leqslant n\, R.D$	$\leqslant n\, R.\mathsf{NNF}(D)$	$\leqslant n\, R.\neg\mathsf{pNNF}(\neg D)$
$\neg\leqslant n\, R.D$	$\geqslant(n+1)\, R.\mathsf{NNF}(\neg D)$	$\geqslant(n+1)\, R.\mathsf{pNNF}(\neg D)$
$\geqslant n\, R.D$	$\geqslant n\, R.\mathsf{NNF}(D)$	$\geqslant n\, R.\mathsf{pNNF}(D)$
$\neg\geqslant 0\, R.D$	$\bot$	$\bot$
$\neg\geqslant n\, R.D \quad (n > 1)$	$\leqslant(n-1)\, R.\mathsf{NNF}(D)$	$\leqslant(n-1)\, R.\neg\mathsf{pNNF}(\neg D)$
A a concept name, a an individual name, R a role name, $D_{(i)}$ concept expressions		

Figure 3.3: Negation normal form transformations for DL concept expressions

reason is that concepts D in expressions $\leqslant n\, R.D$ also occur under a negative *polarity*, i.e. they behave like negated subexpressions; see also Section 6.1. Therefore a modified version of negation normal form is more effective for simplifying formal arguments. Here we define both notions for comparison.

Definition 3.1.10 A $\mathcal{SROIQ}^{\text{free}}$ concept expression C is in *negation normal form* (NNF) if all subconcepts $\neg D$ of C are such that D is of the form $\neg A$ (A a concept name), $\neg\{a\}$, or $\neg\exists R.\mathsf{Self}$.

A $\mathcal{SROIQ}^{\text{free}}$ concept C is in *positive negation normal form* (pNNF) if

– if $\leqslant n\, R.D$ is a subconcept of C, then D has the form $\neg D'$, and

– every other occurrence of $\neg$ in C is part of a subconcept $\neg D$ where D is of the form $\neg A$ (A a concept name), $\neg\{a\}$, or $\neg\exists R.\mathsf{Self}$. ◇

Every concept expression C can be transformed into a semantically equivalent concepts expression $\mathsf{NNF}(C)$ ($\mathsf{pNNF}(C)$) that is in negation normal form (positive negation normal form). It is easy to see that this can be achieved in linear time using the recursive definitions of Fig. 3.3. Also note that, when using $\exists$ and $\forall$ in DLs that do not support cardinality restrictions, we find that $\mathsf{NNF}(C) = \mathsf{pNNF}(C)$ for all concepts C.

Role expressions and RBox axioms also allow for a number of simplifications. $\mathsf{Sym}(R)$ and $\mathsf{Tra}(R)$ are equivalent to $R^- \sqsubseteq R$ and $R \circ R \sqsubseteq R$, respectively.

$\mathsf{Ref}(R)$ is equivalent to $\top \sqsubseteq \exists R.\mathsf{Self}$ but the latter is not admissible in $\mathcal{SROIQ}$ if R is not simple. As an alternative, $\mathsf{Ref}(R)$ can be semantically emulated by $\{\top \sqsubseteq \exists S.\mathsf{Self}, S \sqsubseteq R\}$ where S is a fresh simple role name. Irreflexivity $\mathsf{Irr}(S)$ and asymmetry $\mathsf{Asy}(S)$ are again equivalently expressed by $\exists S.\mathsf{Self} \sqsubseteq \bot$ and $\mathsf{Dis}(S, \mathrm{Inv}(S))$, respectively. In summary, $\mathsf{Dis}(S, T)$ is the only role characteristic that is not expressible in terms of other constructs in most DLs.

Finally, a number of simplifications can be applied to ABox axioms as well. Most importantly, DLs that support nominals can typically express ABox assertions as TBox axioms by transforming axioms $C(a)$, $R(a, b)$, $a \approx b$ into $\{a\} \sqsubseteq C$, $\{a\} \sqsubseteq \exists R.\{b\}$, and $\{a\} \sqsubseteq \{b\}$, respectively. Even without this expressivity, it is possible to restrict concept assertions in ABoxes to concept names: given a knowledge base KB with an ABox axiom $C(a)$ and a fresh concept name A, the knowledge base $\mathrm{KB} \cup \{A(a), A \sqsubseteq C\} \setminus \{C(a)\}$ semantically emulates KB. We will explicitly mention if any of these simplifications is to be used in a given part of this work.

3.2 Relationship of DLs to Other Logics

Description logics have close connections to first-order logic and various fragments thereof. The early development of DLs and their formal semantics had been driven by the goal of creating a knowledge representation formalism that improves and extends semantic networks and frame-based systems (see Section 1.2). However, it was soon recognised that DLs are closely related to *modal logics* [BvBW06], which was first articulated by Schild in 1991 [Sch91]. Indeed, universal and existential quantifiers can be considered as notational variants of modal box and diamond operators, respectively. The role names in DL then distinguish distinct underlying frame structures, corresponding to a multi-modal logic with separate box and diamond for each role.

The correspondence to modal logics extends to various basic description logics, but it does not capture advanced features such as nominals, counting quantifiers, or complex role inclusions, and actually not even ABox assertions. These discrepancies have been addressed, e.g., by further extending *Propositional Dynamic Logics* (PDLs) – the modal logics studied by Schild – as discussed in [GL94]. A related approach are *hybrid logics* that extend modal logics with nominals that allow formulae to refer to individual worlds [BT98, AdR02].

Another important approach is to interpret DLs as fragments of first-order logic with equality $\mathbf{FOL}_{\approx}$. This is achieved by recursively translating DL axioms and expressions into first-order formulae, as shown in Fig. 3.4 for $\mathcal{SROIQ}^{\mathrm{free}}$. Given a $\mathcal{SROIQ}^{\mathrm{free}}$ knowledge base KB, we can then define its first-order translation $\pi(\mathrm{KB}) \coloneqq \{\pi(\alpha) \mid \alpha \in \mathrm{KB}\}$. DL role and concept names are mapped to binary and unary predicate symbols, and individual names are interpreted in the same

Role Expressions (t, u arbitrary first-order variables or individual symbols)

$$
\begin{aligned}
\pi(U,t,u) &= \top \\
\pi(R,t,u) &= R(t,u) \text{ if } R \in \mathbf{N} \\
\pi(R^-,t,u) &= \pi(R,u,t)
\end{aligned}
$$

Concept Expressions (t an arbitrary first-order variable or individual symbol)

$$
\begin{aligned}
\pi(\top,t) &= \top \\
\pi(\bot,t) &= \bot \\
\pi(A,t) &= A(t) \\
\pi(\{a\},t) &= a \approx t \\
\pi(\exists R.\mathsf{Self},t) &= \pi(R,t,t) \\
\pi(\neg C,t) &= \neg\pi(C,t) \\
\pi(C \sqcap D,t) &= \pi(C,t) \land \pi(D,t) \\
\pi(C \sqcup D,t) &= \pi(C,t) \lor \pi(D,t) \\
\pi(\forall R.C,t) &= \forall x.(\pi(R,t,x) \rightarrow \pi(C,x)) \\
\pi(\exists R.C,t) &= \exists x.(\pi(R,t,x) \land \pi(C,x)) \\
\pi(\leqslant n\, R.C,t) &= \forall x_1,\ldots,x_{n+1}.\left(\bigwedge_{i=1}^{n+1} (\pi(R,t,x_i) \land \pi(C,x_i)) \rightarrow \bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n+1} x_i \approx x_j\right) \\
\pi(\geqslant n\, R.C,t) &= \exists x_1,\ldots,x_n.\, \bigwedge_{i=1}^{n} \left(\pi(R,t,x_i) \land \pi(C,x_i) \land \bigwedge_{j=i+1}^{n} x_i \not\approx x_j\right)
\end{aligned}
$$

ABox Axioms

$$
\begin{aligned}
\pi(C(a)) &= \pi(C,a) \\
\pi(R(a,b)) &= \pi(R,a,b) \\
\pi(a \approx b) &= a \approx b
\end{aligned}
$$

TBox Axioms

$$
\begin{aligned}
\pi(C \sqsubseteq D) &= \forall x.(\pi(C,x) \rightarrow \pi(D,x)) \\
\pi(C \equiv D) &= \pi(C \sqsubseteq D) \land \pi(D \sqsubseteq C)
\end{aligned}
$$

RBox Axioms

$$
\begin{aligned}
\pi(R_1 \circ \ldots \circ R_n \sqsubseteq T) &= \forall x_1,\ldots,x_{n+1}.\left(\bigwedge_{i=1}^{n} \pi(R_i,x_i,x_{i+1}) \rightarrow \pi(T,x_1,x_{n+1})\right) \\
\pi(\mathsf{Tra}(R)) &= \pi(R \circ R \sqsubseteq R) \\
\pi(\mathsf{Irr}(S)) &= \forall x.\neg\pi(S,x,x) \\
\pi(\mathsf{Ref}(R)) &= \forall x.\pi(R,x,x) \\
\pi(\mathsf{Sym}(R)) &= \forall x,y.(\pi(R,x,y) \rightarrow \pi(R,y,x)) \\
\pi(\mathsf{Asy}(R)) &= \forall x,y.(\pi(R,x,y) \rightarrow \neg\pi(R,y,x)) \\
\pi(\mathsf{Dis}(S,T)) &= \forall x,y.\neg\pi(S,x,y) \lor \neg\pi(T,x,y)
\end{aligned}
$$

Figure 3.4: Transforming $\mathcal{SROIQ}^{\text{free}}$ axioms to first-order logic with equality

way in $\mathcal{SROIQ}^{\text{free}}$ and $\mathbf{FOL}_{\approx}$. Role and concept expressions thus correspond to first-order formulae with free variables, which is why the transformation function π takes additional parameters for representing the respective arguments in these cases.

The following result explains in which sense the direct DL semantics and the first-order translation agree.

Proposition 3.2.1 *Given $\mathcal{SROIQ}^{\text{free}}$ knowledge bases* KB *and* KB′*, we find that* KB $\models$ KB′ *iff* $\pi(\text{KB}) \models \pi(\text{KB}')$.

Proof. Similar results have been shown in various works, see e.g. [Mot06]. The proof is established by showing that every model of $\pi(\text{KB})$ can be considered as a model of KB and vice versa. □

The transformation π thus allows us to study the semantic interaction between DL and other fragments of first-order logic with equality. In the following, we will often apply notions of first-order logic such as the ones that were introduced in Section 2.2 to description logic knowledge bases or axioms. Whenever this is done, we tacitly assume that the according DL axioms have been replaced by their first-order translation as given by π above.

Due to the limited interaction of quantifiers in DL, it is often possible to represent axioms in terms of $\mathbf{FOL}_{\approx}$ formulae with at most two variables, thus establishing a correspondence of certain description logics and the two-variable fragment of first-order logic. This is not possible for $\mathcal{SROIQ}^{\text{free}}$, since number restrictions and role inclusion axioms require a higher number of variables. For the former case, this problem can be solved by introducing *counting quantifiers* into first-order logic. For example, an expression of the form $\exists_{\geq 3}x.\varphi$ states that there are at least three distinct values for x such that φ is satisfied. The two-variable fragment of first-order logic with counting quantifiers, usually denoted as C^2, is an important framework for studying the complexity of DLs (see, e.g., Section 5.2 or 9.3).

Finally, another general framework for studying DLs are so-called *Guarded Fragments* (GFs) as introduced in [AvBN98]. These fragments provide a generalisation of various modal, hybrid, and description logics based on the observation that typical uses of quantifiers in these formalisms involve "guard" formulae that restrict their applicability [Grä98]. For example, a typical universal role restriction $\forall x.R(a,x) \to C(x)$ uses $R(a,x)$ as a guard. However, GFs require all variables that occur in the filler ($C(x)$ in our example) to occur within a *single* atom in the guard, so that complex RIAs are not guarded. Various generalisations of GFs have been introduced in the literature, but we will not go into further details here.

3.3 Description Logic Nomenclature

$\mathcal{SROIQ}$ constitutes one of the most expressive description logics that have been studied, and it certainly is the most comprehensive DL for which practical implementations are available today. Many other DLs have been proposed and studied in the past two decades of research, and a great number of them can be considered as fragments of $\mathcal{SROIQ}$ or $\mathcal{SROIQ}^{\text{free}}$. A common goal when restricting to simpler logics is to reduce the complexity of inference problems, and to allow for different algorithmic approaches that are found to yield practical advantages. Moreover, many basic research questions are significantly harder to answer when considering $\mathcal{SROIQ}$ as a whole, as exemplified by the discussion in Chapter 7.

Fragments of description logics can be defined in various ways. The most obvious method is to restrict the available set of role and concept constructors, and axiom types. This will be the approach that is elaborated in most detail below. Another possibility is to restrict the usage of available constructors within axioms, e.g. by allowing unions of concepts only on the right-hand side of GCIs as exemplified by the Horn DLs that are studied in Chapter 6. Finally, it is possible to restrict to a certain set of description logic knowledge bases by imposing structural restrictions that refer to sets of axioms. Typical examples are the regularity restrictions that distinguish $\mathcal{SROIQ}$ from $\mathcal{SROIQ}^{\text{free}}$, or the restriction to *acyclic* TBoxes that has sometimes been considered in DL research [BCM+07].

The name that is given to a description logic typically reflects the logical operators that it supports, while further restrictions on the use of these concepts or on the overall structure of knowledge bases may not be mentioned explicitly. Since many combinations of operators are independent from one another, a general nomenclature has emerged for labelling DLs. Typically, each of the calligraphic letters of a DL's name represents a particular feature, while the order of letters is mostly governed by notational conventions. For example, the $\mathcal{I}$ in $\mathcal{SROIQ}$ indicates the availability of inverse roles, while $\mathcal{Q}$ indicates the availability of qualified number restrictions.

In addition, there are a number of DLs whose names have been coined historically without refering to a general naming scheme. Examples that are relevant in the context of this work include $\mathcal{ALC}$ and $\mathcal{EL}$. $\mathcal{ALC}$ is the *attribute language with complement*[1] that supports all Boolean operators on concepts ($\sqcap$, $\sqcup$, $\neg$) as well as universal and existential role restrictions. $\top$ and $\bot$ can be expressed indirectly but are typically included explicitly. The logic $\mathcal{EL}$ is the fragment of $\mathcal{ALC}$ that supports only concept conjunction and existential role restrictions.

Figure 3.5 gives an overview of common notations for specific features. Together with the above definitions, this suffices to understand the names of many common DLs. For example, $\mathcal{ALCHO}$ is the extension of $\mathcal{ALC}$ with role hierarchies and nominals, while $\mathcal{ALCHOIQ}$ further extends $\mathcal{ALCHO}$ with inverse roles and qualified number restrictions. Unfortunately, not all letters that occur in DL names have a clearly defined meaning, and features that are very common in a given line of research may not appear explicitly in the name of a DL at all. For example, there is no letter for indicating concept intersection. In some cases,

[1]The original source of this name attributes C to "complement" [SSS91] but "complex negation" might be more accurate given that $\mathcal{AL}$ already supports atomic complements.

Symbol	Expressive Feature	Example
$\mathcal{I}$	inverse roles	R^-
$\mathcal{O}$	nominals	$\{a\}$
$\mathcal{U}$	concept union	$C \sqcup D$
$\mathcal{Q}$	qualified number restrictions	$\leqslant 3\, R.C$, $\geqslant 2\, S.D$
$\mathcal{N}$	unqualified number restrictions	$\leqslant 3\, R.\top$, $\geqslant 2\, S.\top$
$\mathcal{F}$	functionality restrictions	$\top \sqsubseteq\, \leqslant 1\, R.\top$ (sometimes "$\mathsf{Func}(R)$")
$\mathcal{H}$	role hierarchies	$R \sqsubseteq T$
$\mathcal{R}$	role inclusion axioms	$R \circ S \sqsubseteq T$

Figure 3.5: Nomenclature for important DL features

there have also been diverging definitions that use the same letters. Notably, the letter $\mathcal{R}$ has sometimes been used to denote conjunctions of roles as discussed in Chapter 5, where we use another notation for this feature. Moreover, the name $\mathcal{FL}$ occurs in a number of basic DLs such as $\mathcal{FL}_0$ which supports only concept conjunctions, universal role restrictions, $\top$ and $\bot$. The historic naming has been motivated since $\mathcal{FL}$ was introduced as a simple *frame language* [BL84], and thus $\mathcal{F}$ is not related to functionality whenever it is followed by $\mathcal{L}$.

Furthermore, the letter $\mathcal{S}$ deserves some special discussion. Originally, $\mathcal{S}$ was introduced as an abbreviation for the extension of $\mathcal{ALC}$ with transitivity characteristics for roles. This interpretation is applicable to the DL $\mathcal{SHOIQ}$ and sublogics thereof such as $\mathcal{SHIQ}$ or $\mathcal{SHOQ}$. Role characteristics like symmetry that can be directly expressed in terms of other constructs are typically not mentioned explicitly in this context.

When considering complex role inclusion axioms, however, the letter $\mathcal{S}$ has received a different interpretation. Namely, the DLs $\mathcal{RIQ}$ and $\mathcal{SRIQ}$ both include $\mathcal{ALC}$ and support transitive roles, where the latter already follows from the presence of $\mathcal{R}$. The difference between $\mathcal{RIQ}$ and $\mathcal{SRIQ}$ is the availability of various other features including local reflexivity (Self), disjointness of roles, and various other role characteristics such as irreflexivity of roles. In the context of $\mathcal{R}$, the letter $\mathcal{S}$ has thus been interpreted to refer to "some additional features." For highly expressive DLs like $\mathcal{SROIQ}$, both of the above readings of $\mathcal{S}$ are applicable. In all other cases of DLs with $\mathcal{SR}$, we will explicitly clarify which constructs are available.

Chapter 4

Combining Description Logics with Datalog

As discussed in Section 1.3, a large number of rule formalisms have been considered in knowledge representation and reasoning, and in computer science in general. In this chapter, we introduce the well-known rule language *datalog* which can be viewed as a fragment of first-order logic but also as a basic logic programming dialect. Our focus within this work will mostly be on the former aspect since it allows us to combine datalog with description logics within the semantic framework of first-order predicate logic. Indeed, since the semantics of description logics can be captured in terms of first-order logic, it is straightforward – semantically speaking – to extend this formalism with first-order rule languages. Yet, as we shall see in many places of this work, much care is needed to preserve favourable computational properties such as decidability or tractability in such an extension.

We begin this chapter by giving an extended introduction to the syntax and semantics of datalog in Section 4.1. Besides providing some basic intuition about datalog and our first-order perspective on this language, this section also discusses the use of equality in datalog which is relevant throughout this work (Section 4.1.3). Thereafter, in Section 4.2.1, we combine datalog with $\mathcal{SROIQ}$ to obtain the (semantic core of) the *Semantic Web Rule Language* (SWRL) that provides the basis for many investigations within this work. After identifying the basic shortcomings of SWRL, we provide an extended overview of the refined approaches for combining datalog with description logics in Section 4.3 and 4.4, where we also point out the relationships of these approaches to the contents of later chapters.

4.1 Datalog as a First-Order Rule Language

As explained in Section 1.3, a natural way to approach the notion of "rule" in classical logic is to consider implications, i.e. formulae that have an implication operator as their outermost connective. Moreover, it makes sense to require that all variables that appear in a rule are universally quantified, thus expressing the fact that the implication is applicable to *all* individuals that satisfy the premise. However, further restrictions are required to arrive at a meaningful notion of rule that does not encompass all first-order logic formulae.

In this section, we introduce a particularly restricted rule language known as *datalog*. In a nutshell, a datalog rule is a logical implication that may only contain conjunctions, constant symbols, and universally quantified variables, but no disjunctions, negations, existential quantifiers, or function symbols. We always consider datalog as a sub-language of first-order logic to which the classical semantics applies. Both syntax and semantics will be explained in more precise terms below in a fully self-contained way.

Before going into further details, it is worth mentioning that datalog has originally been developed for querying databases. Rules and queries indeed have much in common. For example, the following datalog rule can be interpreted as a means of *querying* a given database for all book authors:

$$\forall x \forall y.(\mathtt{Person}(x) \land \mathtt{authorOf}(x, y) \land \mathtt{Book}(y) \rightarrow \mathtt{Bookauthor}(x)).$$

In this case, one would assume information about Person, authorOf, and Book to be stored in a database, while Bookauthor is derived from this data as a "query result." It is always possible to regard single rules as descriptions of relevant "views" on the data, and much work on datalog is related to the use of rules in this sense.

When considering datalog as a rule language, however, we also want to allow rules to be applied *recursively*. This means that the result of a rule can again be used by other rules to derive further conclusions, continuing until no further conclusions can be obtained from any rule. This use of recursion has been an important topic in the area of *deductive databases* as well, and semantic technologies can build on the results that were obtained in this field. A notable difference to our treatment is that many database-related applications define datalog based on a logic programming semantics or with certain "closure axioms." This is useful for achieving a *closed-world* semantics that is desirable for a database: if a fact is not in the database, it should be concluded that it is false. Such *non-monotonic* behaviour, however, is only obtained when extending datalog with further features, especially with non-monotonic negation. We do not consider any form of non-monotonicity in this chapter. For plain datalog, our definitions lead to exactly the same deductions as the closed-world approach. See [AHV94, Chapter 12] for a discussion and comparison of both approaches.

Another characteristic that is often considered in work on deductive databases is *safety* of rules which requires that all variables in the head of a rule occur also in its body. In the cases that we consider, such a restriction is not required, but we will encounter related notions when studying DL-safety in Chapter 9.

4.1.1 Syntax of Datalog

The following definition introduces central notions regarding the syntax of datalog. An example datalog program that illustrates this definition is given in Fig. 4.1.

Definition 4.1.1 A *signature* $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$ for datalog consists of a finite set of *individual names* (or *constant symbols*) $\mathbf{I}$, a finite set of *predicate names* (or *predicate symbols*) $\mathbf{P}$, and a finite set of *variable names* $\mathbf{V}$, all of which are mutually disjoint. The function $\mathsf{ar} : \mathbf{P} \to \mathbb{N}$ associates a natural number $\mathsf{ar}(P)$ with each predicate $P \in \mathbf{P}$ that defines the (unique) *arity* of P.

Based on a datalog signature $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$, we define the following notions:

– A datalog *term* is an element $t \in \mathbf{I} \cup \mathbf{V}$, i.e. an individual or variable name.

– A datalog *atom* is a formula of the form $P(t_1, \dots, t_n)$ given that $t_1, \dots, t_n$ are datalog terms, and $P \in \mathbf{P}$ is a predicate name of arity n, i.e. $\mathsf{ar}(P) = n$.

– A *datalog rule* is a formula of the form

$$\forall x_1 \dots \forall x_m.(B_1 \wedge \dots \wedge B_k \to H),$$

where $B_1, \dots, B_k$ are datalog atoms or $\top$, H is a datalog atom or $\bot$, and the variables $x_1, \dots, x_m$ are exactly those variables that occur within these atoms. A rule with $k = 1$ and $B_1 = \top$ is called a *fact*, and a rule with $H = \bot$ is called a *constraint*.

The premise of a datalog rule is called the *rule body* while the conclusion is called the *rule head*. A set of datalog rules is called a *datalog program* which hints at the relationship to logic programming. ◇

Since all variables in datalog are always universally quantified at the level of rules, it is common to omit the $\forall$ quantifiers from datalog rules. Moreover, $\bot$ in rule heads is sometimes not written explicitly, and facts $\top \to H$ are typically simply given as H. We adopt these simplifications whenever there is no danger of additional confusion. When clear from the context, we also omit the prefix “datalog” and simply speak of “terms,” “atoms,” “rules” etc. Finally, we will also typically leave the specific signature implicit in our considerations: details of the

$$\begin{array}{lr@{\;}l}
(1) & \mathtt{Vegetarian}(x) \land \mathtt{FishProduct}(y) \rightarrow & \mathtt{dislikes}(x, y) \\
(2) & \mathtt{orderedDish}(x, y) \land \mathtt{dislikes}(x, y) \rightarrow & \mathtt{Unhappy}(x) \\
(3) & \mathtt{orderedDish}(x, y) \rightarrow & \mathtt{Dish}(y) \\
(4) & \mathtt{dislikes}(x, z) \land \mathtt{Dish}(y) \land \mathtt{contains}(y, z) \rightarrow & \mathtt{dislikes}(x, y) \\
(5) & \top \rightarrow & \mathtt{Vegetarian}(\mathtt{markus}) \\
(6) & \mathtt{Happy}(x) \land \mathtt{Unhappy}(x) \rightarrow & \bot
\end{array}$$

Figure 4.1: Example datalog program

signature are usually inessential as long as it provides all syntactic symbols of a given datalog program (with the correct arity).

Figure 4.1 gives an example of a datalog program based on a datalog signature with set of constant symbols $\mathbf{I} = \{\mathtt{markus}\}$ and set of predicate symbols $\mathbf{P} = \{\mathtt{Dish}, \mathtt{Vegetarian}, \mathtt{FishProduct}, \mathtt{Happy}, \mathtt{Unhappy}, \mathtt{dislikes}, \mathtt{orderedDish}\}$. It is not hard to read the intended meaning from this set of datalog rules:

(1) "Every vegetarian dislikes all fish products."[1]

(2) "Anyone who ordered a dish that he or she dislikes is unhappy." This rule shows that not all variables occurring in a rule body need to appear in the rule head.

(3) "Everything that can be ordered as a dish actually is a dish."

(4) "If someone dislikes something that is contained in a certain dish, then this person will also dislike the whole dish."

(5) "Markus is a vegetarian."

(6) "Nobody can be happy and unhappy at the same time."

Note that some of the rules might be more widely applicable than desired. For example, rule (2) does not require that it was a person who ordered the dish. In practice, one might add further preconditions to ensure that such implicit assumptions do really hold. For our purposes, however, a simpler formalisation is preferred over a more correct one.

This example also illustrates that rules can often be read and understood rather easily, which is one reason why they might sometimes be preferred over other types of ontological axioms. Yet, some care is needed when dealing with rules: while the intention of a single rule can seem obvious, there are still many possibly unexpected conclusions that can be drawn from a set of rules. In particular, one must be aware that rules in first-order logic "work in both directions": if a rule

[1]Some "pesco-vegetarians" might disagree. We follow the historic definition of the *Vegetarian Society* here.

body is true then the rule head must of course also be true, but conversely, if a rule head is false, then the rule body must also be false. In other words, every rule is equivalent to its *contrapositive*: $p \rightarrow q$ is equivalent to $\neg q \rightarrow \neg p$. This may seem trivial, but it is often not relevant in pure logic programming settings due to the absence of explicit (classical) negation, and hence it may easily be overlooked in practice. Assume, e.g., that the following facts are added to the program of Fig. 4.1 (we assume that the new constant symbols have been added to the signature):

```
                 Happy(markus)
       orderedDish(markus,crêpeSuzette)
      FishProduct(worcestershireSauce)
```

With these additional assertions, we can (rightly) conclude that Crêpe Suzette does *not* contain Worcestershire Sauce: Since Markus is happy, he cannot be unhappy (6), and hence he did not order any dish he dislikes (2). Thus, since he ordered Crêpe Suzette, Markus does not dislike this dish. On the other hand, as a vegetarian (5) Markus dislikes Worcestershire Sauce on account of being a fish product (1). Thus, since Crêpe Suzette is a dish (3), and since Markus does not dislike it, rule (4) ensures us that the crêpe does not contain any Worcestershire Sauce.

The proper formal basis for such derivations is provided by the logical semantics of datalog as specified in the next section.

4.1.2 Semantics of Datalog

As mentioned in the previous section, we consider datalog as a sub-language of first-order logic, and its formal semantics is already determined by this fact. In this section, we give a self-contained presentation of the datalog semantics which can be slightly simplified due to the fact that function symbols and various first-order logical operators do not need to be addressed. As usual for first-order logic, the semantics of datalog is *model-theoretic*, i.e. it is based on defining which "models" a datalog program has. A correct conclusion from a datalog program then is any formula that is satisfied by all models of this program. As usual, a model is a special kind of *interpretation*, one that makes a given datalog program true. Hence we first explain what a datalog interpretation is and what it means for it to satisfy some datalog rule.

Definition 4.1.2 A *datalog interpretation* $\mathcal{I}$ is a tuple $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, consisting of a non-empty *interpretation domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$. The domain is a set of *individuals* that defines the (abstract) world within which all symbols are interpreted. The interpretation function establishes the mapping from symbols into this domain:

- If $a \in \mathbf{I}$ is an individual name, then $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, i.e. a is interpreted as an element of the domain.
- If $P \in \mathbf{P}$ is a predicate symbol of arity $\mathsf{ar}(P) = n$, then $P^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$, i.e. P is interpreted as an n-ary relation over the domain.

A *variable assignment* $\mathcal{Z}$ for $\mathcal{I}$ is a mapping $\mathcal{Z} : \mathbf{V} \to \Delta^{\mathcal{I}}$. For a term $t \in \mathbf{I} \cup \mathbf{V}$ we write $t^{\mathcal{I},\mathcal{Z}}$ to mean $t^{\mathcal{I}}$ if $t \in \mathbf{I}$, and $\mathcal{Z}(t)$ if $t \in \mathbf{V}$. Given an interpretation $\mathcal{I}$ and a variable assignment $\mathcal{Z}$ for $\mathcal{I}$, the *truth value* of a datalog formula is defined as follows:

- We set $\top^{\mathcal{I},\mathcal{Z}} := \textit{true}$ and $\bot^{\mathcal{I},\mathcal{Z}} := \textit{false}$.
- For a datalog atom $P(t_1, \ldots, t_n)$, we set $P(t_1, \ldots, t_n)^{\mathcal{I},\mathcal{Z}} := \textit{true}$ if we find that $\langle t_1^{\mathcal{I},\mathcal{Z}}, \ldots, t_n^{\mathcal{I},\mathcal{Z}} \rangle \in P^{\mathcal{I}}$, and $P(t_1, \ldots, t_n)^{\mathcal{I},\mathcal{Z}} := \textit{false}$ otherwise.
- For a conjunction $B_1 \wedge \ldots \wedge B_n$ of datalog atoms $B_1, \ldots, B_n$, we set $(B_1 \wedge \ldots \wedge B_n)^{\mathcal{I},\mathcal{Z}} := \textit{true}$ if $B_i^{\mathcal{I},\mathcal{Z}} = \textit{true}$ for all $i = 1, \ldots, n$. We set $(B_1 \wedge \ldots \wedge B_n)^{\mathcal{I},\mathcal{Z}} :=$ *false* otherwise.
- For a datalog rule $B \to H$, with B an arbitrary conjunction of datalog atoms, we set $(B \to H)^{\mathcal{I}} := \textit{true}$ if, for all variable assignments $\mathcal{Z}$ for $\mathcal{I}$, we find that either $B^{\mathcal{I},\mathcal{Z}} = \textit{false}$ or $H^{\mathcal{I},\mathcal{Z}} = \textit{true}$. We set $(B \to H)^{\mathcal{I}} := \textit{false}$ otherwise. ◇

Note that the truth of a rule does not depend on a particular variable assignment, since the (implicit) universal quantifiers bind all variables in all rules. The above definition gives rise to the usual notion of model-theoretic satisfiability and consequence:

Definition 4.1.3 An interpretation $\mathcal{I}$ *satisfies* a datalog rule $B \to H$ if $(B \to H)^{\mathcal{I}} = \textit{true}$, and it satisfies a datalog program if it satisfies all rules of the program. A rule (program) that is satisfied by some interpretation is called *satisfiable* or *consistent*, and each satisfying interpretation is called a *model* for the rule (program). A rule is a *conclusion* (or *consequence*) of a program if the rule is satisfied by all models of the program. ◇

Observe that the definition of semantic consequence includes all types of rules, so in particular it defines in which cases a certain fact is entailed by a datalog program. The entailment of facts is by far the most common reasoning problem for datalog, and many implementations are specifically tailored toward the derivation of facts.

The above finishes the formal definition of the datalog semantics. To illustrate the definitions, we describe a particularly interesting model for the example in Fig. 4.1 and the related facts on page 44. As a domain of interpretation, we pick the set of constant symbols of the given signature, i.e. $\Delta^{\mathcal{I}} = \{$`markus`, `crêpeSuzette`,

$$\begin{aligned}
\texttt{Vegetarian}^{\mathcal{I}} &= \texttt{Happy}^{\mathcal{I}} = \{\texttt{markus}\}\\
\texttt{FishProduct}^{\mathcal{I}} &= \{\texttt{worcestershireSauce}\}\\
\texttt{dislikes}^{\mathcal{I}} &= \{\langle\texttt{markus}, \texttt{worcestershireSauce}\rangle\}\\
\texttt{orderedDish}^{\mathcal{I}} &= \{\langle\texttt{markus}, \texttt{crêpeSuzette}\rangle\}\\
\texttt{Dish}^{\mathcal{I}} &= \{\texttt{crêpeSuzette}\}\\
\texttt{Unhappy}^{\mathcal{I}} &= \texttt{contains}^{\mathcal{I}} = \emptyset
\end{aligned}$$

Figure 4.2: Example datalog interpretation of predicate symbols

worcestershireSauce}. The mapping $\cdot^{\mathcal{I}}$ on constant symbols is defined to be the identity function, i.e. every constant symbol is mapped to itself. The interpretation of the predicate symbols is given in Fig. 4.2. It is straightforward to check that this interpretation is indeed a model for the given datalog program. For plain datalog programs that are consistent, it is always possible to construct models in this particularly simple fashion by just taking the set of constant symbols as interpretation domain, and such models are known as *Herbrand models*.[2] Moreover, it is always possible to find a model that satisfies as few datalog atoms as possible, such that no other model satisfies less datalog facts. The existence of such *least Herbrand models* is of great significance and can be exploited for practical implementations. Unfortunately, this nice property is lost as soon as we introduce description logics into the picture.

Even the availability of least Herbrand models does not make inferencing an easy task, computationally speaking. The following summarises some well-known complexity results for reasoning in datalog.

Fact 4.1.4 *Checking satisfiability of arbitrary datalog programs P is* ExpTime-*complete w.r.t. the size of the program, and* P-*complete w.r.t. the number of facts if the non-fact rules are assumed to be fixed (data complexity). Checking satisfiability in the class of datalog programs with at most k variables per rule can be done in polynomial time w.r.t.* $\#(\mathbf{I})^k$, *and is thus* P-*complete w.r.t. the size of the program.*

The same results hold for the problem of checking the entailment of ground facts from datalog programs.

See [DEGV01] for details and proofs. Upper bounds in all cases can be obtained by reducing datalog programs to propositional Horn logic programs by *grounding*, i.e. by uniformly replacing the variables of each rule by constant symbols in all possible ways. Grounding is obviously exponential in the number of variables per rule, and polynomial in the number of constants.

[2]After the French mathematician *Jacques Herbrand*, i.e. pronounced /*ɛrbrɑ̃*/ with *H* silent

4.1.3 Equality

An important aspect of first-order logic and thus of description logics is the fact that different constant symbols may refer to the same semantic individual. In contrast, logic programming often adopts the *Unique Name Assumption* (UNA) that requires differently named individuals to be distinct. This is also reflected in the notion of Herbrand models mentioned above which use the set of all constant symbols as their domain, such that distinct constants represent distinct domain elements. In this section, we discuss how explicit equality can be introduced into datalog so as to align it with description logics.

First it must be noted that the semantics of datalog as introduced above does not make the UNA, so it is indeed possible that distinct constant symbols are interpreted as the same domain element. This does hardly play a rôle for the conclusions that can be drawn from a datalog program, since datalog provides no way of stating or checking the equality of two individuals. This is why the least Herbrand model can still be used for reasoning: models with less domain elements may exist, but the least Herbrand model is guaranteed to entail the least amount of positive information.[3] Datalog does, however, provide a way of asserting the inequality of two individuals by requiring them to have incompatible properties. For example, the program $\{P(a), Q(b), P(x) \wedge Q(x) \to \bot\}$ implies that a and b are interpreted differently in all models.

A straightforward solution for allowing explicit positive equality assertions in datalog is to simply consider datalog as a fragment of first-order logic with equality $\mathbf{FOL}_{\approx}$, and to allow the equality symbol $\approx$ to be used like a binary predicate. Besides this extension, the above definitions of syntax and semantics of datalog carry over to this new setting. However, datalog programs with equality may no longer have (least) Herbrand models. For example, the program $\{x \approx a\}$ requires all domain elements to be equal to the interpretation of the constant a, and hence admits only models with singleton domains. We will see below how the underlying ideas of Herbrand models can still be recovered within this setting.

Another practical problem that we encounter when adding equality to datalog is the lack of inference engines. While datalog can be processed by almost any logic programming tool, including a number of dedicated datalog reasoners, there are hardly any implementations of datalog with equality. Fortunately, it is well-known that equality can be axiomatised using rules of datalog without equality; see, e.g., [Fit96]. More formally, given a datalog program P over a signature $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$, the datalog program $P_{\approx}$ is defined to consist of the following rules:

[3] Indeed, "least" refers to the amount of positive information as compared to other Herbrand models, not to the size of the interpretation domain.

$$\begin{aligned}
&& a \approx a && \text{for all } a \in \mathbf{I}\\
x \approx y &\to& y \approx x &&\\
x \approx y \wedge y \approx z &\to& x \approx z &&\\
P(x_1, \ldots, x_i, \ldots, x_n) \wedge x_i \approx y_i &\to& P(x_1, \ldots, y_i, \ldots, x_n) && \text{for all } P \in \mathbf{P}
\end{aligned}$$

It is well-known that, for all ground atoms φ (possibly including the symbol $\approx$) over the signature $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$, we have that $P \models \varphi$ in $\mathbf{FOL}_{\approx}$ iff $P \cup P_{\approx} \models \varphi$ holds in first-order logic without equality when considering $\approx$ as a new binary predicate [Fit96]. Indeed, it is easy to see that every **FOL** model of $P \cup P_{\approx}$ has a corresponding $\mathbf{FOL}_{\approx}$ model of P that is obtained by factorising the interpretation domain based on the equivalence relation induced by $\approx$, and, conversely, every $\mathbf{FOL}_{\approx}$ model of P leads to a **FOL** model of $P \cup P_{\approx}$ that interprets $\approx$ as identity relation on the domain. In this sense, the relation of P and $P \cup P_{\approx}$ is close to our notion of emulation, although the use of different underlying logics is not encompassed there. Together with the observation that the size of $P_{\approx}$ is linearly bounded in the size of the underlying signature, we can conclude that the worst-case complexity of reasoning in datalog is not increased when adding equality.

This provides us with an alternative perspective on equality as an auxiliary predicate with a fixed axiomatisation. This view is useful since it allows us to understand the notion that corresponds to least (Herbrand) models in datalog with equality. Namely, the "least models" of a datalog program with equality can be considered to be the models that are obtained by constructing $\mathbf{FOL}_{\approx}$ models from least **FOL** models of the datalog program with axiomatised equality predicate. For example, the least Herbrand model leads to a canonical model in $\mathbf{FOL}_{\approx}$ which is obtained as a factorisation of the Herbrand model. We will use this correspondence in some arguments, and in particular there are cases where it is more convenient to construct a model by explicitly specifying the extension of $\approx$ instead of dealing with equivalence classes that are obtained by an according factorisation.

Since both views are essentially equivalent, we freely change between the **FOL** and $\mathbf{FOL}_{\approx}$ perspective when considering datalog, and in particular we typically use $\approx$ in datalog without specifying which logical framework is to be used. While the extension of datalog to a fragment of $\mathbf{FOL}_{\approx}$ in this sense does not increase its modelling power, we point out that an axiomatisation as above is often not satisfactory for devising efficient implementations. The reason is that a general-purpose equality theory creates a large number of possible inference patterns that may thwart some common optimisations and that may even lead to non-termination in some logic programming systems. A common inference method for datalog and logic programming is resolution, and a number of additional techniques are known to reduce the amount of unnecessary inferences in resolution-based calculi [BG98]. In many cases, it is also possible to use a more restricted equality theory without losing consequences.

4.2 Datalog ∪ Description Logics: SWRL

In this section, we consider the approach of combining all expressive features of datalog with all expressive features of DL in a rather straightforward way that was first proposed in [HPS04]. The approach has become popular through the *Semantic Web Rule Language* (SWRL, pronounced "swirl") that was published as a W3C member submission [HPSB+04]. The original proposal of SWRL also includes a number of built-in functions for handling datatype values, and it uses an XML-based syntax that is not discussed here. Yet, it is still closely related to the formalism that is discussed in this section, and we will use the name "SWRL" throughout to refer to this language.

4.2.1 Defining SWRL

Even though the paradigm of rule-based modelling is quite different from the ontological modelling in description logic, it is not hard to see that a combination of datalog and DL is indeed meaningful. Both languages can be viewed as sublanguages of $\mathbf{FOL}_{\approx}$, so the combination of a datalog program with a DL ontology can always be viewed as a collection of first-order logic formulae with the usual first-order semantics. So, at least conceptually, there are no major problems. For clarity, we explicitly define the syntax and semantics of SWRL below.

Definition 4.2.1 A *signature* of SWRL is a signature of datalog $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$ as in Definition 4.1.1 with designated disjoint subsets of concept names $\mathbf{A} \subseteq \mathbf{P}$, simple role names $\mathbf{N}_s \subseteq \mathbf{P}$, and non-simple role names $\mathbf{N}_n \subseteq \mathbf{P}$, such that

– $A \in \mathbf{A}$ implies $\mathsf{ar}(A) = 1$, and

– $R \in \mathbf{N}_s \cup \mathbf{N}_n$ implies $\mathsf{ar}(R) = 2$.

In particular, $\langle \mathbf{I}, \mathbf{A}, \mathbf{N} \rangle$ with $\mathbf{N} = \mathbf{N}_s \cup \mathbf{N}_n$ is a DL signature.

A *SWRL atom* is a datalog atom over $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$, or an expression of the form $C(x)$ or $R(x, y)$ where $x, y \in \mathbf{V}$, and $C \in \mathbf{C}$ and $R \in \mathbf{R}$ are $\mathcal{SROIQ}^{\text{free}}$ concept and role expressions over the signature $\langle \mathbf{I}, \mathbf{A}, \mathbf{N} \rangle$. SWRL *rules* and *programs* are defined like datalog rules and programs but based on SWRL atoms. A SWRL program is also called a SWRL *rule base*.

Interpretations and *variable assignments* in SWRL are the same as for datalog, and satisfaction of datalog atoms is defined as before. An interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and a variable assignment $\mathcal{Z}$ for $\mathcal{I}$ *satisfy* a SWRL atom of the form $C(x)$ with $C \in \mathbf{C}$ if $\mathcal{Z}(x) \in C^{\mathcal{I}}$. Here, $C^{\mathcal{I}}$ denotes the (DL) extension of C under $\mathcal{I}$ as specified in Definition 3.1.5. Similarly, $\mathcal{I}$ and $\mathcal{Z}$ *satisfy* a SWRL atom of the form $R(x, y)$ with $R \in \mathbf{R}$ if $\langle \mathcal{Z}(x), \mathcal{Z}(y) \rangle \in R^{\mathcal{I}}$. Satisfaction for rules and programs is

(7) $\exists\texttt{orderedDish}.\texttt{ThaiCurry}(\texttt{markus})$
(8) $\texttt{ThaiCurry} \sqsubseteq \exists\texttt{contains}.\texttt{FishProduct}$

Figure 4.3: Description logic axioms extending the datalog program from Fig. 4.1

defined as in Definition 4.1.2 using the extended satisfaction relation for SWRL atoms. Logical consequence is defined as in Definition 4.1.3. ◇

Since this definition explicitly allows for datalog predicates of arbitrary arity, SWRL in this sense clearly is a proper syntactic superset of datalog. This is not the case for $\mathcal{SROIQ}^{\text{free}}$, since we do not explicitly include concept subsumptions, role inclusion axioms, or other role assertions. However, all of these concepts can easily be expressed in SWRL:

- concept subsumptions $C \sqsubseteq D$ can be expressed by rules $C(x) \to D(x)$,
- generalised role inclusion axioms $R_1 \circ \ldots \circ R_n \sqsubseteq T$ can be expressed by rules $R_1(x_1, x_2) \wedge \ldots \wedge R_n(x_n, x_{n+1}) \to T(x_1, x_{n+1})$,
- role disjointness axioms $\mathsf{Dis}(R, T)$ can be expressed by rules (integrity constraints) $R(x, y) \wedge T(x, y) \to \bot$.

The remaining role characteristics symmetry, asymmetry, transitivity, reflexivity, and irreflexivity can be expressed in $\mathcal{SROIQ}$ as explained in Section 3.1.3, hence can be omitted without losing expressivity. Thus, when specifying SWRL rule bases,[4] we will freely use axioms in DL syntax to denote the corresponding SWRL rules, but we do not need to explicitly consider these cases in formal arguments.

Further note that Definition 4.2.1 does not require all unary and binary predicates to be interpreted as DL concept names and role names, respectively. Later, in Chapter 9, we will discuss cases where a distinction between DL atoms and non-DL atoms is indeed useful. For now, it will not be necessary to distinguish datalog and DL components in SWRL rule bases.

As an example for such a combined knowledge base, consider again the datalog rules from Fig. 4.1 together with the additional description logic axioms given in Fig. 4.3 (using DL syntax as explained above). By (7), Markus has ordered some Thai curry dish, and, according to this example, all Thai curries contain some fish product. Combining these statements with the rules of Fig. 4.1, we would intuitively expect the conclusion that Markus is now unhappy. Using the above semantics, we can support our intuition with a more formal argument.

[4]We prefer the term "rule base" since it emphasises the relationship to (DL) knowledge bases avoids the procedural connotation of the term "program" that is common for datalog theories.

Using the semantics of $\exists$ from Definition 3.1.5, we find that every interpretation $\mathcal{I}$ that satisfies (7) must have some element e in its domain $\Delta^{\mathcal{I}}$ such that $(\mathtt{markus}^{\mathcal{I}}, e) \in \mathtt{orderedDish}^{\mathcal{I}}$ and $e \in \mathtt{ThaiCurry}^{\mathcal{I}}$. But if $\mathcal{I}$ also satisfies rule (3), then we must have $e \in \mathtt{Dish}^{\mathcal{I}}$ as well. This last conclusion can be obtained as follows: Clearly there is a variable assignment $\mathcal{Z}$ with $\mathcal{Z}(x) = \mathtt{markus}^{\mathcal{I}}$ and $\mathcal{Z}(y) = e$. Since $\mathcal{Z}$ and $\mathcal{I}$ satisfy the body of rule (3), they must also satisfy its head. So we obtain $\mathcal{Z}(y) \in \mathtt{Dish}^{\mathcal{I}}$ as claimed. Normally, it is not required to explain conclusions from rules in that much detail, and one usually just says that $e \in \mathtt{Dish}^{\mathcal{I}}$ follows by *applying* rule (3).

Now if (8) is satisfied, then $e \in (\exists\mathtt{contains}.\mathtt{FishProduct})^{\mathcal{I}}$. Again this means that there must be some element $f \in \Delta^{\mathcal{I}}$ such that $(e, f) \in \mathtt{contains}^{\mathcal{I}}$ and $f \in \mathtt{FishProduct}^{\mathcal{I}}$. Applying rules (5) and (1), we also know that $(\mathtt{markus}^{\mathcal{I}}, f) \in \mathtt{dislikes}^{\mathcal{I}}$. Thus we can apply rule (4) with a variable assignment $\mathcal{Z}$ with $\mathcal{Z}(x) = \mathtt{markus}^{\mathcal{I}}$, $\mathcal{Z}(y) = e$, and $\mathcal{Z}(z) = f$ to conclude that $(\mathtt{markus}^{\mathcal{I}}, e) \in \mathtt{dislikes}^{\mathcal{I}}$. Thus, we have established that Markus dislikes the (unnamed) dish e which he ordered. Therefore rule (2) can be applied to conclude that $\mathtt{markus}^{\mathcal{I}} \in \mathtt{Unhappy}^{\mathcal{I}}$.

The above conclusions were drawn by assuming merely that $\mathcal{I}$ satisfies the rules and axioms (1)–(9), and they are thus valid for an arbitrary model of our combined knowledge base. In other words, every model of the above rules and axioms must also satisfy $\mathtt{Unhappy}(\mathtt{markus})$, which is therefore a logical conclusion of the knowledge base.

4.2.2 Reasoning in SWRL

The previous section showed how the formal semantics of datalog and description logics can be used to derive conclusions. The argument we gave there, however, was still somewhat informal and required some amount of thought on our side. It would clearly be desirable to automate this process, i.e. to develop software tools that automatically draw conclusions from SWRL rule bases. Unfortunately, it turns out that this is not possible: all standard reasoning problems for SWRL are undecidable, even if further restricting the underlying description logic. Before providing a simple proof for this result below, we discuss the practical consequences of this situation.

Undecidability of SWRL might be somewhat disappointing since it ensures us that it is impossible to ever devise a software tool that can compute *all* conclusions from *all* possible SWRL rule bases. But this formulation also already hints at two ways of escaping this problem. As a first option, one might be content with a tool that draws at least *some* conclusions which are for certain, i.e. an inferencing program that is sound but incomplete. Alternatively, one can try to find reasoning methods that are sound and complete, but that cannot be applied to all possible rule bases. As we will see in the following chapters, both approaches are often closely related: having identified a fragment of SWRL rules for which a sound and complete reasoning algorithm is available, one can devise ways for modifying arbitrary SWRL rule bases to make them processable by the same algorithm. Such modifications are typically *weakenings* – the modified rule base is a logical consequence of the original one – and hence the reasoning algorithm, while sound and complete for the modified rule base, is still sound but no longer complete for the original one. Two main approaches for obtaining fragments of SWRL for which reasoning is decidable are *Description Logic Rules* (Chapter 8) and *DL-safe Rules* (Chapter 9).

In the remainder of this section, we elaborate on the undecidability of SWRL. There are, in fact, many different ways of illustrating this undecidability, since many well-known undecidability proofs for description logics are easily adapted for SWRL rule bases. Examples include the undecidability proof for non-simple roles in number restrictions [HST99], non-regular role boxes [HS04], and even the basic undecidability result for KL-ONE – a predecessor of today's description logics – given in [SS89]. The argument given for SWRL's undecidability in [HPSBT05] is based on representing an infinite tiling (domino) problem in SWRL. Intuitively speaking, the undecidability of SWRL is a result of the interplay of two features: (1) SWRL does not have a finite model property, so that a rule base can require the existence of infinitely many domain elements; (2) SWRL can describe and entail complex, irregular relationships between domain elements. Item (1) is typical for DLs which can use existential quantifiers to derive new elements, but attention can typically be restricted to (finite or infinite) DL models which have a rather regular structure that allows reasoning algorithms to use blocking approaches to ensure termination. Conversely, in the case of datalog, irregular relationships between domain elements can be described but the supply of such elements is naturally limited by the amount of constant symbols that are used: satisfiable datalog programs always have a finite model – any Herbrand model. The following proof is given to further illustrate this intuition using the *Post Correspondence Problem* as a classical undecidable problem.

Fact 4.2.2 *Satisfiability of SWRL rule bases is undecidable.*

Proof. The undecidable *Post Correspondence Problem* (PCP) is described as follows: given two lists of words $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ over some alphabet Σ, is there a sequence of numbers $i_1, \ldots, i_k$ $(1 \leq i_j \leq n)$ such that $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$?

To reduce this problem to SWRL satisfiability, words are represented by chains of binary relations. Thus assume there are distinct role names $\{U_j | 1 \leq j \leq n\} \cup \{V_j | 1 \leq j \leq n\} \cup \{R_\sigma \mid \sigma \in \Sigma\} \subseteq \mathbf{N}$. For each word $u_j = \sigma_{j1} \ldots \sigma_{jm}$ and corresponding role U_j, add a rule $R_{\sigma_{j1}}(x_1, x_2) \wedge \ldots \wedge R_{\sigma_{jm}}(x_m, x_{m+1}) \to U_j(x_1, x_{m+1})$, and likewise for words v_j. Moreover, add facts of the form $\exists R_\sigma.\top(x)$ for each $\sigma \in \Sigma$.

This construction ensures that every model contains a role chain for all possible sequences of letters, and that roles U_j and V_j connect the first and last elements of each role chain that represents the word u_j and v_j, respectively.

We wish to ensure that the constructed rule base is unsatisfiable whenever the given instance of the PCP can be solved. For this it is not adequate to generally disallow that some sequences of words U_j and V_j connect the same individuals: a contradiction should only arise if the same indices $i_1, \dots, i_k$ have been used in the construction of both sequences. For this, we introduce another set of (distinct) role names $\{M_j | 1 \leq j \leq n\} \cup \{U, V\} \subseteq \mathbf{N}$ and facts $\exists M_j.\top(x)$ for each $j \in \{1, \dots, n\}$. The roles M_j are markers to record a sequence of words u_j and v_j. This is implemented by adding rules of the form $U_j(x, y) \wedge M_j(y, z) \to U(x, z)$ and $U_j(x, y) \wedge U(y, y') \wedge M_j(y', z) \to U(x, z)$ for each $j \in \{1, \dots, n\}$, and likewise for V. Finally, the rule $U(x, y) \wedge V(x, y) \to \bot$ constraints the models of the rule base as intended.

It is not hard to see that this rule base is unsatisfiable iff the initial instance of the PCP has a solution. Otherwise, we can find a canonical model $\mathcal{I}$ in which the graph structure established by the relationships $R_\sigma^{\mathcal{I}}$ and $M_\sigma^{\mathcal{I}}$ is tree-shaped, i.e. contains no cycles and no parallel edges. In particular, this means that no two distinct sequences of edges $M_{i1}^{\mathcal{I}} \dots M_{ik}^{\mathcal{I}}$ lead to the same domain element. The canonical model can further be assumed to contain exactly those relationships $U_{(j)}^{\mathcal{I}}$ and $V_{(j)}^{\mathcal{I}}$ that are required by the rules with non-empty heads (this can be ensured, in essence, since the underlying logic is monotonic). It is easy to see that the rule $U(x, y) \wedge V(x, y) \to \bot$ is also satisfied if a model is constructed like this: Suppose for a contradiction that it is not, i.e. that there is a tuple in $U^{\mathcal{I}} \cap V^{\mathcal{I}}$. An easy induction over the length of the entailment of $U^{\mathcal{I}}$ and $V^{\mathcal{I}}$ then shows that there is a corresponding sequence of words in the original PCP – a contradiction.

Conversely, it is easy to show that, whenever the PCP has a solution sequence, this sequence will lead to a contradiction in each interpretation of the rule base. This finishes the proof. □

We point out that the SWRL rules that are used in the above proof could also be expressed in $\mathcal{SROIQ}^{\text{free}}$ by using role chains and role disjointness axioms, where the resulting RBox would not satisfy the regularity constraints of $\mathcal{SROIQ}$. The proof thus could be adapted to show undecidability of reasoning in $\mathcal{SROIQ}^{\text{free}}$.

4.3 Approaches for Combining Rules and DLs

The Semantic Web Rule Language is a rather straightforward approach for combining rule languages and datalog, and its undecidability is clearly a disadvantage. A variety of further approaches have been considered for combining rules in the sense of first-order logic or logic programming with DLs. In this section, we provide a short survey of related works, and point out the approaches that we are going to pursue in the remainder of this work.

A first approach for reconciling DL and rules in a decidable fashion was introduced under the name *Description Logic Programming* (DLP) in [GHVD03, Vol04]. In contrast to SWRL, DLP strives to define a common "subset" of the DL $\mathcal{SHOIQ}$ and datalog, and thus it is strictly weaker than either formalism. In the absence of a common syntax, however, it is not obvious how to define the "intersection" of two logical formalisms, and a number of variants of DLP-like languages have been considered. The most recent incarnation of DLP can be recognised in the OWL 2 profile OWL RL, which can be viewed as an extended version of DLP for $\mathcal{SROIQ}$. In this work, we encounter similar DLs in Chapter 6 when considering Horn description logics, and in particular in Section 6.2. Moreover, we return to the question of how to define the "intersection" of DL and datalog in Chapter 7, where the logic $\mathcal{DLP}$ is introduced as a DLP-type sublanguage of $\mathcal{SROIQ}$ that is maximal in a concrete sense.

Partly inspired by DLP, Motik et al. proposed a reduction of $\mathcal{SHIQ}$ knowledge bases to disjunctive datalog programs[5] that could be used to compute ground entailments from datalog, and that has been implemented in the KAON2 system [MS06, Mot06]. Using the terminology of Section 2.2, we can say that the disjunctive datalog program as computed by KAON2 $\mathbf{FOL}_{\approx}^{\text{ground}}$-*emulates* the original $\mathcal{SHIQ}$ knowledge base where $\mathbf{FOL}_{\approx}^{\text{ground}}$ is the variable-free fragment of first-order logic with equality. It turned out that some knowledge bases could be translated into non-disjunctive datalog, and that the low reasoning complexity w.r.t. the number of facts in a datalog program could thus be exploited in these cases. The corresponding fragment of $\mathcal{SHIQ}$ was called Horn-$\mathcal{SHIQ}$ [HMS05]. It is the basis for our investigation of a number of further Horn DLs in Chapter 6, where additional related work on these topics can be found.

Some other approaches have focussed on identifying decidable sub-languages of SWRL. The datalog reduction in KAON2 suggested the combination of $\mathcal{SHIQ}$ knowledge bases with additional datalog rules that would simply be added to the program that is obtained from a transformation. Doing this does, of course, not capture the semantics of arbitrary SWRL rule bases, but it suffices to derive all entailments that can be obtained when restricting the applicability of the added datalog rules to *named individuals*, i.e. to individuals that are referred to by a constant name of the original knowledge base. A class of SWRL rules for which this suffices to obtain a complete reasoning procedure has been introduced in [MSS05] under the name *DL-safe rules*. Related work is further discussed in Chapter 9, where we extend this approach by combining it with another family of decidable SWRL fragments known as *Description Logic Rules*.

[5]Disjunctive datalog is the extensions of datalog that allows disjunctions in rule heads.

Description Logic Rules have first been introduced in [KRH08a] and, independently, in [GSH08]. The key idea of this approach is to identify fragments of SWRL that can be expressed (or rather emulated) by DL knowledge bases, but which may require some non-trivial encodings for this purpose. In this sense, DL Rules do not actually increase the expressiveness of a description logic, but rather provide a rule-based perspective that might be more adequate for some applications. DL Rules are introduced in detail in Chapter 8, and they provide an important basis for *DL+safe* rules, a generalisation of DL-safe rules that is presented in Chapter 9. The latter chapter also introduces the new light-weight DL-based rule language ELP [KRH08b].

Other prominent approaches for a first-order integration of DL and datalog have been CARIN [LR98] and (first-order) $\mathcal{DL}$*+log* [Ros06], both of which consider certain forms of DL-safety conditions to ensure decidability. See Section 9.6 for a more detailed discussion on how these relate to DL-safe rules and to our extension thereof. In short, it turns out that our DL+safe rules generalise both DL-safe rules and recursive role-safe CARIN, while $\mathcal{DL}$*+log* remains incomparable.

In addition to these foundational works, the interoperability of DL and rules also imposes practical challenges. Indeed, the major contribution of the SWRL member submission [HPSB$^+$04] was not to invent a novel knowledge representation paradigm, but to specify a structural and syntactic framework for using SWRL rule base in actual systems. Although not a formal standard, SWRL continues to be the main rule syntax that is used in current implementations. More recently, extensions have been proposed to improve the compatibility of SWRL with various syntactic forms of OWL, and to incorporate an explicit notion of DL-safety into SWRL rule bases [GHPPS09]. Meanwhile, the W3C initiated an effort for standardising a so-called *Rule Interchange Format* (RIF) as a unified framework for increasing the interoperability of rule-based systems. However, this effort was not scoped to any particular notion of "rule" and, in consequence, lead to a number of different rule languages that are only weakly compatible. A language that resembles datalog (though extended with frame-like syntax and some other features) has been developed under the name *RIF Core* [BHK$^+$09], and it can further be extended to the *RIF Basic Logic Dialect* that encompasses first-order Horn logic. Both languages use a first-order semantics that is compatible with the rule languages studied within this work. Moreover, RIF includes a document on RDF and OWL compatibility [dB09] that suggests a combined semantics of OWL and RIF that is closely related to SWRL, and that provides a basis for defining DL-safety in RIF rule bases. However, it remains to be seen to which extent RIF will be adopted in practical applications.

Besides the close integration of rules and DLs in the framework of SWRL, there are further approaches of combining some forms of rule languages with description logics. Many of those approaches consider rules in the sense of logic programming (see [Llo88] for a textbook introduction), focusing specifically on the non-monotonic inferencing features of the latter. An important kind of works are so-called *hybrid* approaches that use DL knowledge bases as source of background knowledge that is used by some form of rule base or logic program. A classical approach of this type is $\mathcal{AL}$-log which semantically resembles SWRL but does not allow DL atoms in rule heads [DLNS98]. While this can still be viewed as a restricted form of DL-safe SWRL rules, Eiter et al. introduced *dl-programs* as a hybrid formalism for combining *answer set programming* – a paradigm of non-monotonic logic programming – with description logics [ELST04]. This approach later has been further generalised to so-called HEX-programs that allow for more general combinations of *h*igher-order logic programs with *ex*ternally defined atoms [EIST05].

Further approaches provide an even tighter integration of description logics and non-monotonic logic programming paradigms. An example is given by $\mathcal{DL}$*+log* that admits a non-monotonic evaluation based on the *Gelfond-Lifschitz-reduct* of a disjunctive datalog program for some interpretation, while still allowing DL axioms to be interpreted under an open world semantics [Ros06]. The idea of integrating closed and open world reasoning by combining DLs and (disjunctive) datalog was further advanced by the introduction of *hybrid MKNF knowledge bases* [MHRS06, MR07] which introduce the paradigm of *Minimal Knowledge and Negation as Failure* into DLs. An alternative approach of interpreting such hybrid knowledge bases under the *well-founded semantics* has been presented in [KAH08].

4.4 Rules and Conjunctive Queries

Another area that is closely related to the combination of rules and DLs is the study of conjunctive queries for description logics. Conjunctive queries are a well-known formalism in the field of databases that can easily be adapted to DL knowledge bases. Using the terminology of this chapter, a *conjunctive query* (CQ) can be defined as a SWRL rule

- that contains only DL atoms in its body, and
- that has a non-DL atom of arbitrary arity as its head.

In this sense, CQs are a special kind of non-recursive rules. The predicate name of the head atom is generally used only in a single query, and hence is inessential. One is typically interested in the instances of the head predicate of the query that are entailed by a given DL knowledge base. Variables that occur in the head of a query are therefore called *distinguished variables*, and all other variables are called *non-distinguished*.

An *answer* to a query $B \to Q(x_1, \ldots, x_n)$ over a given DL knowledge base KB is a tuple $\langle a_1, \ldots, a_n \rangle$ of individual names $a_i \in \mathbf{I}$ such that $Q(a_1, \ldots, a_n)$ is a logical consequence of $\mathrm{KB} \cup \{B \to Q(x_1, \ldots, x_n)\}$. The *query entailment problem* is the problem of deciding whether or not a tuple is an answer to a CQ in this sense, while *query answering* is the task of finding all such tuples.

We thus find that the entailment of CQs is a special case of entailment checking for SWRL rule bases, and thus closely related to the subject of this work. This relation is well-known and has been exploited in some works. [Ros06] shows a close relationship between decidability of $\mathcal{DL}$+*log* and decidability of conjunctive query answering for the respective DL. [LR98] study non-recursive versions of CARIN that are closely related to the CQ entailment problem over the given DL. However, rule languages typically become significantly more complicated when allowing for recursion, thus requiring additional restrictions that are not relevant for CQs.

Conversely, research in CQs typically considers arbitrary forms of queries, and studies appropriate restrictions on the DL instead. CQ entailment in this general form is often a very hard problem indeed. If knowledge bases contain only ABox information, i.e. assertional data, query entailment corresponds to finding patterns in a labelled graph, and thus is already NP-complete. For the tractable description logic $\mathcal{EL}^{++}$, CQ entailment becomes PSpace-complete [KRH07b, KR07]. This result requires $\mathcal{EL}^{++}$ to be restricted to regular RBoxes since CQ entailment is undecidable otherwise.

In fact, CQ complexity often is significantly higher than the complexity of standard reasoning problems, although there are still a number of open questions especially regarding conjunctive queries for more expressive DLs. It is known that CQ entailment for $\mathcal{SHIQ}$ is 2ExpTime-complete [GLHS08, Lut08]. Omitting transitive and inverse roles simplifies the problem complexity to ExpTime [Lut08]. Allowing transitive roles leads to co-NExpTime-hardness, and even to 2ExpTime-hardness if role hierarchies are also allowed [ELOS09].

CQ entailment for $\mathcal{SHOQ}$ is known to be decidable in 2ExpTime but it remains open if this is optimal [GHS08]. As of today, it is unknown whether CQ entailment for $\mathcal{SHOIQ}$ – the DL that is closely related to the 2004 OWL standard – is decidable, but it has recently been shown that this is the case if the query contains simple roles only [GR09]. This result hinges on a decision procedure for CQ entailment in $\mathcal{ALCHOIQ}b$ ($\mathcal{ALCHOIQ}(b)$ in the notation of Chapter 5) from which no upper complexity bounds can be obtained.

Of the above approaches, [KRH07b] is the only one that considers (regular) RBoxes with general RIAs. Approaches to extend CQs toward expressive DLs with RIAs are based on extensions of CQs with regular expressions over roles, leading to so-called *regular path queries* [CEO07]. This line of research has recently lead to first decidability results for query entailment in DLs that extend $\mathcal{SR}$ [CEO09].

As of today, CQs and rules still constitute two separate fields of research, albeit with many touching points. The main thrust of CQ research is currently aimed at extending CQs to even more expressive DLs, leading to highly complex yet decidable rule languages in the spirit of $\mathcal{DL}$+*log* [Ros06]. The focus of this work, in contrast, is the development of DL rule languages that can directly be used even with highly expressive DLs while ensuring decidability by restricting the expressive power of the rule component.

Chapter 5

Extending Description Logics with Role Constructors

In rule languages like datalog, the uses of an atom in a rule does not depend on the arity of the atom's predicate. In description logics, in contrast, concept expressions are typically much richer than role expressions. For example, the rule $C(x) \wedge D(x) \rightarrow E(x)$ can be expressed by the concept inclusion axiom $C \sqcap D \sqsubseteq E$, whereas the rule $R(x,y) \wedge S(x,y) \rightarrow T(x,y)$ is not expressible in $\mathcal{SROIQ}$. This fundamental imbalance in expressive power restricts the interoperability of DLs and rules, and the objective of this chapter therefore is to investigate the use of more expressive role constructors in description logics.

In DL history, Boolean constructors (negation, conjunction, disjunction) on roles have occurred and have been investigated sporadically in many places, but have never been integrated into the mainstream of researched languages nor influenced standardisation efforts. *Concept products* of the form $C \times D$ have been suggested as a means of describing a role that relates all instances of C to all instances of D, leading to another role constructor that has not been adopted in many approaches. In this chapter, we show that such constructors can – sometimes with appropriate restrictions – be incorporated into several of the most prominent DL languages, thereby significantly enhancing expressivity without increasing worst-case complexity of standard reasoning problems.

To illustrate this gain in expressivity, we give some examples of the modelling capabilities of role constructors:

Universal Role Using role negation, the universal role U that connects all individuals of a domain can be defined as $\top \times \top \sqsubseteq U$. Alternatively, it can be obtained as $\neg N \sqsubseteq U$, i.e. as the negation of the *empty role* N. The latter can readily be axiomatised by the GCI $\top \sqsubseteq \forall N.\bot$.

Role Conjunction This modelling feature allows us to describe situations where two individuals are interconnected by more than one role, a relationship that can not be captured in classical DLs that are confined to tree-like relations. For example, the fact that somebody testifying against a relative is not put under oath can be formalised by $\exists(\mathit{testifiesAgainst} \sqcap \mathit{relativeOf}).\top \sqsubseteq \neg \mathit{UnderOath}$. Likewise, role conjunction allows for specifying disjointness of roles, as $\mathsf{Dis}(R, S)$ can be paraphrased as $\top \sqsubseteq \forall(R \sqcap S).\bot$.

Concept Products The concept product statement $C \times D \sqsubseteq R$ expresses that every instance of C is connected to every instance of D via the role R. As an example, the fact that (all) alkaline solutions neutralise (all) acid solutions can be expressed by the concept product axiom $\mathit{AlkalineSolution} \times \mathit{AcidSolution} \sqsubseteq \mathit{neutralises}$. Using role negation, this can equivalently be stated by a GCI $\mathit{AlkalineSolution} \sqsubseteq \forall(\neg \mathit{neutralises}).\neg \mathit{AcidSolution}$.

Ranges and Domains Allowing concept products on the right-hand side of role inclusion axioms essentially allows us to specify range and domain restrictions on roles. For example, the fact that the role *authorOf* connects persons to publications can be expressed as $\mathit{authorOf} \sqsubseteq \mathit{Person} \times \mathit{Publication}$. Using GCIs, the same statements would be given by GCIs $\exists\, \mathit{authorOf}.\top \sqsubseteq \mathit{Person}$ and $\top \sqsubseteq \forall \mathit{authorOf}.\mathit{Publication}$.

Qualified Role Inclusion The specialisation of role inclusions based on concept memberships of the involved individuals can be expressed. The rule-like **FOL** statement $C(x) \wedge R(x, y) \wedge D(y) \rightarrow S(x, y)$, expressing that all R-related instances of C and D are also related by S, can be cast into the GCI $C \sqsubseteq \forall(R \sqcap \neg S).\neg D$. Alternatively, we can use a role inclusion axiom $(C \times D) \sqcap R \sqsubseteq S$. For example, the fact that any person of age having signed a contract which is legal is bound to that contract can be expressed by $(\mathit{OfAge} \times (\mathit{Contract} \sqcap \mathit{Legal})) \sqcap \mathit{hasSigned} \sqsubseteq \mathit{boundTo}$, or by $\mathit{OfAge} \sqsubseteq \forall(\mathit{hasSigned} \sqcap \neg \mathit{boundTo}).\neg(\mathit{Contract} \sqcap \mathit{Legal})$.

These examples also illustrate that role constructors can provide a more general view on expressive features like the universal role or role disjointness that are very common in DL today.

The outline of this chapter is as follows. The considered role constructors are introduced formally in Section 5.1 to obtain the DL $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ that provides a framework for the subsequent investigations. In Section 5.2, we show that constructors on simple roles can generally be added to $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$ without increasing the worst-case complexity of standard reasoning tasks. A similar result can be established for $\mathcal{SHIQ}$ in Section 5.3, but in this case we need to restrict to so-called *safe* role expressions to prevent increased reasoning complexities. In Section 5.4, we introduce the description logic $\mathcal{SROEL}(\sqcap_s, \times)$ as an extension of the well-known tractable DL $\mathcal{EL}^{++}$. In order to ensure that reasoning tasks can still be solved in polynomial time, we need to restrict to conjunctions of simple roles and to certain forms of admissible concept product axioms. We conclude this chapter with a summary in Section 5.5 and an overview of related work in Section 5.6.

Various results of this chapter were published in [RKH08a, RKH08b, Krö10].

5.1 Introducing Role Expressions

We now give a definition of the expressive description logic $\mathcal{SROIQ}(B_s, \times)$ which is obtained from the definition of $\mathcal{SROIQ}$ in Section 3.1 by allowing arbitrary Boolean constructors on simple roles and concept products. As before, we also define $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ as a DL that does not require the additional structural restrictions that are relevant for ensuring decidability of reasoning in $\mathcal{SROIQ}(B_s, \times)$ but which are not needed in all fragments of $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ that are considered below. Some further notes regarding our nomenclature are found below the following definition.

Definition 5.1.1 Consider a DL signature $\mathscr{S} = \langle \mathbf{I}, \mathbf{A}, \mathbf{N} \rangle$ with $\mathbf{N} = \mathbf{N}_s \cup \mathbf{N}_n$. The set $\mathbf{R}_s$ of $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ *simple role expressions* (or *simple roles*) for $\mathscr{S}$ is defined by the following grammar:

$$\mathbf{R}_s ::= U \mid (\mathbf{C} \times \mathbf{C}) \mid \mathbf{N}_s \mid \mathbf{N}_s^- \mid \neg\mathbf{R}_s \mid (\mathbf{R}_s \sqcap \mathbf{R}_s) \mid (\mathbf{R}_s \sqcup \mathbf{R}_s).$$

where $\mathbf{C}$ denotes the set of $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ concept expressions as defined below. A simple role expression is *safe* if it contains neither concept products nor the universal role, and if every disjunct in its disjunctive normal form contains at least one non-negated role name. The set $\mathbf{R}_n$ of *non-simple role expressions* is defined as $\mathbf{R}_n := \mathbf{N}_n \cup \{R^- \mid R \in \mathbf{N}_n\} \cup \{(C \times D) \mid C, D \in \mathbf{C}\}$. A $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ *role expression* (or *role*) is a simple role expression or a non-simple role expression, and the set $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ role expressions is denoted by $\mathbf{R} = \mathbf{R}_s \cup \mathbf{R}_n$.

The set $\mathbf{C}$ of $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ *concept expressions* (or *concepts*) for $\mathscr{S}$ is defined like the set of $\mathcal{SROIQ}^{\text{free}}$ concept expressions (Definition 3.1.1) but using $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ role expressions instead of $\mathcal{SROIQ}^{\text{free}}$ role expressions in all constructions.

$\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ RBox axioms, TBox axioms, ABox axioms, and knowledge bases are defined as in Definitions 3.1.2 and 3.1.3 using $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ role expressions instead of $\mathcal{SROIQ}^{\text{free}}$ role expressions in all constructions.

Name	Syntax	Semantics
inverse roles	R^-	$\{\langle x,y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle y,x\rangle \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
concept product	$C \times D$	$C^\mathcal{I} \times D^\mathcal{I}$
role negation	$\neg S$	$\{\langle x,y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle x,y\rangle \notin R^\mathcal{I}\}$
role conjunction	$S \sqcap T$	$S^\mathcal{I} \cap T^\mathcal{I}$
role disjunction	$S \sqcup T$	$S^\mathcal{I} \cup T^\mathcal{I}$

Figure 5.1: Semantics of $\mathcal{SROIQ}(B_\mathrm{s},\times)$ roles for an interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I}\rangle$

A $\mathcal{SROIQ}(B_\mathrm{s},\times)$ concept expression C is a $\mathcal{SROIQ}(B_\mathrm{s},\times)^\mathrm{free}$ concept expression such that all subconcepts D of C that are of the form $\exists S.\mathsf{Self}$, $\geqslant n\, S.E$, or $\leqslant n\, S.E$ are such that $S \in \mathbf{R}_\mathrm{s}$ is a simple role expression. $\mathcal{SROIQ}(B_\mathrm{s},\times)$ is the fragment of $\mathcal{SROIQ}(B_\mathrm{s},\times)^\mathrm{free}$ that contains only $\mathcal{SROIQ}(B_\mathrm{s},\times)$ concept expressions and regular RBoxes, where regularity of $\mathcal{SROIQ}(B_\mathrm{s},\times)^\mathrm{free}$ RBoxes is defined as for $\mathcal{SROIQ}^\mathrm{free}$ in Definition 3.1.4. $\diamond$

Note that the previous definition introduces concept products both as simple and as non-simple role expressions. In other words, we do not impose any restrictions on the use of concept products in axioms of $\mathcal{SROIQ}(B_\mathrm{s},\times)$.

DL nomenclature is already based on a number of non-systematic conventions, and the addition of role constructors imposes further challenges for coining suitable names for DLs. For better readability, information about role constructors is added in parentheses to the name of the underlying DL. The letter B represents Boolean role constructors, and $\times$ indicates the availability of the concept product. The lower-case b is used to denote safe Boolean role constructors, while the subscript s indicates that only simple role expressions are considered. When restricting to particular role constructors, we will simply list the according operator symbols in parentheses, as in the case of $\mathcal{SROEL}(\sqcap_\mathrm{s},\times)$ below.

Now the semantics of $\mathcal{SROIQ}(B_\mathrm{s},\times)$ is defined by extending the semantics of $\mathcal{SROIQ}$ to take role expressions into account.

Definition 5.1.2 A $\mathcal{SROIQ}^\mathrm{free}$ *interpretation* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I}\rangle$ as defined in Definition 3.1.5 is extended to $\mathcal{SROIQ}(B_\mathrm{s},\times)^\mathrm{free}$ role expressions as specified in Fig. 5.1.

Satisfaction of axioms and knowledge bases, as well as knowledge base consistency and entailment are defined as for $\mathcal{SROIQ}^\mathrm{free}$ in Definitions 3.1.6 and 3.1.7, using the extended interpretation for $\mathcal{SROIQ}(B_\mathrm{s},\times)^\mathrm{free}$ role expressions. $\diamond$

Based on this definition, it is obvious how to define a suitable generalisation of the function Inv that was introduced earlier to map a role expression to its inverse. Given a role name R, we define $\mathrm{Inv}(R) := R^-$ and $\mathrm{Inv}(R^-) := R$ as before. Concept products are treated by setting $\mathrm{Inv}(C \times D) := D \times C$. For other role expressions, we simply set $\mathrm{Inv}(U) := U$, $\mathrm{Inv}(R \sqcap S) := \mathrm{Inv}(R) \sqcap \mathrm{Inv}(S)$, $\mathrm{Inv}(R \sqcup S) := \mathrm{Inv}(R) \sqcup \mathrm{Inv}(S)$, and $\mathrm{Inv}(\neg R) := \neg\mathrm{Inv}(R)$. It is easy to see that, for any interpretation $\mathcal{I}$ and role expression R, we find $\mathrm{Inv}(R)^{\mathcal{I}} = \{\langle \epsilon, \delta \rangle \mid \langle \delta, \epsilon \rangle \in R^{\mathcal{I}}\}$ as desired.

Reasoning in description logics is typically based on certain structural properties of (relevant) models. In particular, it is typically possible to restrict attention to models that are tree-shaped or forest-shaped, possibly with some generalisations. Tableau algorithms typically attempt to construct (finite representations of) such models, and the tree-like structure of models can be exploited for finding termination criteria. When introducing role expressions, the structural properties of models are changed fundamentally, since role expressions like $\neg R$ may establish relations between hitherto unrelated parts of the model. In particular, any tree-like structure is lost when allowing for such expressions. Concept products have a similar effect, since they imply role relationships based solely on concept memberships while neglecting the relational structure of the model.

This is the motivation for introducing the notion of safety of role expressions in Definition 5.1.1. Indeed, the extension of safe role expressions is always a subset of the union of the extensions of atomic roles. Thus, intuitively speaking, safe role expressions never establish a relationship between hitherto unconnected parts of a model, and a general tree-shape can be preserved. However, even without this property, DLs with unsafe role expressions can still be decidable, as shown in the next section.

5.2 Role Expressions for $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$

In this section, we show that adding arbitrary (i.e. also unsafe) role expressions to the description logics $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$ does not increase their worst-case reasoning complexities – N2ExpTime [Kaz08] and NExpTime [Tob01], respectively – if the new role expressions are restricted to simple roles. In the sequel, $\mathcal{SHOIQ}$ (resp. $\mathcal{SHOIQ}(B_s, \times)$) will be treated as a special case of $\mathcal{SROIQ}$ (resp. $\mathcal{SROIQ}(B_s, \times)$), as most considerations hold for both cases. Our first result shows that we can easily restrict to $\mathcal{SROIQ}(B_s)$ and $\mathcal{SHOIQ}(B_s)$.

Proposition 5.2.1 *Every $\mathcal{SROIQ}(B_s, \times)$ ($\mathcal{SHOIQ}(B_s, \times)$) knowledge base* KB *is semantically emulated by a $\mathcal{SROIQ}(B_s)$ ($\mathcal{SHOIQ}(B_s)$) knowledge base* KB′ *that can be computed in linear time w.r.t. the size of* KB.

Proof. KB′ is obtained by iteratively eliminating concept products from KB. Initialise KB′ := KB. In each step, select one concept product $C \times D$ that occurs as a sub-expression in some axioms of KB′ such that C, D do not contain subexpressions with concept products. Introduce fresh role names $R_{C\times D} \in \mathbf{N}_n$ and

$A \sqsubseteq \forall \hat{S}.B$	$\mapsto$	$\{A \sqsubseteq \forall T.B, \hat{S} \sqsubseteq T\}$
$A \sqsubseteq \geqslant n \hat{S}.B$	$\mapsto$	$\{A \sqsubseteq \geqslant n T.B, T \sqsubseteq \hat{S}\}$
$A \sqsubseteq \leqslant n \hat{S}.B$	$\mapsto$	$\{A \sqsubseteq \leqslant n T.B, \hat{S} \sqsubseteq T\}$
$A \sqsubseteq \exists \hat{S}.\mathsf{Self}$	$\mapsto$	$\{A \sqsubseteq \exists T.\mathsf{Self}, T \sqsubseteq \hat{S}\}$
$\mathsf{Dis}(S, S')$	$\mapsto$	$\{S \sqcap S' \sqsubseteq T, \top \sqsubseteq \forall T.\bot\}$
$R_1 \circ \ldots \circ \hat{S} \circ \ldots \circ R_n \sqsubseteq R$	$\mapsto$	$\{R_1 \circ \ldots \circ T \circ \ldots \circ R_n \sqsubseteq R, \hat{S} \sqsubseteq T\}$

$A, B \in \mathbf{A}$ concept names, $R \in \mathbf{N}$ a role name, $R_1, \ldots, R_n \in \mathbf{R}$ role expressions,
$S, S', \hat{S} \in \mathbf{R}_s$ simple role expressions where $\hat{S} \notin \mathbf{N}_s$ is not a role name,
$T \in \mathbf{N}_s$ a fresh simple role name

Figure 5.2: Normal form transformation for $\mathcal{SROIQ}(B_s)$ axioms

$S_{C\times D} \in \mathbf{N}_s$, and extend KB′ with the GCIs $C \sqsubseteq \forall(\neg S_{C\times D}).\neg D$, $\exists R_{C\times D}.\top \sqsubseteq C$ and $\top \sqsubseteq \forall R_{C\times D}.D$, and with a RIA $S_{C\times D} \sqsubseteq R_{C\times D}$. Then replace all uses of $C \times D$ in the RBox of KB′ with $R_{C\times D}$, and all uses in the TBox of KB′ with $S_{C\times D}$. It is easy to see that KB′ entails $R_{C\times D} \sqsubseteq S_{C\times D}$ and $S_{C\times D} \sqsubseteq R_{C\times D}$ (using two roles is necessary to avoid any violation of restrictions on simple roles). Hence KB′ semantically emulates KB at each stage of the computation. Clearly, the transformation terminates after linear time to return a $\mathcal{SROIQ}(B_s)$ ($\mathcal{SHOIQ}(B_s)$) knowledge base. □

We will therefore disregard concept products for the remaining arguments. As shown in [Kaz08], any $\mathcal{SROIQ}$ ($\mathcal{SHOIQ}$) knowledge base can be transformed into an equisatisfiable knowledge base containing only axioms of the form:

$$\begin{array}{lcl} A \sqsubseteq \forall R.B & \bigsqcap A_i \sqsubseteq \bigsqcup B_j & S_1 \sqsubseteq S_2 \\ A \sqsubseteq \geqslant n S.B & A \equiv \{a\} & S_1 \sqsubseteq S_2^- \\ A \sqsubseteq \leqslant n S.B & A \equiv \exists S.\mathsf{Self} & \mathsf{Dis}(S_1, S_2) \\ & R_1 \circ \ldots \circ R_n \sqsubseteq R. & \end{array}$$

where $R \in \mathbf{N}$ and $S, S_1, S_2 \in \mathbf{N}_s$. In fact, it is easy to see that the transformed knowledge base semantically emulates the original one. The same normalisation can be applied to $\mathcal{SROIQ}(B_s)$ ($\mathcal{SHOIQ}(B_s)$) as well, yielding the same types of axioms but with $\mathcal{SROIQ}(B_s)$ role expressions in the place of $\mathcal{SROIQ}$ roles. A second transformation is carried out by exhaustively applying the transformation steps depicted in Fig. 5.2. As a result, we obtain a normalised knowledge base that obviously still semantically emulates the original knowledge base. The normalised knowledge base contains only the original axiom types depicted above (using simple role names in places of $S_{(i)}$ and role names in places of R_i) and one additional axiom type $S \sqsubseteq T$ with $S, T \in \mathbf{R}_s$ simple role expressions. As shown in [Kaz08], any of these original axiom types except the one containing role concatenation can be translated into C^2, the two-variable fragment of first order logic with counting quantifiers. The remaining axioms of the new type can be transformed into C^2 statements $\forall x, y.(\pi(S, x, y) \to \pi(T, x, y))$ where we extend the first-order mapping π that was defined for $\mathcal{SROIQ}$ role expressions in Fig. 3.4 as follows:

$$\begin{aligned}
\pi(U, x, y) &= \top \\
\pi(R, x, y) &= R(x, y) \text{ if } R \in \mathbf{N} \\
\pi(R^-, x, y) &= \pi(R, y, x) \\
\pi(\neg S, x, y) &= \neg\pi(S, x, y) \\
\pi(S \sqcap T, x, y) &= \pi(S, x, y) \wedge \pi(T, x, y) \\
\pi(S \sqcup T, x, y) &= \pi(S, x, y) \vee \pi(T, x, y)
\end{aligned}$$

Note that the remaining definition of π that was given in Fig. 3.4 readily provides us with a mapping from $\mathcal{SROIQ}(B_s)$ to $\mathbf{FOL}_\approx$, whereas it is not suitable as a transformation to C^2.

Further following the argumentation from [Kaz08], the remaining complex role inclusions not directly convertible into C^2 can be taken into account by cautiously materialising the consequences resulting from their interplay with axioms of the type $A \sqsubseteq \forall R.B$ through automata encoding techniques. Further details on this part of the transformation are provided in Section 9.3. This way, one obtains a C^2 theory that is satisfiable exactly if the original knowledge base is. In the case of $\mathcal{SROIQ}$ (and hence $\mathcal{SROIQ}(B_s)$), this can result in an exponential blow-up of the knowledge base while for $\mathcal{SHOIQ}(B_s)$ (and hence $\mathcal{SHOIQ}$) the transformation is time-polynomial. Thus we see that the upper complexity bounds for $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$ carry over to $\mathcal{SROIQ}(B_s)$ and $\mathcal{SHOIQ}(B_s)$ by just a slight extension of the according proofs from [Kaz08] while the lower bounds follow directly from those of $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$. Together with Proposition 5.2.1, we thus establish the following theorem.

Theorem 5.2.2 *All standard reasoning tasks for $\mathcal{SROIQ}(B_s, \times)$ are* N2ExpTime-*complete, and all standard reasoning tasks for $\mathcal{SHOIQ}(B_s, \times)$ are* NExpTime-*complete w.r.t. the size of the knowledge base.*

While the results established in this section are rather straightforward consequences of known results, their implications for practice might be more significant: they show that the DLs underlying OWL and OWL 2 can be extended by arbitrary Boolean constructors on simple roles without increasing the worst-case complexity of reasoning. On the other hand, worst-case complexity estimations do of course not suffice to show practical utility. In particular, we are not aware of any dedicated inference engine for C^2.

5.3 Safe Role Expressions for $\mathcal{SHIQ}$

$\mathcal{SHIQ}$ is a rather expressive fragment obtained from $\mathcal{SHOIQ}$ by disallowing nominals. In contrast to the NExpTime-completeness encountered with $\mathcal{SHOIQ}$, the worst-case complexity of standard reasoning tasks in $\mathcal{SHIQ}$ is known to be ExpTime [Tob01]. In this section, we introduce the extension of $\mathcal{SHIQ}$ by safe role expressions on simple roles. Thereafter, we present a technique for eliminating transitivity statements in $\mathcal{SHIQ}(b_s)$ knowledge bases while preserving satisfiability. Together with the observation that hierarchies of (simple) roles can be expressed in terms of safe Boolean role expressions, this yields two results. On the one hand, we provide a way to use existing reasoning procedures for $\mathcal{ALCIQ}(b)$, such as the ones described in [Tob01, CEO07, RKH08d], to solve $\mathcal{SHIQ}(b_s)$ reasoning tasks. On the other hand, as the transformation is possible in polynomial time, the known upper bound for the complexity of reasoning in $\mathcal{ALCIQ}(b)$ – namely ExpTime – carries over to $\mathcal{SHIQ}(b_s)$. This result depends on the safety condition on role expressions since dropping it would lead to a DL that subsumes $\mathcal{ALC}(B)$ for which reasoning is known to be NExpTime-complete [LS02].

Definition 5.3.1 A $\mathcal{SHIQ}(b_s)$ knowledge base is a $\mathcal{SHOIQ}(B_s)$ knowledge base that contains no nominals and only safe role expressions. $\diamond$

We now show that any $\mathcal{SHIQ}(b_s)$ knowledge base can be transformed into an equisatisfiable knowledge base not containing transitivity statements. This slightly generalises according results from [Tob01, Mot06] as we allow safe Boolean expressions in GCIs and role inclusion axioms already for the source DL.

For a fixed $\mathcal{SHIQ}(b_s)$ knowledge base KB, let $\sqsubseteq^*$ denote the least partial order on $\mathbf{N}_n$ such that $R \sqsubseteq^* S$ and $\mathrm{Inv}(R) \sqsubseteq^* \mathrm{Inv}(S)$ for every RBox axiom $R \sqsubseteq S \in$ KB. In other words, $\sqsubseteq^*$ is the reflexive transitive closure of the role hierarchy on non-simple roles that is specified by the RBox of the knowledge base.

Definition 5.3.2 Given a $\mathcal{SHIQ}(b_s)$ knowledge base KB, let $\mathsf{clos}(\mathrm{KB})$ denote the smallest set of concept expressions where

- $\mathsf{NNF}(\neg C \sqcup D) \in \mathsf{clos}(\mathrm{KB})$ for any TBox axiom $C \sqsubseteq D$,
- $D \in \mathsf{clos}(\mathrm{KB})$ for every subconcept D of some concept $C \in \mathsf{clos}(\mathrm{KB})$,
- $\mathsf{NNF}(\neg C) \in \mathsf{clos}(\mathrm{KB})$ for any $\leqslant n\, R.C \in \mathsf{clos}(\mathrm{KB})$,
- $\forall S.C \in \mathsf{clos}(\mathrm{KB})$ whenever $\mathsf{Tra}(S) \in$ KB and $S \sqsubseteq^* R$ for a role R with $\forall R.C \in \mathsf{clos}(\mathrm{KB})$.

Moreover, let $\Omega(\mathrm{KB})$ denote the knowledge base obtained from KB by

- removing all transitivity axioms $\mathsf{Tra}(S)$, and

– adding the axiom $\forall R.C \sqsubseteq \forall S.(\forall S.C)$ for every $\forall R.C \in \mathsf{clos}(\mathrm{KB})$ with $\mathsf{Tra}(S) \in \mathrm{KB}$ and $S \sqsubseteq^* R$. $\diamondsuit$

Proposition 5.3.3 *Every $\mathcal{SHIQ}(b_s)$ knowledge base* KB *is equisatisfiable to the corresponding knowledge base* $\Omega(\mathrm{KB})$.

Proof. Obviously, we have that $\mathrm{KB} \models \Omega(\mathrm{KB})$, and hence every model of KB is also a model of $\Omega(\mathrm{KB})$.

For the other direction, consider a model $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ of $\Omega(\mathrm{KB})$. We define a new interpretation $\mathcal{J} = (\Delta^\mathcal{J}, \cdot^\mathcal{J})$ as follows:

- $\Delta^\mathcal{J} := \Delta^\mathcal{I}$,
- for individual names $a \in \mathbf{I}$, we set $a^\mathcal{J} := a^\mathcal{I}$,
- for concept names $A \in \mathbf{A}$, we set $A^\mathcal{J} := A^\mathcal{I}$,
- for simple role names $S \in \mathbf{N}_\mathrm{s}$, we set $S^\mathcal{J} := S^\mathcal{I}$,
- for non-simple role names $R \in \mathbf{N}_\mathrm{n}$, we define $R^\mathcal{J}$ to be the transitive closure of $R^\mathcal{I}$ if $\mathsf{Tra}(R) \in \mathrm{KB}$, and we define $R^\mathcal{J} := R^\mathcal{I} \cup \bigcup_{S \sqsubseteq^* R, S \in \mathbf{N}_\mathrm{n}} S^\mathcal{J}$ otherwise.

As a direct consequence of this definition, note that for all simple role expressions $S \in \mathbf{R}_\mathrm{s}$ we have $S^\mathcal{J} = S^\mathcal{I}$ (†).

We now prove that $\mathcal{J}$ is a model of KB by considering all axioms, starting with the RBox. Every transitivity axiom of KB is clearly satisfied by definition of $\mathcal{J}$. We need to show that every role inclusion $S \sqsubseteq T$ axiom is also satisfied. Indeed, if both S and T are simple role expressions this is a trivial consequence of (†). If S is a simple role expression and T is a non-simple role, the claim follows from (†) and the fact that, by construction of $\mathcal{J}$, for every non-simple role R we find that $R^\mathcal{I} \subseteq R^\mathcal{J}$. It remains to consider the case that both S and T are non-simple roles. If T is not transitive, the claim follows directly from the definition. Otherwise, the desired conclusion follows from the fact that the transitive closure is a monotone operation w.r.t. set inclusion.

We proceed by examining the concept expressions $C \in \mathsf{clos}(\mathrm{KB})$ and show via structural induction that $C^\mathcal{I} \subseteq C^\mathcal{J}$. As base case, for every concept of the form A or $\neg A$ with $A \in \mathbf{A}$, this claim follows directly from the definition of $\mathcal{J}$. We proceed with the induction steps for all possible forms of a complex concept C (mark that all $C \in \mathsf{clos}(\mathrm{KB})$ are in negation normal form):

- Clearly, if $D_1^\mathcal{I} \subseteq D_1^\mathcal{J}$ and $D_2^\mathcal{I} \subseteq D_2^\mathcal{J}$ by induction hypothesis, we can directly conclude $(D_1 \sqcap D_2)^\mathcal{I} \subseteq (D_1 \sqcap D_2)^\mathcal{J}$ as well as $(D_1 \sqcup D_2)^\mathcal{I} \subseteq (D_1 \sqcup D_2)^\mathcal{J}$.
- Likewise, as we have $S^\mathcal{I} \subseteq S^\mathcal{J}$ for all role expressions S, and again $D^\mathcal{I} \subseteq D^\mathcal{J}$ by the induction hypothesis, we obtain $(\exists S.D)^\mathcal{I} \subseteq (\exists S.D)^\mathcal{J}$ and $(\geqslant n\, S.D)^\mathcal{I} \subseteq (\geqslant n\, S.D)^\mathcal{J}$.

– Now, consider $C = \forall S.D$. If S is a simple role expression, we know that $S^{\mathcal{J}} = S^{\mathcal{I}}$, whence we can derive $(\forall S.D)^{\mathcal{I}} \subseteq (\forall S.D)^{\mathcal{J}}$ from the induction hypothesis.

  It remains to consider the case $C = \forall R.D$ for non-simple role expressions R. Assume $\delta \in (\forall R.D)^{\mathcal{I}}$. If there is no δ' with $(\delta, \delta') \in R^{\mathcal{J}}$, then $\delta \in (\forall R.D)^{\mathcal{J}}$ is trivially true. Now assume there are such δ'. For each of them, we can distinguish two cases:

  - $(\delta, \delta') \in R^{\mathcal{I}}$. Then $\delta' \in D^{\mathcal{I}}$ and, by the induction hypothesis, $\delta' \in D^{\mathcal{J}}$,
  - $(\delta, \delta') \notin R^{\mathcal{I}}$. By construction of $\mathcal{J}$, this means that there is a role S with $S \sqsubseteq^* R$ and $\mathsf{Tra}(S) \in$ KB and a sequence $\delta = \delta_0, \ldots, \delta_n = \delta'$ with $(\delta_k, \delta_{k+1}) \in S^{\mathcal{I}}$ for all $0 \leq k < n$. By definition of Ω, the knowledge base Ω(KB) contains the axiom $\forall R.D \sqsubseteq \forall S.(\forall S.D)$, hence we have $\delta \in \forall S.(\forall S.D)$ wherefrom a simple inductive argument ensures $\delta_k \in D^{\mathcal{I}}$ for all δ_k including $\delta_n = \delta'$.

  So we can conclude that for all such δ' we have $\delta' \in D^{\mathcal{I}}$. Via the induction hypothesis follows $\delta \in D^{\mathcal{J}}$ and hence we can conclude $\delta \in (\forall R.D)^{\mathcal{J}}$.

– Finally, consider $C = \leqslant n\, R.D$ and assume $\delta \in (\leqslant n\, R.D)^{\mathcal{I}}$. Since R must be simple, we obtain $R^{\mathcal{J}} = R^{\mathcal{I}}$. Moreover, since both D and $\mathsf{NNF}(\neg D)$ are contained in clos(KB) the induction hypothesis gives $D^{\mathcal{J}} = D^{\mathcal{I}}$. Those two facts together directly imply $\delta \in (\leqslant n\, R.D)^{\mathcal{I}}$.

Now considering an arbitrary KB TBox axiom $C \sqsubseteq D$, we find $(\mathsf{NNF}(\neg C) \sqcup D)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ as $\mathcal{I}$ is a model of KB. Moreover – by the correspondence just shown – we have $(\mathsf{NNF}(\neg C) \sqcup D)^{\mathcal{I}} \subseteq (\mathsf{NNF}(\neg C) \sqcup D)^{\mathcal{J}}$ and hence also $(\mathsf{NNF}(\neg C) \sqcup D)^{\mathcal{J}} = \Delta^{\mathcal{J}}$, so that $C \sqsubseteq D$ must be satisfied in $\mathcal{J}$ as required. □

Taking into account that the presented transformation is time-polynomial, this result can now be employed to determine the complexity of $\mathcal{SHIQ}(b_s)$.

Theorem 5.3.4 *All standard reasoning tasks for $\mathcal{SHIQ}(b_s)$ knowledge bases are* ExpTime*-complete w.r.t. the size of the considered knowledge bases.*

Proof. As shown in Proposition 3.1.9, all standard reasoning problems can be reduced to knowledge base satisfiability checking. Using Proposition 5.3.3, any given $\mathcal{SHIQ}(b_s)$ knowledge base KB can be transformed into an equisatisfiable $\mathcal{ALCHIQ}(b)$ knowledge base Ω(KB) in polynomial time. Furthermore, all role inclusion axioms can be removed from Ω(KB) as follows. First, all role names contained in Ω(KB) can be declared to be simple without violating the syntactic constraints. Second, every role inclusion axiom $S \sqsubseteq T$ (with $S, T \in \mathbf{R}_s$ being safe by definition) can be equivalently transformed into a GCI $\top \sqsubseteq \forall(S \sqcap \neg T).\bot$. Note that $S \sqcap \neg T$ is admissible here since it is necessarily safe. Moreover the transformation is obviously time-linear. Hence we obtain an $\mathcal{ALCIQ}(b)$ knowledge base. The satisfiability checking problem for $\mathcal{ALCIQ}(b)$ is known to be ExpTime-complete [Tob01]. □

We thus have shown that allowing safe Boolean expressions on simple roles does not increase the ExpTime reasoning complexity of $\mathcal{SHIQ}$. This is not the case when Boolean expressions of non-simple roles are allowed: as shown in [GK08], standard inference tasks become 2ExpTime-complete when extending $\mathcal{SHIQ}$ with conjunctions of non-simple roles.

5.4 A Tractable DL with Role Expressions

The tractable description logic $\mathcal{EL}^{++}$ is an extension of $\mathcal{EL}$ with nominals, role inclusion axioms, and suitable concrete domains that allow for the representation of datatypes and related predicates [BBL05]. We do not investigate the latter within this work, and we thus focus on the DL $\mathcal{ELRO}$ that is obtained as the intersection of $\mathcal{EL}^{++}$ and $\mathcal{SROIQ}^{\text{free}}$. In this section, we show that standard reasoning tasks can still be performed in polynomial time if $\mathcal{ELRO}$ is extended with role conjunction, concept products, local reflexivity (Self), the universal role, and role disjointness, given that certain structural restrictions on concept products are added. The basic underlying description logic without these restrictions is called $\mathcal{SROEL}(\sqcap_s, \times)$.

Definition 5.4.1 A $\mathcal{SROEL}(\sqcap_s, \times)$ role expression is a $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ role expression that may contain the universal role (U), role conjunction ($\sqcap$), and concept products ($\times$), but no role disjunction ($\sqcup$), negation ($\neg$), or inverses ($\cdot^-$). A $\mathcal{SROEL}(\sqcap_s, \times)$ concept expression is a $\mathcal{SROIQ}(B_s)^{\text{free}}$ concept expression that contains only top ($\top$), bottom ($\bot$), concept names, nominal concepts, conjunction ($\sqcap$), local reflexivity (Self), or existential role restrictions ($\exists$) on $\mathcal{SROEL}(\sqcap_s, \times)$ role expressions.

$\mathcal{SROEL}(\sqcap_s, \times)$ is the fragment of $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ that contains only role and concept expressions of $\mathcal{SROEL}(\sqcap_s, \times)$, as well as role assertions of the form $\mathsf{Tra}(R)$, $\mathsf{Dis}(S_1, S_2)$, $\mathsf{Irr}(S)$, and $\mathsf{Ref}(R)$, with $R \in \mathbf{R}$ and $S_{(i)} \in \mathbf{R}_s$. ◇

Note that we do not have any requirement for regularity of roles but we have to introduce the notion of role simplicity in the context of $\mathcal{SROEL}(\sqcap_s, \times)$. Besides the new role operators, we have introduced some other features of $\mathcal{SROIQ}$ which are convenient when applying some of our constructions in later chapters to $\mathcal{SROEL}(\sqcap_s, \times)$. It is easy to see that most of the additional features are not increasing the expressive power: the universal role can be expressed as $\top \times \top$, role disjointness can be expressed as $S_1 \sqcap S_2 \sqsubseteq N$ with the empty role N axiomatised as $\exists N.\top \sqsubseteq \bot$, and the remaining role characteristics can be expressed as discussed in Section 3.1.3. Accordingly, we will not explicitly consider these features below, whereas local reflexivity must be taken into account explicitly.

Unfortunately, the standard reasoning problems for $\mathcal{SROEL}(\sqcap_s, \times)$ cannot be solved in polynomial time. Indeed, it is not hard to see that GCIs of the form $C \sqsubseteq \forall R.D$ can be emulated in $\mathcal{SROIQ}(B_s, \times)$ by knowledge bases $\{C \times \top \sqsubseteq V, R \sqcap V \sqsubseteq \top \times D\}$ where V is a fresh role name. It is well-known that reasoning in $\mathcal{EL}^{++}$ becomes ExpTime-hard when extended with universal role restrictions of this form. This is also a direct consequence of the ExpTime-hardness of reasoning in the logic Horn-$\mathcal{FLE}$ that is shown in Section 6.4, which shows that the problem persists even if all roles of a knowledge base are simple. Since we are interested in a tractable DL with role constructors in this section, we will introduce a class of $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases with additional restrictions below.

To simplify our investigations, we first observe that any $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base can be converted into a normal form.

Definition 5.4.2 A $\mathcal{SROEL}(\sqcap_s, \times)$ concept expression is *basic* if it is a concept name or nominal, and the according set $\mathbf{B}$ of basic concepts thus is $\mathbf{B} = \mathbf{A} \cup \{\{a\} \mid a \in \mathbf{I}\}$. A $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base KB is in *normal form* if it contains only axioms of one of the following forms:

$$A \sqsubseteq C \qquad A \sqcap B \sqsubseteq C \qquad \exists R.A \sqsubseteq C \qquad A \sqsubseteq \exists R.B \qquad \exists R.\mathsf{Self} \sqsubseteq C \qquad A \sqsubseteq \exists R.\mathsf{Self}$$
$$R \sqsubseteq T \qquad R \circ S \sqsubseteq T \qquad R \sqcap S \sqsubseteq T \qquad A \times B \sqsubseteq R \qquad R \sqsubseteq C \times D$$

where $A, B \in \mathbf{B} \cup \{\top\}$, $C, D \in \mathbf{B} \cup \{\bot\}$, and $R, S, T \in \mathbf{N}$. ◇

Proposition 5.4.3 *Every $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base* KB *is semantically emulated by a $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base in normal form that can be computed in linear time with respect to the size of* KB.

Proof. The elimination of role characteristics has already been discussed above, and ABox axioms can readily be expressed as discussed in Section 3.1.3. We thus assume that KB contains only $\mathcal{SROEL}(\sqcap_s, \times)$ GCIs and RIAs. The transformation is accomplished by exhaustively applying the rules of Fig. 5.3, where each rule describes the replacement of the axiom on the right-hand side by the set of axioms on the left-hand side. It is easy to see that the resulting axioms semantically emulate the original axioms for each rule, so the result follows by induction. It is also easy to see that only a linear number of steps are required, where it is important to note that the rules for $S \sqsubseteq T \sqcap R$ and $A \sqsubseteq C \sqcap D$ are only applicable if S and A are no compound expressions, so that the duplication of S and A still leads to only a linear increase in size. □

$R_1 \circ \ldots \circ R_{n-1} \circ R_n$	$\sqsubseteq$	S	$\mapsto$	$\{R_1 \circ \ldots \circ R_{n-1} \sqsubseteq V, V \circ R_n \sqsubseteq S\}$
$\hat{R}_1 \circ R_2$	$\sqsubseteq$	S	$\mapsto$	$\{\hat{R}_1 \sqsubseteq V, V \circ R_2 \sqsubseteq S\}$
$R_1 \circ \hat{R}_2$	$\sqsubseteq$	S	$\mapsto$	$\{\hat{R}_2 \sqsubseteq V, R_1 \circ V \sqsubseteq S\}$
$\hat{R}_1$	$\sqsubseteq$	$\hat{R}_2$	$\mapsto$	$\{\hat{R}_1 \sqsubseteq V, V \sqsubseteq \hat{R}_2\}$
S	$\sqsubseteq$	$T \sqcap R$	$\mapsto$	$\{S \sqsubseteq T, S \sqsubseteq R\}$
$\hat{R} \sqcap S$	$\sqsubseteq$	T	$\mapsto$	$\{\hat{R} \sqsubseteq V, V \sqcap S \sqsubseteq T\}$
$\hat{C} \times D$	$\sqsubseteq$	T	$\mapsto$	$\{\hat{C} \sqsubseteq X, X \times D \sqsubseteq T\}$
$C \times \hat{D}$	$\sqsubseteq$	T	$\mapsto$	$\{\hat{D} \sqsubseteq X, C \times X \sqsubseteq T\}$
T	$\sqsubseteq$	$\hat{C} \times D$	$\mapsto$	$\{X \sqsubseteq \hat{C}, T \sqsubseteq X \times D\}$
T	$\sqsubseteq$	$C \times \hat{D}$	$\mapsto$	$\{X \sqsubseteq \hat{D}, T \sqsubseteq C \times X\}$
$\hat{C}$	$\sqsubseteq$	$\hat{D}$	$\mapsto$	$\{\hat{C} \sqsubseteq X, X \sqsubseteq \hat{D}\}$
C	$\sqsubseteq$	$\top$	$\mapsto$	$\emptyset$
$\bot$	$\sqsubseteq$	C	$\mapsto$	$\emptyset$
$\hat{C} \sqcap A$	$\sqsubseteq$	B	$\mapsto$	$\{\hat{C} \sqsubseteq X, X \sqcap A \sqsubseteq B\}$
A	$\sqsubseteq$	$C \sqcap D$	$\mapsto$	$\{A \sqsubseteq C, A \sqsubseteq D\}$
$\exists R.\hat{C}$	$\sqsubseteq$	A	$\mapsto$	$\{\hat{C} \sqsubseteq X, \exists R.X \sqsubseteq A\}$
A	$\sqsubseteq$	$\exists R.\hat{C}$	$\mapsto$	$\{A \sqsubseteq \exists R.X, X \sqsubseteq \hat{C}\}$
$\exists \hat{R}.C$	$\sqsubseteq$	A	$\mapsto$	$\{\hat{R} \sqsubseteq V, \exists V.C \sqsubseteq A\}$
A	$\sqsubseteq$	$\exists \hat{R}.C$	$\mapsto$	$\{A \sqsubseteq \exists V.C, V \sqsubseteq \hat{R}\}$
$\exists \hat{R}.\mathsf{Self}$	$\sqsubseteq$	A	$\mapsto$	$\{\hat{R} \sqsubseteq V, \exists V.\mathsf{Self} \sqsubseteq A\}$
A	$\sqsubseteq$	$\exists \hat{R}.\mathsf{Self}$	$\mapsto$	$\{A \sqsubseteq \exists V.\mathsf{Self}, V \sqsubseteq \hat{R}\}$

A, B basic concept expressions, $\top$, or $\bot$; X a fresh concept name;
$C, D, \hat{C}, \hat{D}$ concept expressions where $\hat{C}, \hat{D}$ are not basic; S, T role names;
$R_{(i)}, \hat{R}_{(i)}$ role expressions where $\hat{R}_{(i)}$ are no role names; V a fresh role name

Figure 5.3: Normal form transformation for $\mathcal{SROEL}(\sqcap_s, \times)$

The above example showed that the interplay of concept products and role conjunctions is sufficiently complex to emulate universal restrictions. This is not surprising given that we have already noted in the initial discussion in this chapter that concept products on the right-hand side of RIAs can express range restrictions $\top \sqsubseteq \forall R.A$. It may be surprising, however, that the restriction to simple role expressions on the left-hand side of such RIAs is not sufficient to retain tractability, since this is known to be the case in $\mathcal{EL}^{++}$ if no role conjunctions are allowed [BBL08]. In the context of the cited work, a criterion for the *admissibility* of range restrictions in $\mathcal{EL}^{++}$ has been defined, and the following definition provides a similar criterion for the case of $\mathcal{SROEL}(\sqcap_s, \times)$.

Definition 5.4.4 Consider a $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base KB in normal form, and define $\mathrm{KB}_R := \{R \sqsubseteq S \in \mathrm{KB} \mid R, S \in \mathbf{N}\}$. For every role name R, the set

Axiom	Datalog Rules
$A \sqsubseteq C$	$\hat{A}(x) \to \hat{C}(x)$
$A \sqcap B \sqsubseteq C$	$\hat{A}(x) \wedge \hat{B}(x) \to \hat{C}(x)$
$\exists R.A \sqsubseteq C$	$R(x,y) \wedge \hat{A}(y) \to \hat{C}(x)$
$A \sqsubseteq \exists R.B$	$\hat{A}(x) \to R(x, d_{R,B}), \hat{A}(x) \to \hat{B}(d_{R,B})$
$\exists R.\mathsf{Self} \sqsubseteq C$	$\mathsf{Self}_R(x) \to \hat{C}(x)$
$A \sqsubseteq \exists R.\mathsf{Self}$	$\hat{A}(x) \to \mathsf{Self}_R(x), \hat{A}(x) \to R(x,x)$
$R \sqsubseteq T$	$R(x,y) \to T(x,y), \mathsf{Self}_R(x) \to \mathsf{Self}_T(x)$
$R \circ S \sqsubseteq T$	$R(x,y) \wedge S(y,z) \to T(x,z)$
$R \sqcap S \sqsubseteq T$	$R(x,y) \wedge S(x,y) \to T(x,y), \mathsf{Self}_R(x) \wedge \mathsf{Self}_S(x) \to \mathsf{Self}_T(x)$
$A \times B \sqsubseteq R$	$\hat{A}(x) \wedge \hat{B}(y) \to R(x,y), \hat{A}(x) \wedge \hat{B}(x) \to \mathsf{Self}_R(x)$
$R \sqsubseteq C \times D$	$R(x,y) \to \hat{C}(x), R(x,y) \to \hat{D}(y)$
Rules with expressions Self_R are generated only if R is a simple role name	

Figure 5.4: Transforming $\mathcal{SROEL}(\sqcap_s, \times)$ to datalog

$\mathsf{ran}(R)$ is defined to contain exactly those concepts B for which there is a role S and concept name A such that $S \sqsubseteq A \times B \in \mathrm{KB}$ and $\mathrm{KB}_R \models R \sqsubseteq S$. The knowledge base KB is *admissible* if the following conditions are satisfied:

- for every RIA $R_1 \circ \ldots \circ R_n \sqsubseteq S \in \mathrm{KB}$ we have $\mathsf{ran}(S) \subseteq \mathsf{ran}(R_n)$, and
- for every RIA $R_1 \sqcap R_2 \sqsubseteq S \in \mathrm{KB}$ we have $\mathsf{ran}(S) \subseteq \mathsf{ran}(R_1) \cup \mathsf{ran}(R_2)$.

An arbitrary $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base is admissible if its normal form is admissible. ◇

It turns out that the standard reasoning problems for the class of admissible $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases are P-complete. To show this, we transform $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases into equisatisfiable datalog programs as follows.

Definition 5.4.5 Given a $\mathcal{SROIQ}(B_s, \times)$ knowledge base KB, a datalog program $\mathsf{P}(\mathrm{KB})$ is defined as follows. The following new symbols are introduced:

- concept names Self_R for each simple role $R \in \mathbf{R}_s$,
- individual names $d_{R,A}$ for each $R \in \mathbf{N}$ and $A \in \mathbf{B} \cup \{\top\}$.

Given a concept expression $C \in \mathbf{B} \cup \{\top, \bot\}$, we define a datalog atom $\hat{C}(x)$ as follows:

- $\hat{C}(x) := C(x)$ if $C \in \mathbf{A}$,
- $\hat{C}(x) := a \approx x$ if $C = \{a\}$ with $a \in \mathbf{I}$,
- $\hat{\top}(x) := \top$ and $\hat{\bot}(x) := \bot$.

Let KB′ denote the $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base in normal form obtained from KB as in Proposition 5.4.3. The datalog program $\mathsf{P}(\mathrm{KB})$ consists of

– a rule $R(a, a) \to \mathsf{Self}_R(a)$ for all $a \in \mathbf{I}$,

– for all axioms $\alpha \in \mathrm{KB}'$, the rules as indicated in Fig. 5.4. ◇

The following result states that $\mathsf{P}(\mathrm{KB})$ can be used to compute logical consequences of KB if it is an admissible knowledge base.

Proposition 5.4.6 *Let* KB *be some admissible* $\mathcal{SROEL}(\sqcap_s, \times)$ *knowledge base.*

– KB *is satisfiable iff* $\mathsf{P}(\mathrm{KB})$ *is satisfiable,*

– *for any* $A \in \mathbf{A}$ *and* $c \in \mathbf{I}$, *we find that* $\mathrm{KB} \models A(c)$ *iff* $\mathsf{P}(\mathrm{KB}) \models A(c)$,

– *for any* $A, B \in \mathbf{A}$, *we find that* $\mathrm{KB} \models A \sqsubseteq B$ *iff* $\mathsf{P}(\mathrm{KB}) \cup \{A(c)\} \models B(c)$ *for a fresh constant c.*

Proof. The proof of the first item of this proposition is a direct consequence of Theorem 8.5.6 in Chapter 8 that shows a similar result for a decidable SWRL fragment called $\mathcal{SROEL}(\sqcap_s, \times)$ *rules*. This later section also discusses that all admissible $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases can be expressed in $\mathcal{SROEL}(\sqcap_s, \times)$ rules; in particular, see the remarks on admissible range restrictions after Definition 8.5.1. Based on this correspondence, it is not hard to see that the transformation in Definition 5.4.5 is indeed a special case of the transformation for $\mathcal{SROEL}(\sqcap_s, \times)$ rules in Definition 8.5.3. The second item follows from the observation that $\mathrm{KB} \models A(c)$ iff $\mathrm{KB} \models \{c\} \sqsubseteq A$ iff $\mathrm{KB} \cup \{\{c\} \sqcap A \sqsubseteq \bot\} \models \bot$ iff $\mathsf{P}(\mathrm{KB}) \cup \{c \approx x \wedge A(x) \to \bot\} \models \bot$ iff $\mathsf{P}(\mathrm{KB}) \models A(c)$. The third item is obtained by similar reasoning. □

We point out that a more direct proof for a similar result has recently been given in [Krö10]. Either approach leads to the following complexity result.

Theorem 5.4.7 *All standard reasoning problems for admissible* $\mathcal{SROEL}(\sqcap_s, \times)$ *knowledge bases are* P*-complete w.r.t. the size of the considered knowledge base.*

Proof. The claim is a direct consequence of Proposition 5.4.6 together with the well-known fact that checking consistency of entailment of ground facts for datalog programs with a bounded number of variables per rule is P-complete (Fact 4.1.4). Indeed, it is easy to see that the rules of $\mathsf{P}(\mathrm{KB})$ as obtained in Definition 5.4.5 have at most three different variables. The fact that $\mathsf{P}(\mathrm{KB})$ is of polynomial size w.r.t. the size of KB follows from Proposition 5.4.3 and the observation that the number of auxiliary symbols $d_{R,A}$ and Self_R that are introduced in Definition 5.4.5 is polynomially bounded. □

We finish this section with some general remarks. First note that conjunction of roles enhances expressivity of $\mathcal{EL}^{++}$ significantly. For example, it allows for the following modelling features:

– Disjointness of two simple roles S, R. This feature, also provided by $\mathcal{SROIQ}$ as $\mathsf{Dis}(S, R)$, can be modelled in $\mathcal{EL}^{++}(\sqcap_s)$ with the axiom $\exists(S \sqcap R).\top \sqsubseteq \bot$.

– Atleast cardinality constraints on the right-hand side of a GCI. The axiom $A \sqsubseteq\ \geqslant n\, R.B$ can be modelled by the axiom set $\{R_i \sqsubseteq R, A \sqsubseteq \exists R_i.B \mid 1 \leq i \leq n\} \cup \{\exists(R_i \sqcap R_j).\top \sqsubseteq \bot \mid 1 \leq i < j \leq n\}$ where $R_1, \ldots, R_n$ are new simple role names.

It is worth noting that the special form of role conjunctions that are used for expressing disjointness of roles in both cases do not affect the admissibility of a $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base. In particular, we thus find that $\mathcal{SROEL}(\sqcap_s, \times)$ subsumes $\mathcal{EL}^{++}$ (without concrete domains) with additional admissible range restrictions. This shows that the $\mathcal{EL}^{++}$-based OWL 2 EL profile of the Web Ontology Language can be extended with role (i.e. "property") disjointness without loosing tractability. In contrast, it is easy to see that incorporating more than just conjunction on simple roles into $\mathcal{EL}^{++}$ would render the respective fragment intractable at best:

– Allowing conjunction on non-simple roles would even lead to undecidability as stated in Theorem 1 of [KRH07b].

– Allowing disjunction or negation on simple roles would allow to model disjunction on concepts: for instance, the GCI $A \sqsubseteq B \sqcup C$ can be expressed by the axiom set $\{A \sqsubseteq \exists(R \sqcup S).\top, \exists R.\top \sqsubseteq B, \exists S.\top \sqsubseteq C\}$, or by the axiom set $\{A \sqcap \exists R.\{a\} \sqsubseteq C, A \sqcap \exists\neg R.\{a\} \sqsubseteq B\}$ for new roles R, S and a new individual name a. Hence, every extension of $\mathcal{EL}^{++}$ with these features is ExpTime-hard [BBL05].

5.5 Summary

In this chapter, we have reviewed a number of role constructors that were proposed for description logics, and we have investigated cases where such constructors can be added to DLs without increasing the worst-case complexity of reasoning. For this purpose, role constructors have been restricted in various ways. In general, we considered only role constructors on simple roles, although it is not settled in all cases whether constructors on non-simple roles would actually lead to an increase in worst-case reasoning complexity. This restriction was sufficient for showing that reasoning $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$ – the DLs that subsume OWL 1 and OWL 2 – is still in NExpTime and N2ExpTime when adding role constructors. An extension of $\mathcal{SHIQ}$ with role constructors, in contrast, required us to further restrict to safe role expressions in order to retain ExpTime completeness of reasoning. In particular, safe role expressions do not comprise any concept products.

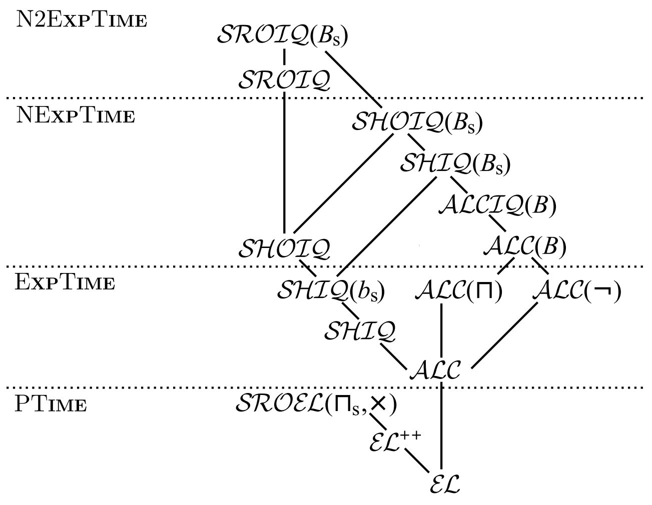

Figure 5.5: Reasoning complexities of DLs with role constructors, where $\mathcal{SROEL}(\sqcap_s, \times)$ represents the class of *admissible* knowledge bases of that DL

For the tractable description logic $\mathcal{EL}^{++}$, conjunctions of simple roles and certain concept products could be introduced to obtain the DL $\mathcal{SROEL}(\sqcap_s, \times)$ for which reasoning is still possible in polynomial time.

The complexity results of this chapter are summarised in Fig. 5.5. The DLs $\mathcal{SROIQ}(B_s, \times)$ and $\mathcal{SHOIQ}(B_s, \times)$ are not displayed, as it has been shown in Proposition 5.2.1 that they do not increase the expressivity of $\mathcal{SROIQ}(B_s)$ and $\mathcal{SHOIQ}(B_s)$, respectively. Likewise, we omit $\mathcal{ALCHIQ}(b)$ since it can directly be expressed in $\mathcal{ALCIQ}(b)$ as shown in Theorem 5.3.4. Moreover, the figure includes some additional description logics that have been studied in the literature (see Section 5.6), where we use the unified notation of Section 5.1. Note that $\mathcal{SROEL}(\sqcap_s, \times)$ is not placed below $\mathcal{SROIQ}(B_s)$ since it does not impose regularity restrictions. We also point out that the position of $\mathcal{EL}^{++}$ below $\mathcal{SROEL}(\sqcap_s, \times)$ ignores the presence of concrete domains in $\mathcal{EL}^{++}$ – as discussed in Section 5.4, the name $\mathcal{ELRO}$ would be more accurate for the corresponding DL.

For the case of $\mathcal{SHIQ}(b_s)$, our results provide a direct way for adapting existing reasoning algorithms for $\mathcal{SHIQ}$ by means of a suitable pre-processing. Likewise, the datalog translation that was introduced for $\mathcal{SROEL}(\sqcap_s, \times)$ provides a promising approach for efficient implementations based on datalog systems. For $\mathcal{SROIQ}(B_s)$ and $\mathcal{SHOIQ}(B_s)$, in contrast, the design of efficient algorithms is left as an interesting direction of future research, since our proof methods in this case do not suggest a practically feasible implementation strategy.

To the best of our knowledge, the complexity results on $\mathcal{SROEL}(\sqcap_s, \times)$ are also the first that establish the tractability of reasoning for a description logic that comprises all expressive features of the OWL 2 EL ontology language.

5.6 Related Work

Concept products have been considered as a role construction operator in early works on description logics (see, e.g. [BBH+90]), and the relationship to domain and range restrictions has been well known. In fact, early works seem to use concept product description mainly as a convenient syntax for concurrently specifying domain and range of a role. Discussions of the impact of concept products on reasoning complexities appear to be much more recent. [Kaz06] introduces a resolution-based algorithm for reasoning in $\mathcal{EL}$ extended with axioms of the form $C \times D \sqsubseteq R$, and the term *cross-products of concepts* is used within this work. A similar extension for $\mathcal{EL}^{++}$ is presented in [RKH08a], and a reasoning algorithm is developed by extending the procedure that was given in [BBL05]. Furthermore, [RKH08a] discusses how concept product axioms can be emulated in $\mathcal{SROIQ}$.

Neither [Kaz06] nor [RKH08a] consider concept products on the right-hand sides of RIAs. Axioms of this kind are more closely related to range and domain restrictions, and the according extension of $\mathcal{EL}^{++}$ has been introduced in [BBL08]. The latter work also shows tractability of reasoning for a large sublanguage of the DL that is underlying the OWL 2 EL ontology language, but it still lacks local reflexivity, the universal role, and disjointness of roles. To the best of our knowledge, our work on $\mathcal{SROEL}(\sqcap_s, \times)$ is the first to establish tractability of a DL that comprises all of these features. Most recently, we have further explicated reasoning in $\mathcal{SROEL}(\sqcap_s, \times)$ in [Krö10]. This work provides an updated view on the use of datalog for $\mathcal{SROEL}(\sqcap_s, \times)$ reasoning together with more direct proofs of its correctness. It also sheds more light on the difficulty of implementing certain $\mathcal{SROEL}(\sqcap_s, \times)$ features by discussing the space complexity of bottom-up reasoning in (fragments of) $\mathcal{SROEL}(\sqcap_s, \times)$.

Boolean constructors on roles have been investigated in the context of both description and modal logics. [Bor96] used them extensively for the definition of a DL that is equivalent to the two-variable fragment of **FOL**. Complexity results for various modal logics with Boolean role constructors have been obtained in [LS02], and initial results for related DLs have been derived from this work.

The description logic $\mathcal{ALCNR}$ that extends $\mathcal{ALC}$ with unqualified number restrictions and role conjunctions has been introduced in [BDS93] where it was also shown that standard inference problems for this DL are decidable. The more recent results from [LS02] show that augmenting $\mathcal{ALC}$ with full Boolean role constructors ($\mathcal{ALC}(B)$) leads to NExpTime completeness of the standard reasoning tasks, while restricting to either role negation ($\mathcal{ALC}(\neg)$) [LS02] or role conjunction ($\mathcal{ALC}(\sqcap)$) [Tob01] retains ExpTime completeness. The complexity of

$\mathcal{ALC}(B)$ does not further increase when allowing for inverses, qualified number restrictions, and nominals. This was shown in [Tob01] via a polynomial translation of $\mathcal{ALCIQ}(B)$ into C^2, the two-variable fragment of first-order logic with counting quantifiers, for which reasoning was proven to be NExpTime-complete in [PH05]. Also the recently considered description logic $\mathcal{ALO}(B)$ (a.k.a. $\mathcal{ALBO}$) falls in that range of NExpTime-complete DLs [ST07].

In contrast, it was also shown in [Tob01] that restricting to safe Boolean role expressions keeps $\mathcal{ALC}$'s reasoning complexity in ExpTime, even when adding inverses and qualified number restrictions ($\mathcal{ALCIQ}(b)$).

For logics including modelling constructs that deal with role concatenation such as transitivity or, more generally, complex role inclusion axioms, results on complexities in the presence of Boolean role constructors are more sparse. [LW05] shows that $\mathcal{ALC}$ can be extended by negation and regular expressions on roles while keeping reasoning within ExpTime. Furthermore, [CEO07] provided ExpTime complexity for a similar logic that includes inverses and qualified number restriction but reverts to safe negation on roles.

The extension of $\mathcal{SHIQ}$ with non-simple role conjunctions has been introduced under the label $\mathcal{SHIQ}^{\sqcap}$ in [GLHS08] in the context of conjunctive query answering, and the results of that work imply an upper bound of 2ExpTime. In [GK08], it was shown that this upper bound is tight, and that the extension of $\mathcal{SHOIF}$ with non-simple role conjunctions is even N2ExpTime-hard. We point out that the support of *arbitrary* non-simple role conjunctions in these works cannot be extended to DLs with complex role inclusion axioms as this would immediately lead to undecidability.

Chapter 6

Horn Logic Fragments of Description Logics

In first-order logic, *Horn clauses* are disjunctions of atomic formulae and negated atomic formulae that contain at most one non-negated formula. Many kinds of rules in logic programming, and especially datalog rules, thus correspond to Horn clauses. In terms of datalog, the restriction to Horn clauses disallows disjunctions in the head of rules, and thus allow for deterministic evaluation strategies. This simplification is also visible in terms of computational complexities: inferencing in datalog is ExpTime-complete w.r.t. the size of the program, while it is NExpTime-complete in disjunctive datalog. Similar differences are found when considering *data complexity*, the complexity of inferencing w.r.t. the number of ground facts of the program, which increases from P to NP when adding disjunctions.

As illustrated in Section 5.4, reasoning in description logics can be possible by reducing inference problems for a given DL knowledge base to inference problems for a corresponding datalog program. A number of further reductions to datalog have been proposed for various description logics, see Section 8.7 for an overview. A common aspect of many of these approaches is that ABox axioms that do not contain complex concept expressions can directly be rewritten to datalog facts.

This has motivated the study of cases where datalog reductions result in non-disjunctive datalog programs, i.e. Horn clauses, and the corresponding description logics have been dubbed *Horn description logics* accordingly. The first and most prominent such DL was Horn-$\mathcal{SHIQ}$, which was obtained naturally from the KAON2 system [MS06], but other well-known DLs such as $\mathcal{EL}^{++}$ also share characteristics of Horn DLs. Due to the direct rewriting of ABox facts, Horn description logics necessarily allow standard inference tasks to be solved in polynomial time w.r.t. the size of the ABox axioms that contain no complex concepts, a measure that is known as the data complexity of a DL. It turned out that this useful property of Horn DLs can also be exploited in inferencing algorithms that do not rely on reductions to datalog.

In this chapter, we generalise the definition of Horn-$\mathcal{SHIQ}$ to arbitrary DLs that are fragments of $\mathcal{SROIQ}$, and we provide a comprehensive analysis of the worst-case complexities of the resulting logics. While low data complexity is a characteristic (and well-known) feature of Horn DLs, our results show that the complexity of inferencing w.r.t. the overall size of the knowledge base is not necessarily lower in the Horn case. However, we are able to identify restricted DLs for which inferencing is significantly harder than for their Horn versions.

Our observations also highlight the close connections of Horn DLs to *Description Logic Programs* that have been proposed as an "intersection" of Horn and description logic. Although we will see in Chapter 7 that this vague characterisation is hardly adequate to describe the complex relationship between DL and datalog, basic DLP languages are interesting simple formalisms that allow for straightforward rule-based implementations. This was one of the central motivations for the definition of the OWL 2 RL ontology language which we can also relate to a suitable Horn DL below.

We begin this chapter in Section 6.1 by defining Horn-$\mathcal{SROIQ}^{\text{free}}$ as a large Horn DL that provides the framework for defining the more specific logics that are considered herein. Increasingly expressive fragments of Horn-$\mathcal{SROIQ}^{\text{free}}$ are studied in subsequent sections. Section 6.2 introduces the tractable Horn-$\mathcal{FL}_0$, Section 6.3 shows reasoning for all DLs between Horn-$\mathcal{FL}^-$ and Horn-$\mathcal{FLOH}^-$ to be PSpace-complete, and Section 6.4 establishes ExpTime-completeness for all DLs between Horn-$\mathcal{FLE}$ and Horn-$\mathcal{SHIQ}$. The results are discussed in Section 6.5 and an overview of related work is provided in Section 6.6.

The results of this chapter have also been published in [KRH07a, KRH06, KHVS06].

6.1 A Horn Fragment of $\mathcal{SROIQ}$

We first provide a direct definition of a Horn fragment of $\mathcal{SROIQ}^{\text{free}}$ that will be the basis for the various Horn DLs studied herein. Our definition is motivated by the DL Horn-$\mathcal{SHIQ}$, and we will show below that it is indeed a generalisation of the original definition of this logic [HMS05].

Definition 6.1.1 A Horn-$\mathcal{SROIQ}^{\text{free}}$ knowledge base over a DL signature $\mathscr{S}$ is a set of $\mathcal{SROIQ}^{\text{free}}$ axioms which are

– $\mathcal{SROIQ}^{\text{free}}$ RBox axioms over $\mathscr{S}$, or

$$\mathbf{C_1} ::= \mathbf{C_0} \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid {\leqslant}0\,\mathbf{R}.\neg\mathbf{C_1} \mid {\leqslant}1\,\mathbf{R}.\neg\mathbf{C_0} \mid {\geqslant}n\,\mathbf{R}.\mathbf{C_1} \mid \mathbf{C_1} \sqcap \mathbf{C_1} \mid \mathbf{C_1} \sqcup \mathbf{C_0}$$
$$\mathbf{C_0} ::= \top \mid \bot \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid {\leqslant}0\,\mathbf{R}.\neg\mathbf{C_0} \mid \mathbf{C_0} \sqcap \mathbf{C_0} \mid \mathbf{C_0} \sqcup \mathbf{C_0}$$

Figure 6.1: Horn-$\mathcal{SROIQ}^{\text{free}}$ concept expressions in positive negation normal form

$C\vert_\epsilon$	$= C$	$\mathsf{pol}(C,\epsilon)$	$= 1$	
$(\neg C)\vert_{1p}$	$= C\vert_p$	$\mathsf{pol}(\neg C, 1p)$	$= -\mathsf{pol}(C,p)$	
$(C_1 \Box C_2)\vert_{ip}$	$= C_i\vert_p$	$\mathsf{pol}(C_1 \Box C_2, ip)$	$= \mathsf{pol}(C_i,p)$	for $\Box \in \{\sqcap,\sqcup\}$, $i \in \{1,2\}$
${\leqslant}n\,R.C\vert_{3p}$	$= C\vert_p$	$\mathsf{pol}({\leqslant}n\,R.C, 3p)$	$= -\mathsf{pol}(C,p)$	
${\geqslant}n\,R.C\vert_{3p}$	$= C\vert_p$	$\mathsf{pol}({\geqslant}n\,R.C, 3p)$	$= \mathsf{pol}(C,p)$	

Figure 6.2: Positions in a concept (left) and their polarity (right)

- GCIs $C \sqsubseteq D$ over $\mathscr{S}$ such that $\mathsf{pNNF}(\neg C \sqcup D)$ is a $\mathbf{C_1}$ concept as defined in Fig. 6.1, or
- ABox axioms $C(a)$ where the $\mathsf{pNNF}(C)$ is a $\mathbf{C_1}$ concept as defined in Fig. 6.1. ◇

Note that Fig. 6.1 exploits some syntactic simplifications as discussed in Section 3.1.3, and in particular that existential and universal restrictions are not mentioned explicitly. When convenient, we will still use this notation when considering fragments on Horn-$\mathcal{SROIQ}^{\text{free}}$ below.

The original definition of Horn-$\mathcal{SHIQ}$ in [HMS05] is rather more complex than the above characterisation, using a recursive function that counts the positive literals that would be needed when decomposing an axiom into equisatisfiable formulae in disjunctive normal forms. In order to show that our definition leads to the same results, we first recall the definition from [HMS05] which requires us to introduce some auxiliary concepts first.

Subconcepts of some description logic concept are denoted by specifying their *position*. Formally, a position p is a finite sequence of natural numbers, where ϵ denotes the empty position. Given a concept C, $C|_p$ denotes the *subconcept* of C at position p, defined recursively as in Fig. 6.2 (left). In this paper, we consider only positions that are defined in this figure, and the set of *all positions in a concept* C is understood accordingly. Given a concept C and a position p in C, the *polarity* $\mathsf{pol}(C,p)$ of C at position p is defined as in Fig. 6.2 (right). Using this notation, we can state the following definition of Horn knowledge bases.

Definition 6.1.2 Let pl^+ and pl^- denote mutually recursive functions that map a $\mathcal{SHIQ}$ concept D to a non-negative integer as specified in Fig. 6.3 where

D	$\mathsf{pl}^+(D)$	$\mathsf{pl}^-(D)$
$\bot$	0	0
$\top$	0	0
A	1	0
$\neg C$	$\mathsf{pl}^-(C)$	$\mathsf{pl}^+(C)$
$\bigsqcap C_i$	$\max_i \mathsf{sgn}(\mathsf{pl}^+(C_i))$	$\sum_i \mathsf{sgn}(\mathsf{pl}^-(C_i))$
$\bigsqcup C_i$	$\sum_i \mathsf{sgn}(\mathsf{pl}^+(C_i))$	$\max_i \mathsf{sgn}(\mathsf{pl}^-(C_i))$
$\geqslant n\, R.C$	1	$\frac{n(n-1)}{2} + n\ \mathsf{sgn}(\mathsf{pl}^-(C))$
$\leqslant n\, R.C$	$\frac{n(n+1)}{2} + (n+1)\ \mathsf{sgn}(\mathsf{pl}^-(C))$	1

Figure 6.3: Definition of $\mathsf{pl}^+(D)$ and $\mathsf{pl}^-(D)$

$\mathsf{sgn}(0) = 0$ and $\mathsf{sgn}(n) = 1$ for $n > 0$. We define a function pl that assigns to each $\mathcal{SHIQ}$ concept C and position p in C a non-negative integer by setting:

$$\mathsf{pl}(C,p) = \begin{cases} \mathsf{pl}^+(C|_p) & \text{if } \mathsf{pol}(D,p) = 1, \\ \mathsf{pl}^-(C|_p) & \text{if } \mathsf{pol}(D,p) = -1, \end{cases}$$

A concept C is *Horn* if $\mathsf{pl}(C,p) \leq 1$ for every position p in C, including the empty position ϵ. A $\mathcal{SHIQ}$ knowledge base KB is *Horn* if $\neg C \sqcup D$ is Horn for each GCI $C \sqsubseteq D$ of KB, and C is Horn for each assertion $C(a)$ of KB. $\Diamond$

The corresponding Definition 1 in [HMS05] refers to $\mathcal{ALCHIQ}$ instead of $\mathcal{SHIQ}$ since an elimination procedure for transitive roles that is considered within this work may introduce axioms that are not Horn in the above sense. However, it turns out that transitive roles – and $\mathcal{SROIQ}$ role chains in general – can also be eliminated without endangering the Hornness of a knowledge base. An according transformation that follows [Kaz08] is reviewed in Section 9.3. Hence we can safely extend the definition to $\mathcal{SHIQ}$.

While suitable as a criterion for *checking* Hornness of single axioms or knowledge bases, this Definition 6.1.2 is not particularly suggestive as a description of the class of Horn knowledge bases as a whole. Indeed, it is not readily seen for which formulae pl yields values smaller or equal to 1 for all possible positions in the formula. Moreover, Definition 6.1.2 is still overly detailed as pl calculates the *exact* number of positive literals being introduced when transforming some (sub)formula.

To show that Definition 6.1.1 is a suitable generalisation of Definition 6.1.2, we first observe that Hornness is not affected by transformation to positive negation normal form.

Lemma 6.1.3 *A $\mathcal{SHIQ}$ concept C is Horn according to Definition 6.1.2 iff its positive negation normal form* $\mathsf{pNNF}(C)$ *is Horn according to this definition.*

Proof. The result is shown by establishing that the steps of the normal form transformation in Fig. 3.3 do not affect the value of pl^+. The same could be shown for pl^- but this part can be omitted by noting that the concepts that are transformed in the recursive definition of pNNF are always in positive positions. The claim clearly holds if C is a concept name, $\top$, or $\bot$. Consider the case that $C = \neg(D_1 \sqcap D_2)$. Then $\mathsf{pl}^+(C) = \mathsf{sgn}(\mathsf{pl}^-(D_1)) + \mathsf{sgn}(\mathsf{pl}^-(D_2)) = \mathsf{sgn}(\mathsf{pl}^+(\neg D_1)) + \mathsf{sgn}(\mathsf{pl}^+(\neg D_2))$. By the induction hypothesis this equals $\mathsf{sgn}(\mathsf{pl}^+(\mathsf{pNNF}(\neg D_1))) + \mathsf{sgn}(\mathsf{pl}^+(\mathsf{pNNF}(\neg D_2))) = \mathsf{pl}^+(\mathsf{pNNF}(\neg(D_1 \sqcap D_2)))$, as required. The other cases of the induction are similar. □

Proposition 6.1.4 *A $\mathcal{SHIQ}$ concept C is Horn according to Definition 6.1.2 iff it is Horn according to Definition 6.1.1.*

Proof. "$\Leftarrow$" We need show that $\mathsf{pNNF}(D) \in \mathbf{C_1}$ ($\mathsf{pNNF}(D) \in \mathbf{C_0}$) implies $\mathsf{pl}^+(D) \leq 1$ ($\mathsf{pl}^+(D) = 0$). Focussing on pl^+ suffices since subconcepts that occur with negative polarity within a concept in positive negation normal form are either atomic or of the form $\neg D'$ with $D' \in \mathbf{C_1}$. By Lemma 6.1.3, it suffices to show that $D \in \mathbf{C_1}$ ($D \in \mathbf{C_0}$) implies $\mathsf{pl}^+(D) \leq 1$ ($\mathsf{pl}^+(D) = 0$). This can be established with some easy inductions over the structure of $\mathbf{C_0}$ and $\mathbf{C_1}$, where all cases follow by straightforward calculation of pl^+, applying the induction hypothesis to obtain results for subexpressions.

"$\Rightarrow$" By Lemma 6.1.3, we can again restrict attention to concepts in positive negation normal form. We first show that, whenever D in pNNF is such that $\mathsf{pl}^+(D) = 0$, we find that $D \in \mathbf{C_0}$. The contrapositive – if $D \notin \mathbf{C_0}$ then $\mathsf{pl}^+(D) \neq 0$ – can be shown by induction over the structure of D. The result is immediate for $D \in \mathbf{A}$, and follows by simple calculation in all other cases. As an example, consider $D = \leqslant n\, R.\neg D'$. If $n > 0$, then $\mathsf{pl}^+(D) \leq 1$ is immediate. If $n = 0$ then $D' \notin \mathbf{C_0}$ and $\mathsf{pl}^+(D') = \mathsf{sgn}(\mathsf{pl}^+(D'))$, where the later is 1 by the induction hypothesis.

To establish the claim, we can now show that, whenever D in pNNF is such that $\mathsf{pl}^+(D) \leq 1$, we find that $D \in \mathbf{C_1}$. The required induction is similar to the $\mathbf{C_0}$ case, so we omit the details. □

The previous result shows that Definition 6.1.1 is indeed a generalisation of the original definition of Horn-$\mathcal{SHIQ}$. The extension with nominals and Self expressions may appear natural, but it remains to be shown that it actually leads to appropriate results. We will not study Horn-$\mathcal{SROIQ}^{\text{free}}$ as such in the following sections, but we will rather consider various fragments of this logic. Recall the following definitions of subboolean description logics from [BCM+07]:

Definition 6.1.5 Consider a $\mathcal{SROIQ}^{\text{free}}$ concept expression C.

– C is an $\mathcal{FLE}$ concept if it uses only only the constructors $\top$, $\bot$, $\sqcap$, $\exists$, and $\forall$.

- C is an $\mathcal{FL}^-$ concept if it is an $\mathcal{FLE}$ concept and all of its existential role restrictions have the form $\exists R.\top$.
- C is an $\mathcal{FL}_0$ concept if it is an $\mathcal{FL}^-$ concept that does not contain existential role restrictions.

The description logics $\mathcal{FLE}$, $\mathcal{FL}^-$, and $\mathcal{FL}_0$ allow for arbitrary GCIs and concept assertions that contain only concept expressions of the respective type. RBox axioms are not supported. $\diamond$

When defining the Horn variant of each of these description logics, it is relevant whether GCIs or globally valid concept expressions are considered when applying the syntactic restrictions. For example, the GCI $A \sqcap B \sqsubseteq C$ is in $\mathcal{FL}_0$ but the corresponding universally valid concept expression $\neg(A \sqcap B) \sqcup C$ and its pNNF $\neg A \sqcup \neg B \sqcup C$ are not. Disjunction could be included to overcome this issue – the Hornness conditions restrict its expressive power as done in Horn-$\mathcal{SHIQ}$ – but then concepts such as $\forall R.\neg A \sqcup \forall S.B$ would be expressible, whereas the corresponding GCI $\exists R.A \sqsubseteq \forall S.B$ cannot be expressed in $\mathcal{FL}_0$. Therefore, we apply restrictions on the level of GCIs and do not include concept unions, thus ensuring that all Horn-$\mathcal{FL}_0$ knowledge bases are also expressible in $\mathcal{FL}_0$. Note that the normal form transformations that were used in Definition 6.1.1 are not affected by such considerations, since Horn restrictions are invariant under negation normal form transformations as illustrated in Lemma 6.1.3.

Definition 6.1.6 The description logic Horn-$\mathcal{FLE}$ (Horn-$\mathcal{FL}^-$, Horn-$\mathcal{FL}_0$) allows for the following axioms:

- GCIs $C \sqsubseteq D$ such that the concepts C, D are in $\mathcal{FLE}$ ($\mathcal{FL}^-$, $\mathcal{FL}_0$) and we find that $\mathsf{pNNF}(\neg C \sqcup D) \in \mathbf{C_1}$, or
- concept assertions $C(a)$ such that the concept C is in $\mathcal{FLE}$ ($\mathcal{FL}^-$, $\mathcal{FL}_0$) and $\mathsf{pNNF}(C) \in \mathbf{C_1}$,

where $\mathbf{C_1}$ is defined as in Fig. 6.1. $\diamond$

These basic Horn DLs form the basis of our subsequent investigations, and it will turn out that they have very different computational properties in spite of the rather similar syntax. We will also extend the previously defined Horn DLs to include further features of Horn-$\mathcal{SROIQ}^{\text{free}}$ that are not included yet. For example, we will consider the logic Horn-$\mathcal{FLOH}^-$ that extends Horn-$\mathcal{FL}^-$ with nominals and role hierarchies.

6.2 A Light-Weight Horn-DL: Horn-$\mathcal{FL}_0$

The description logic $\mathcal{FL}_0$ is indeed very simple: $\top$, $\bot$, $\sqcap$, and $\forall$ are the only operators allowed. Yet, checking the satisfiability of $\mathcal{FL}_0$ knowledge bases is already ExpTime-complete [BBL05]. It is not hard to see that Horn-$\mathcal{FL}_0$ is in P, and thus is much simpler than its non-Horn counterpart.

Proposition 6.2.1 *All standard reasoning problems for Horn-$\mathcal{FL}_0$ are* P*-complete.*

Proof. An axiom of Horn-$\mathcal{FL}_0$ is in normal form if it is of one of the following forms: $A \sqsubseteq C$, $A \sqcap B \sqsubseteq C$, $A \sqsubseteq \bot$, $\top \sqsubseteq C$, $A \sqsubseteq \forall R.C$, $C(a)$, $R(a, b)$, where A, B, C are concept names, R is a role name, and a, b are individual names. Now it is easy to see that every Horn-$\mathcal{FL}_0$ knowledge base KB is semantically emulated by a Horn-$\mathcal{FL}_0$ knowledge base in normal form that can be computed in linear time w.r.t the size of KB. An according normal form transformation is detailed for Horn-$\mathcal{FLOH}^-$ in Lemma 6.3.6, and the transformation for Horn-$\mathcal{FL}_0$ is an easy special case thereof, with the only difference that GCIs $\{a\} \sqsubseteq C$ must be written as $C(a)$ in Horn-$\mathcal{FL}_0$.

It is easy to see that every Horn-$\mathcal{FL}_0$ knowledge base in normal form can be translated to a semantically equivalent datalog program. Indeed, this translation is obtained by applying the standard transformation of $\mathcal{SROIQ}$ axioms to first-order logic with equality as described in Section 3.2. Since all of the rules that are obtained by translating normal form axioms have at most three variables, the result follows from fact that satisfiability checking is P-complete for datalog programs with a bounded number of variables per rule (Fact 4.1.4). Moreover, we also note that the reductions of standard reasoning problems to satisfiability checking (Proposition 3.1.9) are possible in Horn-$\mathcal{FL}_0$ as well. □

It is easy to see that this simple result could be established even when extending Horn-$\mathcal{FL}_0$ with further expressive features. In particular, this is the case for all features of $\mathcal{SROIQ}$ for which the first-order translation of Section 3.2 would lead to datalog axioms, possibly with equality as discussed in Section 4.1.3. This includes nominals, inverse roles, role chains, local reflexivity (Self), and the universal role, where the normalisation of role chains could be established as in Proposition 5.4.3. Moreover, role conjunctions and concept products as discussed in Chapter 5 are easily integrated into this setting as well, even without restricting to simple roles.

Description logics that can be expressed in – or rather can be semantically emulated in – datalog have been called *Description Logic Programs* (DLP). The observations of the previous paragraph show that Horn-$\mathcal{FLSROI}_0(\sqcap)$ is a DLP language in this sense, but the literature on DLP also considers cases where a particular combination of constructs enables the translation of a DL axiom into datalog. For example, the axiom $A \sqsubseteq \exists R.\{a\}$ can be expressed as $A(x) \to R(x, a)$ although existential restrictions are not generally supported in datalog. This shows that our above observations do not yet lead to a largest possible DLP, and it raises the question of whether and how a maximal DLP language can be found. Answers to these questions are given in Chapter 7.

Two additional features – disjunction and qualified functionality restrictions $\leqslant 1\, R.C$ – are of interest for us to obtain a Horn DL that is more closely related to the OWL RL profile [MCH⁺09]. Considering Definition 6.1.1 and Fig. 6.1, we observe that Horn DLs allow for at most one $\mathbf{C_1}$ concept in each disjunction. Every GCI that is Horn in this sense can therefore be expressed in a form where said $\mathbf{C_1}$ concept constitutes the right-hand side of the concept inclusion axiom, while all other disjunctions occur on the left-hand side. Such disjunctions on the left-hand side of GCIs, however, can easily be eliminated during normal form transformation since $A \sqcup B \sqsubseteq C$ is equivalent to $\{A \sqsubseteq C, A \sqsubseteq C\}$. Therefore, the addition of Horn-disjunction does not increase the expressiveness of the DL. Disjunctions in subboolean DLs have traditionally been denoted by the letter $\mathcal{U}$, and hence we extend our results to an extension of Horn-$\mathcal{FLU}_0$.

Qualified functionality restrictions in turn are only allowed in $\mathbf{C_1}$ expressions of the form $\leqslant 1\, R.\neg C$ with $C \in \mathbf{C_0}$. Such expressions can be simplified by replacing $\neg C$ with a fresh concept name A while introducing a new axiom $\neg C \sqsubseteq A$ (this is Horn since $C \in \mathbf{C_0}$). In addition, it is easy to see that axioms of the form $B \sqsubseteq \leqslant 1\, R.A$ are translated to datalog rules $B(x) \wedge R(x, y_1) \wedge A(y_1) \wedge R(x, y_2) \wedge A(y_2) \to y_1 \approx y_2$, so they can indeed be included into an extension of Horn-$\mathcal{FL}_0$. Summing up the above discussion, we obtain the following result:

Proposition 6.2.2 *Let Horn-$\mathcal{SROIQ}(\sqcap)^{free}$ be the extension of Horn-$\mathcal{SROIQ}^{free}$ with arbitrary conjunctions of roles, and let $\mathcal{RL}$ denote the fragment of Horn-$\mathcal{SROIQ}(\sqcap)^{free}$ comprising all knowledge bases that contain no maximality restrictions for numbers other than* 1, *no existential restrictions, and no minimality restrictions. The standard reasoning problems for $\mathcal{RL}$ are* P*-complete.*

Proof. It has been sketched in the above discussion how to extend the normal form transformation of Lemma 6.3.6 to cover Horn disjunction of concepts and qualified functionality restrictions on the right-hand side of GCIs. A suitable normal form for GCIs is defined by requiring all left-hand sides to be of the forms $\top$, A or $A \sqcap B$, and all right-hand sides to be of the form $\bot$, A, $\forall R.A$, or $\leqslant 1\, R.A$, where A and B are concept names, nominals, or expressions $\exists S.\mathsf{Self}$, and where R, S are role names. A normal form of RIAs allows only axioms of the form $R \sqsubseteq T$, $R \circ S \sqsubseteq T$, and $R \sqcap S \sqsubseteq T$, where R, S, T are role names, inverses of role names, or the universal role. Clearly, any $\mathcal{RL}$ knowledge base is semantically emulated by an $\mathcal{RL}$ knowledge base in normal form that can be computed in polynomial time

– in fact, since all transformations can be accomplished in a single pass, it is even possible to achieve the normalisation in LOGSPACE.

A polynomial-time inferencing algorithm is obtained by further translating normalised $\mathcal{RL}$ knowledge bases into datalog programs with a bounded number of variables per rule, as in Proposition 6.2.1. □

In terms of the nomenclature that was introduced for DLs in earlier chapters, $\mathcal{RL}$ could also be described by the more explicit but less readable name Horn-$\mathcal{FLUSROIF}_0(\sqcap)$, although this would still neglect qualified functionality restrictions since $\mathcal{F}$ usually denotes only unqualified functionality restrictions of the form $\leqslant 1\, R.\top$. The reason why the rather exotic description logic $\mathcal{RL}$ is specifically mentioned here is that it includes essentially all features of the OWL 2 RL ontology language which are not related to datatypes [MCH+09]. Adding datatypes is no major difficulty but requires extended preliminary discussions that are beyond the scope of this work.

The only syntactic feature of OWL RL that $\mathcal{RL}$ is missing are existential quantifiers on the left-hand side of GCIs which do not increase expressiveness but which syntactically extend OWL RL. Horn DLs do not restrict the use of existentials, so introducing them to $\mathcal{RL}$ would require additional constraints that do not fit well into the framework of Horn DLs. In contrast, restrictions on the use of existentials appear naturally when studying DLP in Chapter 7. This indicates that Horn DLs are based on first-order Horn logic with functions, while DLP refers to the function-free fragment datalog. Overall, $\mathcal{RL}$ still establishes a close relationship of OWL 2 RL with the formalisms considered within this work, especially with Horn description logics and DLP.

6.3 PSPACE-Complete Horn DLs: From Horn-$\mathcal{FL}^{-}$ to Horn-$\mathcal{FLOH}^{-}$

Horn-$\mathcal{FL}^{-}$ is the Horn fragment of $\mathcal{ALC}$ that allows $\top$, $\bot$, $\sqcap$, $\forall$, and unqualified $\exists$, i.e. concept expressions of the form $\exists R.\top$. Although Horn-$\mathcal{FL}^{-}$ is only a very small extension of Horn-$\mathcal{FL}_0$, we will see that it is PSPACE-complete. Moreover, not all of the extensions that could be added to Horn-$\mathcal{FL}_0$ can also be added to Horn-$\mathcal{FL}^{-}$ without further increasing the complexity. The extension of $\mathcal{FL}^{-}$ that we will consider below is defined as follows.

Definition 6.3.1 The description logic $\mathcal{FLOH}^{-}$ is the extension of $\mathcal{FL}^{-}$ with nominals, and role hierarchies. The logic Horn-$\mathcal{FLOH}^{-}$ is the restriction of $\mathcal{FLOH}^{-}$ that contains only GCIs $C \sqsubseteq D$ and concept assertions $E(a)$ such that $\mathsf{pNNF}(\neg C \sqcup D) \in \mathbf{C}_1$ and $\mathsf{pNNF}(E) \in \mathbf{C}_1$. ◇

In the following sections, we show that all logics between Horn-$\mathcal{FL}^-$ and Horn-$\mathcal{FLOH}^-$ are PSpace-complete.

6.3.1 Hardness

We directly show that Horn-$\mathcal{FL}^-$ is PSpace-hard by reducing the halting problem for polynomially space-bounded Turing machines to checking unsatisfiability in Horn-$\mathcal{FL}^-$.

Definition 6.3.2 A *deterministic Turing machine* (TM) $\mathcal{M}$ is a tuple (Q, Σ, Δ, q_0) where

- Q is a finite set of states,
- Σ is a finite *alphabet* that includes a *blank* symbol $\square$,
- $\Delta \subseteq (Q \times \Sigma) \times (Q \times \Sigma \times \{l, r\})$ is a *transition relation* that is deterministic, i.e. $(q, \sigma, q_1, \sigma_1, d_1), (q, \sigma, q_2, \sigma_2, d_2) \in \Delta$ implies $q_1 = q_2$, $\sigma_1 = \sigma_2$, and $d_1 = d_2$.
- $q_0 \in Q$ is the *initial state*, and
- $Q_A \subseteq Q$ is a set of *accepting states.*

A *configuration* of $\mathcal{M}$ is a word $\alpha \in \Sigma^* Q \Sigma^*$. A configuration α' is a *successor* of a configuration α if one of the following holds:

1. $\alpha = w_l q \sigma \sigma_r w_r$, $\alpha' = w_l \sigma' q' \sigma_r w_r$, and $(q, \sigma, q', \sigma', r) \in \Delta$,
2. $\alpha = w_l q \sigma$, $\alpha' = w_l \sigma' q' \square$, and $(q, \sigma, q', \sigma', r) \in \Delta$,
3. $\alpha = w_l \sigma_l q \sigma w_r$, $\alpha' = w_l q' \sigma_l \sigma' w_r$, and $(q, \sigma, q', \sigma', l) \in \Delta$,

where $q \in Q$ and $\sigma, \sigma', \sigma_l, \sigma_r \in \Sigma$ as well as $w_l, w_r \in \Sigma^*$. Given some natural number s, the possible *transitions in space s* are defined by additionally requiring that $|\alpha'| \leq s + 1$.

The set of *accepting configurations* is the least set which satisfies the following conditions. A configuration α is accepting iff

- $\alpha = w_l q w_r$ and $q \in Q_A$, or
- at least one the successor configurations of α are accepting.

$\mathcal{M}$ *accepts* a given word $w \in \Sigma^*$ (in space s) iff the configuration $q_0 w$ is accepting (when restricting to transitions in space s). ◇

The complexity class PSpace is defined as follows.

(1) Left and right transition rules:

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq \exists S.\top \sqcap \forall S.(A_{q'} \sqcap H_{i+1} \sqcap C_{\sigma',i})$ with $\delta = (q,\sigma,q',\sigma',r)$, $i < p(|w|) - 1$

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq \exists S.\top \sqcap \forall S.(A_{q'} \sqcap H_{i-1} \sqcap C_{\sigma',i})$ with $\delta = (q,\sigma,q',\sigma',l)$, $i > 0$

(2) Memory:

$H_j \sqcap C_{\sigma,i} \sqsubseteq \forall S.C_{\sigma,i}$ $\quad i \neq j$

(3) Failure:

$F \sqcap A_q \sqsubseteq \bot$ $\quad q \in Q_A$

(4) Propagation of failure:

$F \sqsubseteq \forall S.F$

The axioms are instantiated for all $q, q' \in Q$, $\sigma, \sigma' \in \Sigma$, $i, j \in \{0, \ldots, p(|w|) - 1\}$, and $\delta \in \Delta$.

Figure 6.4: Knowledge base $\mathrm{KB}_{\mathcal{M},w}$ simulating a polynomially space-bounded TM

Definition 6.3.3 A language L is accepted by a polynomially space-bounded TM iff there is a polynomial p such that, for every word $w \in \Sigma^*$, $w \in L$ iff w is accepted in space $p(|w|)$. ◇

In this section, we exclusively deal with polynomially space-bounded TMs, and so we omit additions such as "in space s" when clear from the context.

In the following, we consider a fixed TM $\mathcal{M}$ denoted as in Definition 6.3.2, and a polynomial p that defines a bound for the required space. For any word $w \in \Sigma^*$, we construct a Horn-$\mathcal{FL}^-$ knowledge base $\mathrm{KB}_{\mathcal{M},w}$ and show that w is accepted by $\mathcal{M}$ iff $\mathrm{KB}_{\mathcal{M},w}$ is unsatisfiable. Intuitively, the elements of an interpretation domain of $\mathrm{KB}_{\mathcal{M},w}$ represent possible configurations of $\mathcal{M}$, encoded by the following concept names

- A_q for $q \in Q$: the TM is in state q,
- H_i for $i = 0, \ldots, p(|w|) - 1$: the TM is at position i on the storage tape,
- $C_{\sigma,i}$ with $\sigma \in \Sigma$ and $i = 0, \ldots, p(|w|) - 1$: position i on the storage tape contains symbol σ.

Based on these concepts, elements in each interpretation of a knowledge base encode certain states of the Turing machine. A role S will be used to encode the successor relationship between states. The initial configuration for word w is described by the concept expression I_w:

$$I_w := A_{q_0} \sqcap H_0 \sqcap C_{\sigma_0,0} \sqcap \ldots \sqcap C_{\sigma_{|w|-1},|w|-1} \sqcap C_{\Box,|w|} \sqcap \ldots \sqcap C_{\Box,p(|w|)-1},$$

where σ_i denotes the symbol at the ith position of w.

It is not hard to describe runs of the TM with Horn-$\mathcal{FL}^-$ axioms, but formulating the acceptance condition is slightly more difficult. The reason is that in absence of statements like $\exists S.A$ and $\forall S.A$ in the condition part of Horn-axioms, one cannot propagate acceptance from the final accepting configuration back to initial configuration. The solution is to introduce a new concept F that states that a state is *not* accepting, and to propagate this assumption forwards along the runs to provoke an inconsistency as soon as an accepting configuration is reached. Thus we arrive at the axioms given in Fig. 6.4.

Next we need to investigate the relationship between elements of an interpretation that satisfies $\mathrm{KB}_{\mathcal{M},w}$ and configurations of $\mathcal{M}$. Given an interpretation $\mathcal{I}$ of $\mathrm{KB}_{\mathcal{M},w}$, we say that an element e of the domain of $\mathcal{I}$ *represents* a configuration $\sigma_1 \ldots \sigma_{i-1} q \sigma_i \ldots \sigma_m$ if $e \in A_q^{\mathcal{I}}$, $e \in H_i^{\mathcal{I}}$, and, for every $j \in \{0, \ldots, p(|w|) - 1\}$, $e \in C_{\sigma,j}^{\mathcal{I}}$ whenever

$$j \leq m \text{ and } \sigma = \sigma_m \qquad \text{or} \qquad j > m \text{ and } \sigma = \Box.$$

Note that we do not require uniqueness of the above, so that a single element might in fact represent more than one configuration. As we will see below, this does not affect our results. If e represents a configuration as above, we will also say that e has state q, position i, symbol σ_j at position j etc.

Lemma 6.3.4 *Consider some interpretation $\mathcal{I}$ that satisfies* $\mathrm{KB}_{\mathcal{M},w}$. *If some element e of $\mathcal{I}$ represents a configuration α and some transition δ is applicable to α, then e has an $S^{\mathcal{I}}$-successor that represents the (unique) result of applying δ to α.*

Proof. Consider an element e, state α, and transition δ as in the claim. Then one of the axioms (1) applies, and e must also have an $S^{\mathcal{I}}$-successor. This successor represents the correct state, position, and symbol at position i of e, again by the axioms (1). By axiom (2), symbols at all other positions are also represented by all $S^{\mathcal{I}}$-successors of e. □

Lemma 6.3.5 *A word w is accepted by $\mathcal{M}$ iff $\{I_w(i), F(i)\} \cup \mathrm{KB}_{\mathcal{M},w}$ is unsatisfiable, where i is a new constant symbol.*

Proof. Let $\mathcal{I}$ be a model of $\{I_w(i), F(i)\} \cup \mathrm{KB}_{\mathcal{M},w}$. $\mathcal{I}$ being a model for $I_w(i)$, $i^{\mathcal{I}}$ clearly represents the initial configuration of $\mathcal{M}$ with input w. By Lemma 6.3.4, for any further configuration reached by $\mathcal{M}$ during computation, $i^{\mathcal{I}}$ has a (not necessarily direct) $S^{\mathcal{I}}$ successor representing that configuration.

Since $\mathcal{I}$ satisfies $F(i)$ and axiom (4) of Fig. 6.4, a simple induction argument shows that F^I contains all $S^{\mathcal{I}}$ successors of $i^{\mathcal{I}}$. But then $\mathcal{I}$ satisfies axiom (3) only if none of the configurations that are reached have an accepting state. Since $\mathcal{I}$ was arbitrary, $\{I_w(i), F(i)\} \cup \mathrm{KB}_{\mathcal{M},w}$ can only have a satisfying interpretation if $\mathcal{M}$ does not reach an accepting state.

$A \sqsubseteq C$	$\top \sqsubseteq C$	$A \sqsubseteq \forall R.C$	$R \sqsubseteq S$
$A \sqcap B \sqsubseteq C$	$A \sqsubseteq \bot$		$R(c,d)$

Figure 6.5: Normal forms for Horn-$\mathcal{FLOH}^-$ with $A, B, C \in \mathbf{B}$ basic concepts (including existential restrictions), R, S role names, and c, d individual names

It remains to show the converse: if $\mathcal{M}$ does not accept w, there is some interpretation $\mathcal{I}$ satisfying $\{I_w(i), F(i)\} \cup \mathrm{KB}_{\mathcal{M},w}$. To this end, we define a canonical interpretation M as follows. The domain of M is the set of all configurations of $\mathcal{M}$ that have size $p(|w|) + 1$ (i.e. that encode a tape of length $p(|w|)$, possibly with trailing blanks). The interpretations for the concepts A_q, H_i, and $C_{\sigma,i}$ are defined as expected so that every configuration represents itself but no other configuration. Especially, I_w^M is the singleton set containing the initial configuration. Given two configurations α and α', and a transition δ, we define $(\alpha, \alpha') \in S^M$ iff there is a transition δ from α to α'. F^M is defined to be the set of all configurations that are reached during the run of $\mathcal{M}$ on w.

It is easy to see that M satisfies the axioms (1), (2), and (3) of Fig. 6.4. Axiom (4) is satisfied since, by our initial assumption, none of the configurations reached by $\mathcal{M}$ is in an accepting state. □

Thus checking satisfiability of Horn-$\mathcal{FL}^-$ knowledge bases is PSpace-hard.

6.3.2 Containment

To show that inferencing for Horn-$\mathcal{FLOH}^-$ is in PSpace, we develop a tableau algorithm for deciding the satisfiability of a Horn-$\mathcal{FLOH}^-$ knowledge base. To this end, we first present a normal form transformation that allows us to restrict attention to simple forms of axioms. Afterwards, we present the tableau construction and show its correctness, and demonstrate that it can be executed in polynomial space.

To simplify notation, we define a $\mathcal{FLOH}^-$ concept expression C to be *basic* if it is of the form $A \in \mathbf{A}$, $\{a\}$, or $\exists R.\top$. The set of all basic concepts is denoted by $\mathbf{B}$, assuming that the underlying signature is irrelevant or clear from the context.

Lemma 6.3.6 *Every Horn-$\mathcal{FLOH}^-$ knowledge base* KB *is semantically emulated by a Horn-$\mathcal{FLOH}^-$ knowledge base that contains only axioms in the normal form given in Fig. 6.5, and that can be computed in linear time with respect to the size of* KB.

Proof. ABox axioms $C(a)$ can clearly be expressed as GCIs $\{a\} \sqsubseteq C$. To semantically emulate arbitrary GCIs, we exhaustively apply the transformation rules in

$$
\begin{array}{rcll}
\hat{C} & \sqsubseteq & \hat{D} & \mapsto \{\hat{C} \sqsubseteq X, X \sqsubseteq \hat{D}\} \\
\bot & \sqsubseteq & C & \mapsto \emptyset \\
C & \sqsubseteq & \top & \mapsto \emptyset \\
\hat{C} \sqcap A & \sqsubseteq & B & \mapsto \{\hat{C} \sqsubseteq X, X \sqcap A \sqsubseteq B\} \\
A & \sqsubseteq & C \sqcap D & \mapsto \{A \sqsubseteq C, A \sqsubseteq D\} \\
A & \sqsubseteq & \forall R.\hat{C} & \mapsto \{A \sqsubseteq \forall R.X, X \sqsubseteq \hat{C}\}
\end{array}
$$

A, B basic concept expressions, $\top$, or $\bot$; X a fresh concept name;
C, D concept expressions; $\hat{C}$, $\hat{D}$ concept expressions that are not basic

Figure 6.6: Normal form transformation for Horn-$\mathcal{FLOH}^-$

Fig. 6.6, where each rule application consists in replacing the axiom on the left-hand side by the axioms on the right-hand side. It is easy to see that the resulting axioms semantically emulate the original axioms for each rule, so the result follows by induction. It is also easy to see that only a linear number of steps are required, where it must be noted that the rule for $A \sqsubseteq C \sqcap D$ is only applicable if A is not a compound term, so that the duplication of A still leads to only a linear increase in size. □

Next, we are going to present a procedure for checking satisfiability of Horn-$\mathcal{FL}^-$ knowledge bases. In the following we assume without loss of generality that the DL signature in consideration has at least one individual name.

Definition 6.3.7 Consider a Horn-$\mathcal{FLOH}^-$ knowledge base KB in normal form, with $\mathbf{B}$ the set of basic concepts, $\mathbf{R}$ the set of roles, and $\mathbf{I}$ the set of individual names. A set of relevant concept expressions is defined by setting

$$\mathsf{cl}(\mathrm{KB}) = \mathbf{B} \cup \{\forall R.C | R \in \mathbf{R}, C \in \mathbf{B}\} \cup \{\top, \bot\}.$$

Given a set I of individual names, a set $\mathcal{T}_I$ of ABox expressions is defined as follows:

$$\mathcal{T}_I := \{C(e) \mid C \in \mathsf{cl}(\mathrm{KB}),\ e \in I\} \cup \{R(e, f) \mid R \in \mathbf{R},\ e, f \in I\}$$

For a set $T \subseteq \mathcal{T}_I$ and individuals $e, f \in I$, we use $T_{e \mapsto f}$ to denote the set

$$\{C(f) \mid C(e) \in T\} \cup \{R(f, g) \mid R(e, g) \in T,\ g \in I\} \cup \{R(g, f) \mid R(g, e) \in T,\ g \in I\}.$$

For the special case that $e = f$, we use the abbreviation $T_e := T_{e \mapsto e}$. A *tableau* for KB is given by a (possibly infinite) set I of individual names, and a set $T \subseteq \mathcal{T}_I$ such that $\mathbf{I} \subseteq I$ and the following conditions hold:

1. if $e \in I$, then $\top(e) \in T$ and, if $e \in \mathbf{I}$, $\{e\} \in T$,

2. if $A(e) \in$ KB ($R(e, f) \in$ KB), then $A(e) \in T$ ($R(e, f) \in T$),
3. if $\{f\}(e) \in T$, then $C(e) \in T$ iff $C(f) \in T$, $R(e, g) \in T$ iff $R(f, g) \in T$, and $R(g, e) \in T$ iff $R(g, f) \in T$, for all $C \in \mathsf{cl}(\text{KB})$, $R \in \mathbf{R}$, and $g \in I$,
4. if $A \sqsubseteq C \in$ KB and $A(e) \in T$, then $C(e) \in T$,
5. if $A \sqcap B \sqsubseteq C \in$ KB, $A(e) \in T$, and $B(e) \in T$, then $C(e) \in T$,
6. if $R \sqsubseteq S \in$ KB and $R(e, f) \in T$, then $S(e, f) \in T$,
7. $\exists R.\top(e) \in T$ iff $R(e, f) \in T$ for some $f \in I$,
8. if $\forall R.C(e) \in T$, then $C(f) \in T$ for all $f \in I$ with $R(e, f) \in T$,

A *tableau* is said contain a *clash* if it contains a statement of the form $\bot(e)$. ◇

Proposition 6.3.8 *A Horn-$\mathcal{FLOH}^-$ knowledge base* KB *is satisfiable iff it has a clash-free tableau.*

Proof. Assume that KB has a clash-free tableau $\langle I, T\rangle$. An interpretation $\mathcal{I}$ is defined as follows. Due to condition 3 in Definition 6.3.7, we can define an equivalence relation $\sim$ on I by setting $e \sim f$ iff there is some $g \in \mathbf{I}$ with $\{\{g\}(e), \{g\}(f)\} \subseteq T$. The domain $I_\sim$ of $\mathcal{I}$ is the set of equivalence classes of $\sim$. The interpretation function is defined by setting $e^\mathcal{I} = [e]_\sim$, $e^\mathcal{I} \in C^\mathcal{I}$ iff $C(e) \in T$, and $(e^\mathcal{I}, f^\mathcal{I}) \in R^\mathcal{I}$ iff $R(e, f) \in T$, for all elements $e, f \in I$, concept names C, and role names R. It is easy to see that $\mathcal{I}$ satisfies KB.

For the converse, assume that KB is satisfiable, and that it thus has some model $\mathcal{I}$. We define a tableau $\langle I, T\rangle$ where I is the domain of $\mathcal{I}$. Further, we set $C(e) \in T$ iff $e \in C^\mathcal{I}$, and $R(e, f) \in T$ iff $(e, f) \in R^\mathcal{I}$, where $C \in \mathsf{cl}(\text{KB})$, and R some role name. Again, it is easy to see that this meets the conditions of Definition 6.3.7. □

As is evident by the Turing machine construction in the previous section, some Horn-$\mathcal{FLOH}^-$ knowledge bases may require a model to contain an exponential number of individuals, even within single paths of the computation. Detecting clashes in polynomial space thus requires special care. In particular, a standard tableau procedure with blocking does not execute in polynomial space. Therefore, we first provide a procedural description of a *canonical tableau* which will form the basis for our below decision algorithm.

Definition 6.3.9 An algorithm that computes a tableau-like structure $\langle I, T\rangle$ is defined as follows. Initially, we set $I := \mathbf{I}$, and $T := \emptyset$. The algorithm executes the following steps:

(1) Iterate over all individuals $e \in I$. To each such e, apply rules (T1) to (T10) of Fig. 6.7.

(T1) $T := T \cup \{\top(e)\}$

(T2) if $e \in \mathbf{I}$ is an individual name, $T := T \cup \{\{e\}(e)\}$

(T3) for each $A(e) \in \mathrm{KB}$, $T := T \cup \{A(e)\}$

(T4) for each $R(e, f) \in \mathrm{KB}$, $T := T \cup \{R(e, f)\}$

(T5) for each $\{f\}(e) \in T$

- (T5a) for each $C(f) \in T$, $T := T \cup \{C(e)\}$,
- (T5b) for each $g \in I$ and each $R(f, g) \in T$, $T := T \cup \{R(e, g)\}$; $R(e, g)$ is marked inactive,
- (T5c) for each $g \in I$ and each $R(g, f) \in T$, $T := T \cup \{R(g, e)\}$; $R(g, e)$ is marked inactive,
- (T5d) for each $C(e) \in T$, $T := T \cup \{C(f)\}$,
- (T5e) for each $g \in I$ and each $R(e, g) \in T$, $T := T \cup \{R(f, g)\}$; $R(f, g)$ is marked inactive,
- (T5f) for each $g \in I$ and each $R(g, e) \in T$, $T := T \cup \{R(g, f)\}$; $R(g, f)$ is marked inactive

(T6) for each $A \sqsubseteq C \in \mathrm{KB}$, if $A(e) \in T$ then $T := T \cup \{C(e)\}$

(T7) for each $A \sqcap B \sqsubseteq C \in \mathrm{KB}$, if $A(e) \in T$ and $B(e) \in T$ then $T := T \cup \{C(e)\}$

(T8) for each $R \sqsubseteq S \in \mathrm{KB}$, do the following:

- (T8a) for each $f \in I$, if $R(e, f) \in T$ and $R(e, f)$ is not inactive, then $T := T \cup \{S(e, f)\}$,
- (T8b) if $\exists R.\top(e) \in T$ then $T := T \cup \{\exists S.\top(e)\}$

(T9) for each $f \in I$ and $R(e, f) \in T$ with $R(e, f)$ not inactive, $T := T \cup \{\exists R.\top(e)\}$

(T10) for each $\forall R.C(e) \in T$ and each $f \in I$ with $R(e, f) \in T$,
if $R(e, f)$ is not inactive, then $T := T \cup \{C(f)\}$

($\exists$) for each $\exists R.\top(e) \in T$, if $R(e, f) \notin T$ for all $f \in I$ then
$I := I \cup \{g\}$ and $T := T \cup \{R(e, g)\}$, where g is a fresh individual

Figure 6.7: Constructing tableaux for Horn-$\mathcal{FLOH}^-$ knowledge bases

(2) If T was changed in the previous step, go to (1).

(3) Apply rule ($\exists$) of Fig. 6.7 to all existing elements $e \in I$.

(4) If T was changed by the previous step, go to (1).

(5) Halt.

$\diamond$

While most rules should be obvious, some require explanations. The rules (T5) are used to ensure that individuals e satisfying a nominal class are synchronised with the respective named individual $f \in \mathbf{I}$. The six sub-rules are needed since one generally cannot add $\{e\}(f)$ to T as e might not be an element of $\mathbf{I}$. However, role statements that are inferred in this way need not be taken into account as premises in other deduction rules, since they are guaranteed to have an active original. Whatever could be inferred using copied role statements and rules (T8a), (T9), or (T10), can as well be inferred via the active original from which the inactive role was initially created. Note that this argument involves an induction over the number of applications of rule (T5).

Rule (T8) is also special. In principle, one could omit (T8b), and use rules (T8a) and (T9) instead. This inference, however, is the only case where a role-successor of some individual e might contribute to the classes inferred for e. By providing rule (T8b), the class expressions containing e can be computed without considering any role successor, and rule (T9) is essential only when role expressions have been inferred from ABox statements. In combination with the delayed application of rule ($\exists$), this ensures that concepts are indeed inferred by (T8b) rather than by (T8a)+(T9), which will be exploited in the proof of Lemma 6.3.13 below.

Also note that the algorithm of Definition 6.3.9 is not a decision procedure, since we do not require the algorithm to halt. What we are interested in, however, is the (possibly infinite) tableau that the algorithm constructs in the limit. The existence of this limit is evident from the fact that all completion rules are finitary, and that each rule monotonically increases the size of the computed structure. It is easy to see that there is a correspondence between the rules of Fig. 6.7 and the conditions of Definition 6.3.7, so that the limit structure will indeed meet all the requirements imposed on a tableau. For a given knowledge base KB, we write $\langle \bar{I}_{\mathrm{KB}}, \bar{T}_{\mathrm{KB}} \rangle$ to denote the *canonical tableau* constructed by the above algorithm from KB, where the subscripts are omitted when understood. It is easy to see that, whenever the canonical tableau contains a clash, this must be the case for all possible tableaux.

The algorithm of Definition 6.3.9 can be viewed as a "breadth-first" construction of a canonical tableau. Due to the explicit procedural description of tableau rules, any role and class expression of the canonical tableau is first computed after a well-defined number of computation steps.[1] Accordingly, we define a total order $\prec$ on $\bar{T}$ by setting $F \prec G$ iff F is computed before G.

The canonical tableau and the order $\prec$ are the main ingredients for showing the correctness of following non-deterministic decision algorithm.

Definition 6.3.10 Consider a Horn-$\mathcal{FLOH}^-$ knowledge base KB with canonical tableau $\langle \bar{I}, \bar{T} \rangle$. A set of individuals I is defined as $I := \mathbf{I} \cup \{a, b\}$, where $a, b \notin \bar{I}$.

[1]For this to be true, one must also specify the order for the involved iterations, e.g. by ordering elements lexicographically and adopting a naming scheme for newly introduced elements. We assume that such an order was chosen.

Non-deterministically select one element $g \in I$, and initialise $T \subseteq \mathcal{T}_I$ by setting $T := \{\bot(g)\}$.

The algorithm repeatedly modifies T by non-deterministically applying one of the following rules:

(N1) Given any $X \in \mathcal{T}_I$, set $T := T \cup \{X\}$. If X is a role statement, decide non-deterministically whether X is marked inactive.

(N2) If there is some individual $e \in I$ and $X \in T$ such that X can be derived from $T \setminus \{X\}$ using one of the rules (T1) to (T10) in Fig. 6.7, set $T := T \setminus \{X\}$. Rules (T5b), (T5c), (T5e), and (T5f) can only be used if X is marked inactive.

(N3) If $T_a = \{R(e,a)\}$ for some $e \in I \setminus \{a\}$ such that $\exists R.\top(e) \in T$, set $T := T \setminus T_a$.

(N4) If $T_a = \emptyset$, set $T := (T \cup T_{b \mapsto a}) \setminus T_b$.

(N5) If $T = \emptyset$, return "unsatisfiable." $\diamond$

Lemma 6.3.11 *The algorithm of Definition 6.3.10 can be executed in polynomially bounded space.*

Proof. Since $|I|$, $|\mathbf{B}|$, and $|\mathbf{R}|$ are polynomially bounded by the size of the knowledge base, so is $\mathsf{cl}(\mathrm{KB})$ and thus T. $\square$

Lemma 6.3.12 *If there is a sequence of choices such that the algorithm of Definition 6.3.10 returns "unsatisfiable" after some finite time,* KB *is indeed unsatisfiable.*

Proof. Intuitively, the non-deterministic algorithm applies rules of the algorithm in Definition 6.3.9 in reverse order, deleting a conclusion whenever it can be derived from the remaining statements. The anonymous individuals a and b are used to dynamically represent (various) elements from the canonical tableau. For a formal proof, assume that the algorithm terminates within finitely many steps, and, without loss of generality, that each step involves a successful application of one of the rules (N1) to (N5). We use T^n to denote the state of the algorithm n steps before termination. In particular, $T^0 = \emptyset$.

We claim that for each T^n there are individuals $e, f \in \bar{I}$, such that $T^n_{a \mapsto e, b \mapsto f} \subseteq \bar{T}$. This is verified by induction over the number of steps executed by the algorithm. Since $T^0 = \emptyset$, the claim for T^0 holds for any $e, f \in \bar{I}$.

For the induction step, assume that $T^n_{a \mapsto e, b \mapsto f} \subseteq \bar{T}$. To show the claim for T^{n+1}, we distinguish by the transformation rule that was applied to obtain T^n from T^{n+1}:

(N1) Since $T^{n+1} \subset T^n$, we conclude $T^{n+1}_{a \mapsto e, b \mapsto f} \subseteq \bar{T}$.

(N2) $T^{n+1} = T^n \cup \{X\}$, where X can be derived from T^n by one of the rules (T1) to (T10). Since those rules have been applied exhaustively in $\bar{T}$, we find $T^{n+1}_{a\mapsto e,\, b\mapsto f} \subseteq \bar{T}$.

(N3) We find $T^n_a = \emptyset$ and, for some $g \in I\setminus\{a\}$ and $R \in \mathbf{R}$, $T^{n+1} = T^n \cup \{R(g,a)\}$ and $\exists R.\top(g) \in T^n$. Define $g' := f$ if $g = b$, and $g' = g$ otherwise. We conclude that $\exists R.\top(g') \in \bar{T}$ and thus there is some individual $e' \in \bar{I}$ with $R(g', e')$. We conclude that $T^{n+1}_{a\mapsto e',\, b\mapsto f} \subseteq \bar{T}$.

(N4) This rule merely exchanges b with (the unused) a, so we have $T^{n+1}_{a\mapsto f,\, b\mapsto e} \subseteq \bar{T}$.

Applying the above induction to the initial state $\{\bot(g)\}$, we find $\{\bot(g)\}_{a\mapsto e,\, b\mapsto f} \in \bar{T}$. Hence $\bar{T}$ indeed contains a clash and KB is unsatisfiable. □

Lemma 6.3.13 *Whenever* KB *is unsatisfiable, there is a sequence of choices such that the algorithm of Definition 6.3.10 returns "unsatisfiable" after some finite time.*

Proof. We first specify a possible sequence of choices, and then show its correctness. If KB is unsatisfiable, there is some element $e \in \bar{I}$ in the canonical tableau such that $\bot(e) \in \bar{T}$. Pick one such e. We use a' and b' to denote the elements of $\bar{I}$ that are currently simulated by a and b. Initially, we set $a' = b' = \square$ for some element $\square \notin \bar{I}$. Rule (N1) of the algorithm will repeatedly be used to close T under relevant inferences that are $\prec$-smaller than some statement X. Given $X \in \bar{T}$, we define:

$$\begin{aligned}\downarrow X = &\Big\{C(f) \in \bar{T} \mid C(f) \preceq X,\ f \in \mathbf{I} \cup \{a', b'\}\Big\}_{a'\mapsto a,\, b'\mapsto b} \quad \cup \\ &\Big\{R(f,g) \in \bar{T} \mid R(f,g) \text{ not inactive}, R(f,g) \preceq X,\ f, g \in \mathbf{I} \cup \{a', b'\}\Big\}_{a'\mapsto a,\, b'\mapsto b}.\end{aligned}$$

This selects all elements in $\bar{T}$ that can be represented using the elements from I with the current representation of a' as a, and b' as b. Throughout the below computation, the following property will be preserved:

$$T_{a\mapsto a',\, b\mapsto b'} \subseteq \bar{T} \qquad (\dagger)$$

Now if $e \in \mathbf{I}$, set $a' := e$. Using the non-deterministic initialisation and rule (N1), the algorithm of Definition 6.3.10 can now compute $T = \downarrow\{\bot(e)\}$. The algorithm now repeatedly executes steps according to the following choice strategy.

Single Step Choice Strategy If T_a is non-empty, let X be the $\prec$-largest element of T_a. Else, let X be the $\prec$-largest element of T. By property (†), there is some $X' \in \bar{T}$ with $\{X\}_{a\mapsto a',\, b\mapsto b'} = \{X'\}$. Applying rule (N1), the algorithm first computes $T := T \cup \downarrow X$ (∗). The algorithm non-deterministically guesses the rule of Fig. 6.7 that was used to infer X', and proceeds accordingly:

- If X' was inferred by one of the rules (T1), (T2), (T3), (T4), (T6), (T7), (T8a), (T8b), and (T9), the premises of a respective rule application in T have been computed in $(*)$. This is so since the required premises are $\prec$-smaller and not inactive, and since they only involve individuals that are also found in X, i.e. which are represented by I with the current choice of a' and b'. Hence the algorithm can apply rule (N2) to reduce X.
- If X' was inferred by one of the rules of (T5), then one of the premises used was of the form $\{f\}(e)$, and thus $f \in \mathbf{I}$. Since inactive roles are not generated by any of the given choices, rules (T5b), (T5c), (T5e), and (T5f) are not relevant. If X' was inferred by rule (T5a) then X can directly be reduced by applying rule (N2). The existence of the premises in T follows again from $(*)$.

  If X' was inferred by rules (T5d), then X' is of the form $C(f)$ and thus $T_a = \emptyset$. If the individual e in the premise is in $\mathbf{I}$, then X again can be reduced by rule (N2). If $e \notin \mathbf{I}$, set $a' = e$ and use rule (N1) to compute $T_a = \{\{f\}(e), C(e)\}$. Apply (N2) to reduce X.
- If X' was inferred by rule (T10), then $X' = C(g)$ for some element g, and there is some element e such that $\{\forall R.C(e), R(e, g)\} \subseteq \bar{T}$. We distinguish two cases:

  - If $g \in \mathbf{I}$, then $X = C(g)$ and $T_a = \emptyset$. Set $a' = e$ and use rule (N1) to compute $T_a = \{\forall R.C(a), R(a, g)\}$. Use rule (N2) to reduce X.
  - If $g \notin \mathbf{I}$, then $X = C(a)$ and $e \neq a'$. If $e \in \mathbf{I} \cup \{b'\}$, then $\{\forall R.C(e), R(e, a)\} \subseteq T$ by $(*)$. Use rule (N2) to reduce X. If $e \notin \mathbf{I} \cup \{b'\}$, then $b' = \square$ and $T_b = \emptyset$, as we will show below. Set $b' = e$ and use rule (N1) to compute $T_b = \{\forall R.C(b), R(b, g)\}$. Use rule (N2) to reduce X.

    We claimed that $b' = \square$ whenever it is not equal to the predecessor e. This is so, since $a' \notin \mathbf{I}$ is ensured by each step of the algorithm, and since elements that are not in $\mathbf{I}$ are involved in active role statements of exactly one predecessor (the one which generated a'). This is easily verified by inspecting the rules that can create role statements.

- If X' was inferred by rule ($\exists$), we have $X' = R(e, g)$ for some newly introduced element $g \notin \mathbf{I}$. Thus X is of the form $R(e', a)$, and, since X was selected to be $\prec$-maximal, $T_a = \{X\}$. Thus we can apply rule (N3) to reduce X. In addition, the algorithm applies rule (4) to copy b to the (now empty) a, and we set $a' := b'$ and $b' := \square$.

With the above choices, the algorithm instantiates elements a on demand, and repeatedly reduces the statements of those elements. The individual rules show that this reduction might require another (predecessor) individual b to be considered, but that no further element is needed. Also note that rule (T8b) is required to ensure that all concept expressions in T_a can be reduced without generating any role successors for a. Hence, it is evident that the above choice strategy ensures that exactly one of the above reductions is applicable in each step.

Finally, we need to show that the algorithm terminates. This claim is established by defining a well-founded termination order. For details on such approaches and the related terminology, see [BN98]. Now considering T as a multiset, the multiset-extension of the well-founded order $\prec$ is a suitable termination order, which is easy to see since in every reduction step, the element X is deleted, and possibly replaced by one or more elements that are strictly smaller than X. □

The above lemmata establish an NPSpace decision procedure for detecting unsatisfiability of Horn-$\mathcal{FLOH}^-$ knowledge bases. But NPSpace is known to coincide with PSpace, and we can conclude the main theorem of this section.

Theorem 6.3.14 *Unsatisfiability of a Horn-$\mathcal{FLOH}^-$ knowledge base* KB *can be decided in space that is polynomially bounded by the size of* KB.

Proof. Combine Lemma 6.3.11, 6.3.12, and 6.3.13 to obtain a non-deterministic time-polynomial decision procedure for detecting unsatisfiability. Apply *Savitch's Theorem* to show the existence of an according PSpace algorithm [Pap94]. □

Summing up the result from the previous two sections, we obtain the following.

Theorem 6.3.15 *The standard reasoning problems for any description logic between Horn-$\mathcal{FL}^-$ and Horn-$\mathcal{FLOH}^-$ are* PSpace*-complete.*

Proof. Combine Lemma 6.3.5 and Theorem 6.3.14. □

6.4 Horn-$\mathcal{SHIQ}$ and Other ExpTime-Complete Horn DLs

$\mathcal{FLE}$ further extends $\mathcal{FL}^-$ by allowing arbitrary existential role quantifications, which turns out to raise the complexity of standard reasoning tasks for Horn-$\mathcal{FLE}$ to ExpTime, thus establishing ExpTime-completeness of Horn-$\mathcal{SHIQ}$. Note that inclusion in ExpTime is obvious since $\mathcal{FLE}$ is a fragment of $\mathcal{SHIQ}$ which is also in ExpTime [Tob01]. To show that Horn-$\mathcal{FLE}$ is ExpTime-hard, we reduce the halting problem of polynomially space-bounded alternating Turing machines, defined next, to the concept subsumption problem.

6.4.1 Alternating Turing Machines

Definition 6.4.1 An *alternating Turing machine* (ATM) $\mathcal{M}$ is a tuple (Q, Σ, Δ, q_0) where

- $Q = U \dot\cup E$ is the disjoint union of a finite set of *universal states* U and a finite set of *existential states* E,
- Σ is a finite *alphabet* that includes a *blank* symbol $\square$,
- $\Delta \subseteq (Q \times \Sigma) \times (Q \times \Sigma \times \{l, r\})$ is a *transition relation*, and
- $q_0 \in Q$ is the *initial state*.

A (universal/existential) *configuration* of $\mathcal{M}$ is a word $\alpha \in \Sigma^* Q \Sigma^*$ ($\Sigma^* U \Sigma^* / \Sigma^* E \Sigma^*$). A configuration α' is a *successor* of a configuration α if one of the following holds:

1. $\alpha = w_l q \sigma \sigma_r w_r$, $\alpha' = w_l \sigma' q' \sigma_r w_r$, and $(q, \sigma, q', \sigma', r) \in \Delta$,
2. $\alpha = w_l q \sigma$, $\alpha' = w_l \sigma' q' \square$, and $(q, \sigma, q', \sigma', r) \in \Delta$,
3. $\alpha = w_l \sigma_l q \sigma w_r$, $\alpha' = w_l q' \sigma_l \sigma' w_r$, and $(q, \sigma, q', \sigma', l) \in \Delta$,

where $q \in Q$ and $\sigma, \sigma', \sigma_l, \sigma_r \in \Sigma$ as well as $w_l, w_r \in \Sigma^*$. Given some natural number s, the possible *transitions in space* s are defined by additionally requiring that $|\alpha'| \leq s + 1$.

The set of *accepting configurations* is the least set which satisfies the following conditions. A configuration α is accepting iff

- α is a universal configuration and all its successor configurations are accepting, or
- α is an existential configuration and at least one of its successor configurations is accepting.

Note that universal configurations without any successors here play the rôle of accepting final configurations, and thus form the basis for the recursive definition above.

$\mathcal{M}$ *accepts* a given word $w \in \Sigma^*$ (in space s) iff the configuration $q_0 w$ is accepting (when restricting to transitions in space s). ◇

This definition is inspired by the complexity classes NP and co-NP, which are characterised by non-deterministic Turing machines that accept an input if either at least one or all possible runs lead to an accepting state. An ATM can switch between these two modes and indeed turns out to be more powerful than classical Turing machines of either kind. In particular, ATMs can solve ExpTime problems in polynomial space [CKS81].

Definition 6.4.2 A language L is accepted by a polynomially space-bounded ATM iff there is a polynomial p such that, for every word $w \in \Sigma^*$, $w \in L$ iff w is accepted in space $p(|w|)$. ◇

Fact 6.4.3 *The complexity class* APSPACE *of languages accepted by polynomially space-bounded ATMs coincides with the complexity class* EXPTIME.

We thus can show EXPTIME-hardness of Horn-$\mathcal{SHIQ}$ by polynomially reducing the halting problem of ATMs with a polynomially bounded storage space to inferencing in Horn-$\mathcal{SHIQ}$. In the following, we exclusively deal with polynomially space-bounded ATMs, and so we omit additions such as "in space s" when clear from the context.

6.4.2 Simulating ATMs in Horn-$\mathcal{FLE}$

In the following, we consider a fixed ATM $\mathcal{M}$ denoted as in Definition 6.4.1, and a polynomial p that defines a bound for the required space. For any word $w \in \Sigma^*$, we construct a Horn-$\mathcal{FLE}$ knowledge base $\mathrm{KB}_{\mathcal{M},w}$ and show that acceptance of w by the ATM $\mathcal{M}$ can be decided by inferencing over this knowledge base.

In detail, $\mathrm{KB}_{\mathcal{M},w}$ depends on $\mathcal{M}$ and $p(|w|)$, and has an empty ABox.[2] Acceptance of w by the ATM is reduced to checking concept subsumption, where one of the involved concepts directly depends on w. Intuitively, the elements of an interpretation domain of $\mathrm{KB}_{\mathcal{M},w}$ represent possible configurations of $\mathcal{M}$, encoded by the following concept names:

- A_q for $q \in Q$: the ATM is in state q,
- H_i for $i = 0, \ldots, p(|w|) - 1$: the ATM is at position i on the storage tape,
- $C_{\sigma,i}$ with $\sigma \in \Sigma$ and $i = 0, \ldots, p(|w|) - 1$: position i on the storage tape contains symbol σ,
- A: the ATM accepts this configuration.

This approach is pretty standard, and it is not too hard to axiomatise a successor relation S and appropriate acceptance conditions in $\mathcal{ALC}$ (see, e.g., [LS05]). But this reduction is not applicable in Horn-$\mathcal{SHIQ}$, and it is not trivial to modify it accordingly.

One problem that we encounter is that the acceptance condition of existential states is a (non-Horn) disjunction over possible successor configurations. To overcome this, we encode individual transitions by using a distinguished successor relation for each translation in Δ. This allows us to explicitly state which conditions

[2]The RBox is empty for $\mathcal{FLE}$ anyway.

(1) Left and right transition rules:

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq \exists S_\delta.(A_{q'} \sqcap H_{i+1} \sqcap C_{\sigma',i})$ with $\delta = (q,\sigma,q',\sigma',r)$, $i < p(|w|) - 1$

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq \exists S_\delta.(A_{q'} \sqcap H_{i-1} \sqcap C_{\sigma',i})$ with $\delta = (q,\sigma,q',\sigma',l)$, $i > 0$

(2) Memory:

$H_j \sqcap C_{\sigma,i} \sqsubseteq \forall S_\delta.C_{\sigma,i} \quad i \neq j$

(3) Existential acceptance:

$A_q \sqcap \exists S_\delta.A \sqsubseteq A$ for all $q \in E$

(4) Universal acceptance:

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqcap \bigsqcap_{\delta\in\tilde{\Delta}}(\exists S_\delta.A) \sqsubseteq A$ $\quad q \in U$, $x \in \{r \mid i < p(|w|) - 1\} \cup \{l \mid i > 0\}$

$\tilde{\Delta} = \{(q,\sigma,q',\sigma',x) \in \Delta\}$

Rules are instantiated for all $q, q' \in Q$, $\sigma, \sigma' \in \Sigma$, $i, j \in \{0, \ldots, p(|w|) - 1\}$, and $\delta \in \Delta$.

Figure 6.8: Knowledge base $\mathrm{KB}_{\mathcal{M},w}$ simulating a polynomially space-bounded ATM

must hold for a particular successor without requiring disjunction. For the acceptance condition, we use a recursive formulation as employed in Definition 6.4.1. In this way, acceptance is propagated backwards from the final accepting configurations.

In the case of $\mathcal{ALC}$, acceptance of the ATM is reduced to concept satisfiability, i.e. one checks whether an accepting initial configuration can exist. This requires that acceptance is faithfully propagated to successor states, so that any model of the initial concept encodes a valid trace of the ATM. Axiomatising this requires many exclusive disjunctions, such as "The ATM always is in *exactly* one of its states H_i." Since it is not clear how to model this in a Horn-DL, we take a dual approach: reducing acceptance to concept subsumption, we require the initial state to be accepting in *all* possible models. We therefore may focus on the task of propagating properties to successor configurations, while not taking care of disallowing additional statements to hold. Our encoding ensures that, whenever the initial configuration is not accepting, there is at least one "minimal" model that reflects this.

After this informal introduction, consider the knowledge base $\mathrm{KB}_{\mathcal{M},w}$ given in Fig. 6.8. The roles S_δ, $\delta \in \Delta$, describe a configuration's successors using the translation δ. The initial configuration for word w is described by the concept expression I_w:

$$I_w := A_{q_0} \sqcap H_0 \sqcap C_{\sigma_0,0} \sqcap \ldots \sqcap C_{\sigma_{|w|-1},|w|-1} \sqcap C_{\square,|w|} \sqcap \ldots \sqcap C_{\square,p(|w|)-1},$$

where σ_i denotes the symbol at the ith position of w. We will show that checking whether the initial configuration is accepting is equivalent to checking whether $I_w \sqsubseteq A$ follows from $\mathrm{KB}_{\mathcal{M},w}$. The following is obvious from the characterisation given in Definition 6.1.1.

Lemma 6.4.4 $\mathrm{KB}_{\mathcal{M},w}$ *and* $I_w \sqsubseteq A$ *are in Horn-*$\mathcal{FLE}$*.*

Next we need to investigate the relationship between elements of an interpretation that satisfies $\mathrm{KB}_{\mathcal{M},w}$ and configurations of $\mathcal{M}$. Given an interpretation $\mathcal{I}$ of $\mathrm{KB}_{\mathcal{M},w}$, we say that an element e of the domain of $\mathcal{I}$ *represents* a configuration $\sigma_1 \ldots \sigma_{i-1} q \sigma_i \ldots \sigma_m$ if $e \in A_q^{\mathcal{I}}$, $e \in H_i^{\mathcal{I}}$, and, for every $j \in \{0, \ldots, p(|w|) - 1\}$, $e \in C_{\sigma,j}^{\mathcal{I}}$ whenever

$$j \leq m \text{ and } \sigma = \sigma_m \qquad \text{or} \qquad j > m \text{ and } \sigma = \square.$$

Note that we do not require uniqueness of the above, so that a single element might in fact represent more than one configuration. As we will see below, this does not affect our results. If e represents a configuration as above, we will also say that e has state q, position i, symbol σ_j at position j etc.

Lemma 6.4.5 *Consider some interpretation* $\mathcal{I}$ *that satisfies* $\mathrm{KB}_{\mathcal{M},w}$*. If some element e of* $\mathcal{I}$ *represents a configuration α and some transition δ is applicable to α, then e has an* $S_\delta^{\mathcal{I}}$*-successor that represents the (unique) result of applying δ to α.*

Proof. Consider an element e, state α, and transition δ as in the claim. Then one of the axioms (1) applies, and e must also have an $S_\delta^{\mathcal{I}}$-successor. This successor represents the correct state, position, and symbol at position i of e, again by the axioms (1). By axiom (2), symbols at all other positions are also represented by all $S_\delta^{\mathcal{I}}$-successors of e. □

Lemma 6.4.6 *A word w is accepted by* $\mathcal{M}$ *iff* $I_w \sqsubseteq A$ *is a consequence of* $\mathrm{KB}_{\mathcal{M},w}$*.*

Proof. Consider an arbitrary interpretation $\mathcal{I}$ that satisfies $\mathrm{KB}_{\mathcal{M},w}$. We first show that, if any element e of $\mathcal{I}$ represents an accepting configuration α, then $e \in A^{\mathcal{I}}$.

We use an inductive argument along the recursive definition of acceptance. If α is a universal configuration then all successors of α are accepting, too. By Lemma 6.4.5, for any δ-successor α' of α there is a corresponding $S_\delta^{\mathcal{I}}$-successor e' of e. By the induction hypothesis for α', e' is in $A^{\mathcal{I}}$. Since this holds for all δ-successors of α, axiom (4) implies $e \in A^{\mathcal{I}}$. Especially, this argument covers the base case where α has no successors.

If α is an existential configuration, then there is some accepting δ-successor α' of α. Again by Lemma 6.4.5, there is an $S_\delta^{\mathcal{I}}$-successor e' of e that represents α', and $e' \in A^{\mathcal{I}}$ by the induction hypothesis. Hence axiom (3) applies and also conclude $e \in A^{\mathcal{I}}$.

Since all elements in $I_w^{\mathcal{I}}$ represent the initial configuration of the ATM, this shows that $I_w^{\mathcal{I}} \subseteq A^{\mathcal{I}}$ whenever the initial configuration is accepting.

It remains to show the converse: if the initial configuration is not accepting, there is some interpretation $\mathcal{I}$ such that $I_w^{\mathcal{I}} \nsubseteq A^{\mathcal{I}}$. To this end, we define a canonical interpretation M of $\mathrm{KB}_{\mathcal{M},w}$ as follows. The domain of M is the set of all configurations of $\mathcal{M}$ that have size $p(|w|) + 1$ (i.e. that encode a tape of length $p(|w|)$, possibly with trailing blanks). The interpretations for the concepts A_q, H_i, and $C_{\sigma,i}$ are defined as expected so that every configuration represents itself but no other configuration. Especially, I_w^M is the singleton set containing the initial configuration. Given two configurations α and α', and a transition δ, we define $(\alpha, \alpha') \in S_\delta^M$ iff there is a transition δ from α to α'. A^M is defined to be the set of accepting configurations.

By checking the individual axioms of Fig. 6.8, it is easy to see that M satisfies $\mathrm{KB}_{\mathcal{M},w}$. Now if the initial configuration is not accepting, $I_w^M \not\sqsubseteq A^M$ by construction. Thus M is a counterexample for $I_w \sqsubseteq A$ which thus is not a logical consequence.□

We can summarise our results as follows.

Theorem 6.4.7 *The standard reasoning problems for any description logic between Horn-$\mathcal{FLE}$ and Horn-$\mathcal{SHIQ}$ are* ExpTime*-complete.*

Proof. Inclusion is obvious as Horn-$\mathcal{SHIQ}$ is a fragment of $\mathcal{SHIQ}$ for which these problems are in ExpTime [Tob01]. Regarding hardness, Lemma 6.4.6 shows that the word problem for polynomially space-bounded ATMs can be reduced to checking concept subsumption in $\mathrm{KB}_{\mathcal{M},w}$. By Lemma 6.4.4, $\mathrm{KB}_{\mathcal{M},w}$ is in Horn-$\mathcal{FLE}$. The reduction is polynomially bounded due to the restricted number of axioms: there are at most $2 \times |Q| \times p(|w|) \times |\Sigma| \times |\Delta|$ axioms of type (1), $p(|w|)^2 \times |\Sigma| \times |\Delta|$ of type (2), $|Q| \times |\Sigma|$ of type (3), and $|Q| \times p(|w|) \times |\Sigma|$ of type (4). □

Note that, even in Horn logics, it is straightforward to reduce knowledge base satisfiability to the entailment of the concept subsumption $\top \sqsubseteq \bot$. The proof that was used to establish the previous result is suitable for obtaining further complexity results for logical fragments that are not above Horn-$\mathcal{FLE}$.

Theorem 6.4.8 *Consider the description logics*

(a) $\mathcal{ELF}$ obtained by extending $\mathcal{EL}$ with number restrictions of the form $\leqslant 1\, R.\top$,

(b) $\mathcal{FL}\circ^-$ obtained by extending $\mathcal{FL}^-$ with composition of roles while restricting to regular RBoxes, and

(c) $\mathcal{FLI}^-$ obtained by extending $\mathcal{FL}^-$ with inverse roles,

and let Horn-$\mathcal{ELF}$, Horn-$\mathcal{FL}\circ^-$, and Horn-$\mathcal{FLI}^-$ denote the respective Horn DLs defined as in Definition 6.1.6.

Horn-$\mathcal{FL}\circ^-$ is ExpTime*-hard. Horn-$\mathcal{ELF}$ and Horn-$\mathcal{FLI}^-$ are* ExpTime*-complete.*

Proof. The results are established by modifying the knowledge base $\mathrm{KB}_{\mathcal{M},w}$ to suite the given fragment. We restrict to providing the required modifications; the full proofs are analogous to the proof for Horn-$\mathcal{FLE}$.

(a) Replace axioms (2) in Fig. 6.8 with the following statements:
$\top \sqsubseteq \leq 1\, S_\delta.\top \qquad H_j \sqcap C_{\sigma,i} \sqcap \exists S_\delta.\top \sqsubseteq \exists S_\delta.C_{\sigma,i},\ i \neq j$

(b) Replace axioms (1) with axioms of the form

$$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq \exists S_\delta.\top \sqcap \forall S_\delta.(A_{q'} \sqcap H_{i\pm1} \sqcap C_{\sigma',i}).$$

Any occurrence of concept A is replaced by $\exists R_A.\top$, with R_A a new role. Moreover, we introduce roles $R_{A\delta}$ for each transition δ, and replace any occurrence of $\exists S_\delta.A$ with $\exists R_{A\delta}.\top$. Finally, the following axioms are added:

$$S_\delta \circ R_A \sqsubseteq R_{A\delta} \qquad \text{for each } \delta \in \Delta.$$

(c) Axioms (1) are replaced as in (b). Any occurrence of $\exists S_\delta.A$ is now replaced with a new concept name $A_{S\delta}$, and the following axioms are added:

$$A \sqsubseteq \forall S_\delta^{-1}.A_{S\delta} \qquad \text{for each } \delta \in \Delta.$$

It is easy to see that those changes still enable analogous reductions. Inclusion results for Horn-$\mathcal{ELF}$ and Horn-$\mathcal{FLI}^-$ are immediate from their inclusion in $\mathcal{SHIQ}$. □

ExpTime-completeness of $\mathcal{ELF}$ was shown in [BBL05] (where it was called $\mathcal{EL}^{\leq 1}$), but the above theorem sharpens this result to the Horn case, and provides a more direct proof. Theorems 6.4.7 and 6.4.8 thus can be viewed as sharpenings of the hardness results on extensions of $\mathcal{EL}$.

6.5 Summary

In this chapter, we have generalised the well-known definition of Horn-$\mathcal{SHIQ}$ to Horn-$\mathcal{SROIQ}^{\text{free}}$, and derived a simplified characterisation of Horn DLs based on a formal grammar. We have then studied a number of increasingly expressive Horn description logics that are obtained as fragments of Horn-$\mathcal{SROIQ}^{\text{free}}$ w.r.t. their worst-case inferencing complexities. The reported results are summarised in Fig. 6.9. Some non-Horn DLs – $\mathcal{EL}$, $\mathcal{RL}$, $\mathcal{SHIQ}$, $\mathcal{SHOIQ}$, and $\mathcal{SROIQ}$ – are also displayed in this context, while $\mathcal{FL}_0$ and $\mathcal{FL}^-$ (both ExpTime) are omitted to simplify the presentation. The complexity results for Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ do not follow from this work: they have been established only very recently [ORS10].

The entry for Horn-$\mathcal{FL}\circ^-$ in Fig. 6.9 is displayed in a dotted box to indicate that its exact position is not certain. We have established ExpTime hardness, which suffices to demonstrate that this extension of Horn-$\mathcal{FL}^-$ does no longer admit reasoning in PSpace.[3] The 2ExpTime upper bound for the complexity follows from

[3]Unless PSpace = ExpTime.

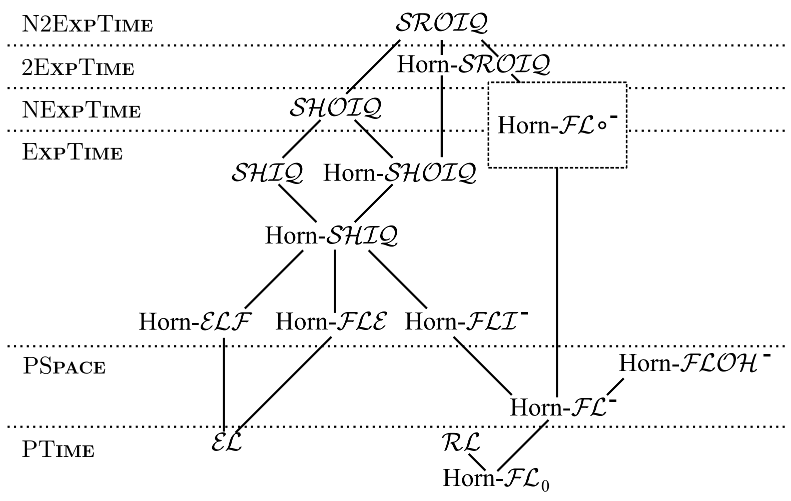

Figure 6.9: Reasoning complexities of Horn DLs; the exact position of Horn-$\mathcal{FL}\circ^-$ is not known

the according result for Horn-$\mathcal{SROIQ}$ [ORS10]. Further checks are needed to determine the exact complexity of Horn-$\mathcal{FL}\circ^-$. But when considering the fact that no Horn DL is known to be cmoplete for a non-deterministic complexity class, it seems to be extremely unlikely that this DL is complete for NExpTime. Indeed, we conjecture that this avoidance of non-determinism is inherent to Horn DLs.

A tableau algorithm for reasoning in description logics between Horn-$\mathcal{FL}^-$ and Horn-$\mathcal{FLOH}^-$ has been devised to show the upper complexity bound for reasoning in these logics. In essence, this algorithm achieves its goal in polynomial space by storing only very small portions of the constructed tableau, corresponding to very restricted “local” environments in the according model. The main result therefore consists in showing that such an extremely limited view suffices for complete reasoning in the considered logics. As opposed to Horn-$\mathcal{FL}_0$, the addition of nominals to Horn-$\mathcal{FL}^-$ significantly complicates reasoning procedures, although it does not lead to increased worst-case complexities. Due to a high amount of unguided non-determinism, the tableau algorithm for Horn-$\mathcal{FLOH}^-$ is clearly unsuitable for practical implementation.

Another important theme of this chapter was to establish hardness results that require only a minimal amount of logical expressivity, and which can therefore be useful to derive hardness results for many other DLs as well. This was achieved by directly simulating Turing machine computations in terms of DL inferencing, where polynomially space-bounded Alternating Turing Machines have been found a convenient tool for showing ExpTime hardness. The versatility of this approach was illustrated by deriving a number of additional hardness results for extensions of $\mathcal{EL}$ and $\mathcal{FL}^-$ which extended or strengthened existing results.

6.6 Related Work

Horn-$\mathcal{SHIQ}$ has originally been introduced in [HMS05] where it has been defined as discussed in Section 6.1 but with additional implicit restrictions related to the presence of transitivity. The latter was caused by a method of transitivity elimination that creates non-Horn axioms of the form $\forall R.A \sqsubseteq \forall R.\forall R.A$ for transitive roles R which must be taken into account when defining Horn-$\mathcal{SHIQ}$. As discussed in Section 6.1, this problem can be avoided by encoding transitivity (and other RIAs) by means of automata encoding techniques as used in [DN05] which have first been applied to DLs in [Kaz08]. See Section 9.3 for further discussion. Taking this into account, our formulation of Horn-$\mathcal{SHIQ}$ is slightly more general than the one in [HMS05] and than the formulations used in precursors to this work [KHVS06, KRH06, KRH07a]. While the data complexity of Horn-$\mathcal{SHIQ}$ has been one of the main motives for defining it in [HMS05], the combined complexity result reported herein is new. Recent investigations revealed that even entailment of conjunctive queries for Horn-$\mathcal{SHIQ}$ can be performed in ExpTime [EGOS08], whereas this problem is known to be 2ExpTime-complete for $\mathcal{SHIQ}$ [GLHS08]. Another recent result established the exact reasoning complexity of Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ to be ExpTime and 2ExpTime, respectively [ORS10].

The lower data complexity of reasoning in Horn-$\mathcal{SHIQ}$ has first been exploited by the KAON2 system as described in [Mot06, MS06]. Further algorithms and implementations have since been able to exploit the simpler structure of Horn knowledge bases to achieve tangible performance gains. An example is the *hypertableau* reasoner HermiT that can handle arbitrary $\mathcal{SROIQ}$ (OWL 2) knowledge bases [MSH08, MSH07]. The "consequence-based" reasoning method of [Kaz09a] is restricted to Horn-$\mathcal{SHIQ}$, but shows outstanding performance for practically relevant ontologies that fall into that fragment.

Other notable examples of Horn DLs are provided by light-weight description logics. Indeed, disjunctive information makes reasoning NP-hard in all DLs that support conjunction and GCIs, and hence it is excluded from DLs that allow for polynomial-time reasoning. Thus, it is no surprise to find that $\mathcal{EL}^{++}$ [BBL05, BBL08] and various versions of DL-Lite [CGL+07] are Horn DLs in the sense of this chapter. The same is true for various formulations of DLP [GHVD03, Vol04], as has already been observed in Section 6.2.

Reducing inference problems of DL to inference problems of corresponding datalog programs has been considered in a number of approaches, some of which avoid the use of disjunctions in datalog if the input knowledge base is Horn. See Section 8.7 for an overview of related works.

The description logic $\mathcal{FL}^{-}$ dates back to [BL84] where it was introduced as a presumably tractable variant of the frame language $\mathcal{FL}$. While subsumption of *individual concept expressions* can indeed be decided in polynomial time, the subsumption problem for $\mathcal{FL}^{-}$ and even in $\mathcal{FL}_0$ is ExpTime-hard in the presence of arbitrary $\mathcal{FL}^{-}$ TBoxes, as was first shown by McAllester in an unpublished manuscript of 1991 [DLNS96].

Chapter 7

The Datalog Fragment of Description Logic

Description Logic Programs (DLP) were introduced as a family of fragments of description logic (DL) that can be expressed in first-order Horn-logic [GHVD03, Vol04]. Since common reasoning tasks are still undecidable for first-order Horn-logic, its function-free fragment *datalog* is of particular interest, and the term "DLP" today is most commonly used to refer to tractable DLs that can be translated to equisatisfiable datalog.

This statement is slightly more concrete than describing DLP as a subset of the "expressive intersection" of DL and datalog [GHVD03], but it is still insufficient to characterise DLP. In particular, other tractable DLs such as $\mathcal{SROEL}(\sqcap_s, \times)$ (and thus $\mathcal{EL}$) can also be translated to equisatisfiable datalog programs, as discussed in Section 5.4. The union of DLP and $\mathcal{EL}$ is intractable since it subsumes Horn-$\mathcal{FLE}$ for which ExpTime hardness of reasoning was shown in Section 6.4, but one may still wonder whether DLP is merely one among several equivalent subsets of the "expressive intersection" of DL and datalog.

But tractability was not among the original design goals of DLP, and one might also weaken this principle to require merely a semantics-preserving transformation to datalog. Could the union of DLP and $\mathcal{EL}$ then be considered as an extended version of DLP? Possibly yes, since it is contained in the DL Horn-$\mathcal{SHIQ}$ for which a satisfiability-preserving datalog transformation is known [HMS05]. However, $\mathcal{EL}$ and DLP can be translated to datalog axiom-by-axiom, i.e. in a *modular* fashion, while the known datalog transformation for Horn-$\mathcal{SHIQ}$ needs to consider the whole knowledge base. But how can we be sure that there is no simpler transformation given that both data-complexity and combined complexity of datalog and Horn-$\mathcal{SHIQ}$ agree? The answer is given in Proposition 7.1.1 below.

In any case, it is obvious that the design principles for DLP – but also for $\mathcal{EL}$

and Horn-$\mathcal{SHIQ}$ – are not sufficiently well articulated to clarify the distinction between those formalisms. This chapter thus approaches an explicit characterisation of DLP, not in terms of concrete syntax but in terms of general design principles, which capture the specifics of the known DLP for datalog. An essential principle is *structurality* of the language: a formula should be in DLP based on its term structure, not based on concrete entity names that it uses. Moreover, we ask whether DLP could be defined as a larger, or even as the *largest*, DL that satisfies our design principles. A positive answer to this question is given by introducing a significantly larger variant of DLP that is proven to be a maximal DLP description logic in the sense of this work.

This chapter begins by discussing the problems of characterising DLP and providing some fundamental results in Section 7.1. Section 7.2 presents a simplified version of the main results by restricting attention to the smaller description logic $\mathcal{ALC}$, where it is significantly easier to define a DLP fragment and prove its maximality. These simplifications allow us to outline the general proof structure and some relevant methods, but they do neither cover all relevant parts of earlier DLP definitions nor all relevant proof techniques needed in the general case. A full definition for an extended language $\mathcal{DLP}$ is then provided in Section 7.3. In Section 7.4, we show how $\mathcal{DLP}$ can be expressed using datalog. Section 7.5 discussed some important model-theoretic constructions for characterising fragments of first-order logic that can be expressed in datalog. These constructions are then used as a basis for showing maximality of $\mathcal{DLP}$ in Section 7.6. We discuss our results in Section 7.7 and give pointers to related work in Section 7.7.

A report on some of the results of this chapter is given in [KRS10].

7.1 Initial Considerations and Problem Definition

In this section, we discuss why defining DLP is not straightforward, and we specify various design principles to guide our subsequent definition. The goal is to arrive at a notion of DLP that is characterised by these principles, as opposed to DLP being some *ad hoc* fragment of description logic that happens to be expressible in datalog without being maximal or canonical in any sense. The first design principle fixes our choice of syntax and underlying DL:

DLP 1 (DL Syntax) Every DLP knowledge base should be a $\mathcal{SROIQ}^{\text{free}}$ knowledge base.

The second principle states that the semantics of every DLP knowledge base can be expressed in datalog. We will see below that it is sometimes useful to introduce auxiliary symbols during the translation to datalog. If this is done, the datalog program can no longer be semantically equivalent to the original knowledge base, even if all consequences with respect to the original predicates are still the same. Yet, *equisatisfiability* – the requirement that a DLP knowledge base is satisfiable iff its datalog translation is – turns out to be too weak for many purposes. A suitable compromise is the notion of *emulation* as introduced in Section 2.2:

DLP 2 (Semantic Correspondence) There should be a transformation function datalog that maps a DLP knowledge base KB to a datalog program datalog(KB) such that datalog(KB) $\mathbf{FOL}_{\approx}$-emulates KB.[1]

DLP 2 is a strong requirement with many useful consequences. For example, it ensures that instance retrieval queries can directly be answered over datalog, without needing to know the details of the datalog transformation: to find out whether KB entails $C(a)$, it suffices to check if datalog(KB) entails $C(a)$. But DLP 2 is much stronger than the requirement of preserving such atomic consequences, since the entailment of any $\mathbf{FOL}_{\approx}$ formula over the signature of KB can be checked in datalog(KB).

The principles DLP 1 and DLP 2 set the stage for defining DLP but they do not yet provide sufficient details to attempt a definition. The description of DLP as the "intersection" of DL and datalog is not a useful basis for defining DLP: the syntactic intersection of the two formalisms contains no terminological axioms at all. This raises the question of how to define DLP in a canonical way. A naive approach would be to define a DL ontology to belong to DLP if it can be expressed by a semantically equivalent datalog program. Such a definition would be of little practical use: every inconsistent ontology can trivially be expressed in datalog, and therefore a DL reasoner is needed to decide whether or not a knowledge base should be considered to be in DLP. This is certainly undesirable from a practical viewpoint. It is therefore preferable to give a definition that can be checked without complex semantic computations:

DLP 3 (Tractability) Containment of a knowledge base KB in a DLP description logic should be decidable in polynomial time with respect to the size of KB.

Note that typical syntactic language definitions are often sub-polynomial, e.g. if they can be decided in logarithmic space (which leads to a linear-time algorithm that can be parallelised). Yet, polynomial-time language definitions might still be acceptable: for example, every decidable DL with transitive roles, number restrictions, and role hierarchies already requires polynomial time for computing a maximal set of simple roles.

[1]Recall that we use the first-order translation π to apply first-order notions such as semantic emulation to description logics; see Section 3.2.

The downside of a syntactic approach is that semantically equivalent transformations on a knowledge base may change its status with respect to DLP. This is not a new problem – many DLs are not syntactically closed under semantically equivalent transformations, e.g. due to simplicity restrictions – but it imposes an additional burden on ontology engineers and implementers. To alleviate this problem, a reasonable further design principle is to require closure under at least some forms of equivalence or satisfiability preserving transformations. Particularly common transformations are the constructions of negation normal form and disjunctive normal form as defined in Section 3.1.3.

DLP 4 (Closure Under NNF and DNF) A knowledge base KB should be in DLP if and only if its negation normal form $\mathsf{NNF}(\mathrm{KB})$ and its disjunctive normal form $\mathsf{DNF}(\mathrm{KB})$ are in DLP.

Closure under NNF will turn out to be mostly harmless, while closure under DNF imposes some real restrictions to our subsequent treatment. We still include it here since it allows us to generally present DL concepts as disjunctions, such that the relationship to datalog rules (disjunctions of literals) is more direct.

The above principles still allow DLP to be defined in such a way that some DLP knowledge base subsumes another knowledge base that is not in DLP. In other words, it might happen that adding axioms to a non-DLP knowledge base turns it into a DLP knowledge base. This "non-monotonic" behaviour is undesirable since it requires implementations and knowledge engineers to consider all axioms of a knowledge base in order to check if it is in DLP. The following principle requires definitions to be more well-behaved:

DLP 5 (Modularity) Consider two knowledge bases KB_1 and KB_2. Then $\mathrm{KB}_1 \cup \mathrm{KB}_2$ should be in DLP if and only if both KB_1 and KB_2 are. Moreover, in this case the datalog transformation should be $\mathsf{datalog}(\mathrm{KB}_1 \cup \mathrm{KB}_2) = \mathsf{datalog}(\mathrm{KB}_1) \cup \mathsf{datalog}(\mathrm{KB}_2)$.

Modularity ensures that one can decide for each axiom of a knowledge base whether or not it belongs to DLP without regarding any other axioms. The goal thus has changed from defining DLP *knowledge bases* to defining DLP *axioms*. Note that $\mathcal{SROIQ}$ with global constraints (regularity, simplicity) does not satisfy DLP 5 (to see this, set $\mathrm{KB}_1 = \{R \circ S \sqsubseteq R\}$ and $\mathrm{KB}_2 = \{R \circ S \sqsubseteq S\}$) which is the main reason for considering $\mathcal{SROIQ}^{\mathrm{free}}$ instead of $\mathcal{SROIQ}$ in this chapter. The above principles already suffice to establish an interesting result about tractability of reasoning in DLP:

Proposition 7.1.1 *Consider a class K of knowledge bases that belong to a DL for which DLP 1 to DLP 3, and DLP 5 are satisfied, and such that the maximal size of axioms in K is bounded. Then deciding satisfiability of knowledge bases in K is possible in polynomial time.*

Proof. By DLP 2, satisfiability of $\mathrm{KB} \in K$ can be decided by checking satisfiability of $\mathsf{datalog}(\mathrm{KB})$. Assume that the size of axioms in knowledge bases in K is at most n. Up to renaming of symbols, there is only a finite number of different axioms of size n. We can assume without loss of generality that the transformation $\mathsf{datalog}$ produces structurally similar datalog for structurally similar axioms, so that there are only a finite number of structurally different datalog theories $\mathsf{datalog}(\{\alpha\})$ that can be obtained from axioms α in K. The maximal number of variables occurring within these datalog programs is bounded by some m. By DLP 5, the same holds for all programs $\mathsf{datalog}(\mathrm{KB})$ with $\mathrm{KB} \in K$. Satisfiability of datalog with at most m variables per rule can be decided in time polynomial in 2^m (Fact 4.1.4). Since m is a constant, this yields a polynomial-time upper bound for deciding satisfiability of knowledge bases in K. □

We do not require DLP 4 in the previous result since no set of formulae that is closed under NNF can be restricted to axioms of a bounded size. Proposition 7.1.1 states that reasoning in any DLP language is necessarily “almost” tractable. Indeed, many DLs allow complex axioms to be decomposed into a number of simpler normal forms of bounded size, and in any such case tractability is obtained. It turns out that there are arbitrarily large DLP axioms that cannot be decomposed in DLP, yet Proposition 7.1.1 clarifies why Horn-$\mathcal{SHIQ}$ cannot be in DLP: it is not hard to modify the proof of Theorem 6.4.7 to establish ExpTime worst-case complexity of reasoning for a class K of Horn-$\mathcal{SHIQ}$ knowledge bases as in the above Proposition 7.1.1. Indeed, the knowledge bases constructed to show ExpTime hardness in Section 6.4.2 contain only two types of axioms that are not of bounded size: the tested GCI $I_w \sqsubseteq A$ and the universal acceptance axioms (4) of Fig. 6.8. Both include conjunctions with a linear number of conjuncts which can easily be decomposed by introducing a linear number of axioms of bounded size.

Note that none of the above principles actually require DLP to contain any knowledge base at all. An obvious approach thus is to define DLP to be the largest DL that adheres to all of the chosen design principles. The question to ask at this point is whether this is actually possible: is there a definition of DLP that adheres to the above principles and that includes as many DL ontologies as possible? The answer is a resounding no:

Proposition 7.1.2 *Consider a description logic $\mathcal{L}_{DLP}$ that adheres to the principles DLP 1 to DLP 5. There is a description logic $\mathcal{L}'_{DLP}$ that adheres to DLP 1 to DLP 5 while covering more knowledge bases, i.e. $\mathcal{L}_{DLP} \subset \mathcal{L}'_{DLP}$.*

Proof. We first need to argue that, even with unlimited resources for the datalog translation, it is not possible that DLP supports all $\mathcal{SROIQ}$ axioms. We show that, if the concept expression C is satisfiable and does not contain the symbols R, A_1, A_2, c and d, then the axiom $\alpha \coloneqq \{c\} \sqsubseteq C \sqcap \exists R.(d \sqcap (A_1 \sqcup A_2))$ cannot be $\mathbf{FOL}_{\approx}$-emulated by any datalog program. For a contradiction, suppose that α is $\mathbf{FOL}_{\approx}$-emulated by a datalog theory $\mathsf{datalog}(\alpha)$. By construction, α is satisfiable, and so is $\{\alpha, A_i \sqsubseteq \bot\}$ for each $i = 1, 2$. By Definition 2.2.1, we find that $\mathsf{datalog}(\alpha) \cup \{A_i \sqsubseteq \bot\}$ is satisfiable, too. Thus, there are models $\mathcal{I}_i$ of $\mathsf{datalog}(\alpha)$ such that $A_i^{\mathcal{I}_i} = \emptyset$. By the least model property of datalog, there is also a model $\mathcal{I}$ of $\mathsf{datalog}(\alpha)$ such that $A_1^{\mathcal{I}} = A_2^{\mathcal{I}} = \emptyset$. But then $\mathsf{datalog}(\alpha) \cup \{A_1 \sqcup A_2 \sqsubseteq \bot\}$ is satisfiable although $\{\alpha, A_1 \sqcup A_2 \sqsubseteq \bot\}$ is not, contradicting the supposed $\mathbf{FOL}_{\approx}$-emulation.

We can now show that there is some unsatisfiable axiom that is not in $\mathcal{L}_{DLP}$. To this end, recall that deciding (un)satisfiability of $\mathcal{SHOIQ}$ concept expressions is NExpTime hard. This follows from NExpTime hardness of deciding consistency of $\mathcal{SHOIQ}$ knowledge bases [Tob01] together with the fact that knowledge base satisfiability in $\mathcal{SROIQ}$ can be reduced to concept satisfiability [Sch94]. However, we just showed that, if the axiom $\alpha = \{c\} \sqsubseteq C \sqcap \exists R.(\{d\} \sqcap (A_1 \sqcup A_2))$ is in $\mathcal{L}_{DLP}$ with symbols R, A_1, A_2, c, d not in C, then the concept C is unsatisfiable. Thus, if $\mathcal{L}_{DLP}$ contains all unsatisfiable $\mathcal{SHOIQ}$ axioms of the form of α, then deciding whether $\alpha \in \mathcal{L}_{DLP}$ is equivalent to deciding whether C is unsatisfiable (since one can clearly construct α from C in polynomial time). By DLP 3, this would yield a polynomial decision procedure for $\mathcal{SHOIQ}$ concept satisfiability – a contradiction.

Therefore, there is an unsatisfiable axiom α with $\alpha \notin \mathcal{L}_{DLP}$. Now let $\mathcal{L}'_{DLP}$ be defined as $\{\text{KB} \mid \mathsf{DNF}(\mathsf{NNF}(\text{KB})) \setminus \{\mathsf{DNF}(\mathsf{NNF}(\alpha))\} \in \mathcal{L}_{DLP}\}$. The transformation is given by $\mathsf{datalog}'(\text{KB}) = \mathsf{datalog}(\text{KB})$ if $\text{KB} \in \mathcal{L}_{DLP}$, and $\mathsf{datalog}'(\text{KB}) = \{\top \to A(x), A(x) \to \bot\} \cup \mathsf{datalog}(\mathsf{DNF}(\mathsf{NNF}(\text{KB})) \setminus \{\mathsf{DNF}(\mathsf{NNF}(\alpha))\})$ otherwise, where A is a new predicate symbol. It is immediate that this defines a DL fragment (DLP 1), and that this definition is tractable (DLP 3). Equisatisfiability (DLP 2) follows since any knowledge base containing an axiom that is equivalent to α is unsatisfiable. Closure under negation normal form (DLP 4) and modularity (DLP 5) are immediate. □

This shows that any attempt to arrive at a maximal definition of DLP based on the above design principles must fail. Summing up, the above design principles are still too weak for characterising DLP: any concrete definition requires further choices that, lacking concrete guidelines, are necessarily somewhat arbitrary. Thus, while it is certainly useful to capture some general requirements in explicit principles, the resulting approach of defining DLP would not be a significant improvement over existing *ad hoc* approaches.

Analysing the proof of Proposition 7.1.2 reveals the reason why DLP 1 to

DLP 5 are still insufficient. Intuitively, a definition of DLP cannot reach the desired maximum since the computations that were required in this case would no longer be polynomial (DLP 3). Even DLP 5 does not ameliorate the situation, since expressive DLs can encode complex semantic relationships within single axioms. The core of the argument underlying Proposition 7.1.2 in this sense is the fact that there is no polynomial-time procedure for deciding whether or not a single $\mathcal{SROIQ}$ axiom can be expressed in datalog.

These considerations highlight a strategy for further constraining DLP to obtain a clearly defined canonical definition instead of infinitely many non-optimal choices. Namely, it is necessary to prevent complicated semantic effects that may arise when considering even single DL axioms from having any impact on the definition of DLP. Intuitively speaking, the reason for the high complexity of evaluating single axioms is that individual parts of an axiom, even if they are structurally separated, may semantically affect each other. In expressive DLs, individual parts of an axiom can capture the semantics of arbitrary terminological axioms: the TBox can be *internalised* into a single axiom.

An important observation now is that the semantic interplay of parts of an axiom usually requires entity names to be reused. For example, the axiom $\top \sqsubseteq A \sqcap \neg A$ is unsatisfiable because the concept name A is used in both conjuncts, while the structurally similar formula $\top \sqsubseteq A \sqcap \neg B$ is satisfiable. So, in order to disallow complex semantic effects within single axioms to affect DLP, we can require DLP to be closed under the exchange of entities in the following sense:

Definition 7.1.3 Let F be a $\mathbf{FOL}_{\approx}$ formula, a DL axiom, or a DL concept expression, and let $\mathscr{S}$ be a signature. An expression F' is a *renaming* of F in $\mathscr{S}$ if F' can be obtained from F by replacing each occurrence of a role/concept/individual name with some role/concept/individual name in $\mathscr{S}$. Multiple occurrences of the same entity name in F need *not* be replaced by the same entity name of $\mathscr{S}$ in this process.

A knowledge base KB′ is a *renaming* of a knowledge base KB if it is obtained from KB by replacing each axiom with a renaming. ◇

Note that we do not require all occurrences of an entity name to be renamed together, so it is indeed possible to obtain $A \sqcap \neg B$ from $A \sqcap \neg A$.

DLP 6 (Structurality) Consider knowledge bases KB and KB′ such that KB′ is an arbitrary renaming of KB. Then KB is in DLP iff KB′ is.

This is clearly a very strong requirement since it forces DLP to be based on the syntactic structure of axioms rather than on the semantic effects that occur for one particular axiom that has this structure. We will thus study the semantics and expressivity of formulae based on their syntactic structure, disregarding any possible interactions between signature symbols. We therefore call a $\mathbf{FOL}_{\approx}$ formula, DL axiom, or DL concept expression F *name-separated* if no signature symbols occur more than once in F.

Together with modularity (DLP 5), the principle of structurality captures the essential difference between a "syntactic" and a "semantic" transformation from DL to datalog. Indeed, if DLP adheres to DLP 5 and DLP 6, then it may only include knowledge bases for which all potential semantic effects can be faithfully represented in datalog. The datalog transformation thus needs to take into account that additional axioms may be added (DLP 5) to state that certain entity names are semantically equivalent, while hardly any semantic consequences can be computed in advance without knowing about these equivalences. In consequence, the semantic computations that determine satisfiability must be accomplished in datalog, and not during the translation. This intuition will turn out to be quite accurate – but a lot more is needed to establish formal results below.

Structurality also interacts with normal form transformations. For example, the concept $(\neg A \sqcup \neg B) \sqcap C$ can be emulated in datalog using rules $\top \rightarrow C(x)$ and $A(x) \wedge B(x) \rightarrow \bot$. But its DNF $(\neg A \sqcap C) \sqcup (\neg B \sqcap C)$ is only in DLP if its renaming $(\neg A \sqcap C) \sqcup (\neg B \sqcap D)$ is, which turns out to not be the case. Therefore, the knowledge base $\{\neg A \sqcup \neg B, C\}$ is in DLP but the knowledge base $\{(\neg A \sqcup \neg B) \sqcap C\}$ is not. We have discussed above why such effects are not avoidable in general. The more transformations are allowed for DLP, the less knowledge bases are contained in DLP. Note that such effects do not occur for negation normal forms.

7.2 The Datalog Fragment of $\mathcal{ALC}$

Our investigations in later sections show that the definition of a maximal DLP fragment of $\mathcal{SROIQ}^{\text{free}}$ is surprisingly complex, and the required proofs for showing its maximality are rather intricate. For this reason, we first characterise the maximal DLP fragment of the much simpler description logic $\mathcal{ALC}$. The absence of nominals and cardinality restrictions simplifies the required constructions significantly. Various basic aspects of the relationship between DL and datalog can also be found in this simpler case, but there are also a number of aspects that are not touched at all. To further simplify the syntactic presentation here, we also drop the requirement DLP 4 where especially closure under DNF otherwise leads to more complicated descriptions that do not serve the didactic purpose of this section.

Throughout this section, we use $\exists$ and $\forall$ instead of $\geqslant 1$ and $\leqslant 0 \ldots \neg$, which yields a more natural syntax for $\mathcal{ALC}$. Exploiting DLP 4 we can simplify the definition of DLP by giving concepts in negation normal form only.

Concepts that are necessarily equivalent to $\top$ and $\bot$
$\mathbf{L}^{\mathcal{A}}_{\top} ::= \top \mid \forall\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{\top} \mid \mathbf{L}^{\mathcal{A}}_{\top} \sqcap \mathbf{L}^{\mathcal{A}}_{\top} \mid \mathbf{L}^{\mathcal{A}}_{\top} \sqcup \mathbf{C}$ $\mathbf{L}^{\mathcal{A}}_{\bot} ::= \bot \mid \exists\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{\bot} \mid \mathbf{L}^{\mathcal{A}}_{\bot} \sqcap \mathbf{C} \mid \mathbf{L}^{\mathcal{A}}_{\bot} \sqcup \mathbf{L}^{\mathcal{A}}_{\bot}$
Body concepts: for C in normal form, $C \in \mathbf{L}^{\mathcal{A}}_{B}$ iff $C \sqcup A$ (or $\neg C \sqsubseteq A$) is in $\mathcal{DLP}_{\mathcal{ALC}}$
$\mathbf{L}^{\mathcal{A}}_{B} ::= \mathbf{L}^{\mathcal{A}}_{\top} \mid \mathbf{L}^{\mathcal{A}}_{\bot} \mid \neg\mathbf{A} \mid \forall\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{B} \mid \mathbf{L}^{\mathcal{A}}_{B} \sqcap \mathbf{L}^{\mathcal{A}}_{B} \mid \mathbf{L}^{\mathcal{A}}_{B} \sqcup \mathbf{L}^{\mathcal{A}}_{B}$
Head concepts: for C in normal form, $C \in \mathbf{L}^{\mathcal{A}}_{H}$ iff $A \sqsubseteq C$ is in $\mathcal{DLP}_{\mathcal{ALC}}$
$\mathbf{L}^{\mathcal{A}}_{H} ::= \mathbf{L}^{\mathcal{A}}_{B} \mid \mathbf{A} \mid \forall\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{H} \mid \mathbf{L}^{\mathcal{A}}_{H} \sqcap \mathbf{L}^{\mathcal{A}}_{H} \mid \mathbf{L}^{\mathcal{A}}_{H} \sqcup \mathbf{L}^{\mathcal{A}}_{B}$
Assertional concepts: for C in normal form, $C \in \mathbf{L}^{\mathcal{A}}_{a}$ iff $C(a)$ is in $\mathcal{DLP}_{\mathcal{ALC}}$
$\mathbf{L}^{\mathcal{A}}_{a} ::= \mathbf{L}^{\mathcal{A}}_{H} \mid \exists\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{a} \mid \mathbf{L}^{\mathcal{A}}_{a} \sqcap \mathbf{L}^{\mathcal{A}}_{a} \mid \mathbf{L}^{\mathcal{A}}_{a} \sqcup \mathbf{L}^{\mathcal{A}}_{B}$

Figure 7.1: Grammars of $\mathcal{DLP}_{\mathcal{ALC}}$ concepts in negation normal form, simplified as discussed in Section 3.1.3

Definition 7.2.1 We define the description logic $\mathcal{DLP}_{\mathcal{ALC}}$ to contain all knowledge bases consisting only of $\mathcal{SROIQ}^{\text{free}}$ axioms which are

- GCIs $C \sqsubseteq D$ such that $\mathsf{NNF}(\neg C \sqcup D)$ is an $\mathbf{L}^{\mathcal{A}}_{H}$ concept as defined in Fig. 7.1, or
- ABox axioms $C(a)$ where $\mathsf{NNF}(C)$ is an $\mathbf{L}^{\mathcal{A}}_{a}$ concept as defined in Fig. 7.1.

Note that the simplifications of Section 3.1.3 are used to specify concepts only up to associativity and commutativity of $\sqcap$ and $\sqcup$. $\diamond$

Intuitively speaking, the grammars $\mathbf{L}^{\mathcal{A}}_{H}$ and $\mathbf{L}^{\mathcal{A}}_{B}$ in Fig. 7.1 serve as "head" and "body" concepts of DLP, and hence play a similar rôle as the concepts $\mathbf{C_1}$ and $\mathbf{C_0}$ have played for Horn DLs in Section 6.1. Concepts of type $\mathbf{L}^{\mathcal{A}}_{a}$ account for cases where Skolemisation is admissible for emulating existential statements. Finally, the languages $\mathbf{L}^{\mathcal{A}}_{\top}$ and $\mathbf{L}^{\mathcal{A}}_{\bot}$ encompass concept expressions that are necessarily equivalent to $\top$ or $\bot$, even under arbitrary renamings.

Following the grammatical structure of $\mathcal{DLP}_{\mathcal{ALC}}$, we specify three auxiliary functions for constructing datalog programs to $\mathbf{FOL}_{\approx}$-emulate a $\mathcal{DLP}_{\mathcal{ALC}}$ knowledge base.

Lemma 7.2.2 *Given a concept name A, and a concept $C \in \mathbf{L}^{\mathcal{A}}_{H}$, Fig. 7.2 recursively defines a datalog program $\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq C)$ that semantically emulates $A \sqsubseteq C$.*

Proof. First note that the definition of $\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq C)$ is well. In particular, programs $\mathsf{dlg}^{\mathcal{A}}_{B}(\neg B \sqsubseteq D)$ are only used if $D \in \mathbf{L}^{\mathcal{A}}_{B}$. The claim is shown by induction over the definitions of $\mathsf{dlg}^{\mathcal{A}}_{B}(\neg A \sqsubseteq C)$ and $\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq C)$, where the hypothesis for the former is that it semantically emulates $\neg A \sqsubseteq C$. The easy induction steps can directly be established by showing that any model of the datalog program can be

C	$\mathsf{dlg}_H^{\mathcal{A}}(A \sqsubseteq C)$
$D \in \mathbf{L}_B^{\mathcal{A}}$	$\mathsf{dlg}_B^{\mathcal{A}}(\neg X \sqsubseteq D) \cup \{A(x) \wedge X(x) \to \bot\}$
B	$\{A(x) \to B(x)\}$
$\forall R.D$	$\mathsf{dlg}_H^{\mathcal{A}}(X \sqsubseteq D) \cup \{A(x) \wedge R(x,y) \to X(y)\}$
$D_1 \sqcap D_2$	$\mathsf{dlg}_H^{\mathcal{A}}(A \sqsubseteq D_1) \cup \mathsf{dlg}_H^{\mathcal{A}}(A \sqsubseteq D_2)$
$D_1 \sqcup D_2 \in (\mathbf{L}_H^{\mathcal{A}} \sqcup \mathbf{L}_B^{\mathcal{A}})$	$\mathsf{dlg}_H^{\mathcal{A}}(X_2 \sqsubseteq D_1) \cup \mathsf{dlg}_B^{\mathcal{A}}(\neg X_1 \sqsubseteq D_2) \cup \{A(x) \wedge X_1(x) \to X_2(x)\}$

C	$\mathsf{dlg}_B^{\mathcal{A}}(\neg A \sqsubseteq C)$
$D \in \mathbf{L}_\top^{\mathcal{A}}$	$\{\}$
$D \in \mathbf{L}_\bot^{\mathcal{A}}$	$\{A(x)\}$
$\neg B$	$\{B(x) \to A(x)\}$
$\forall R.D$	$\mathsf{dlg}_B^{\mathcal{A}}(\neg X \sqsubseteq D) \cup \{R(x,y) \wedge X(y) \to A(x)\}$
$D_1 \sqcap D_2 \in (\mathbf{L}_B^{\mathcal{A}} \sqcap \mathbf{L}_B^{\mathcal{A}})$	$\mathsf{dlg}_B^{\mathcal{A}}(\neg A \sqsubseteq D_1) \cup \mathsf{dlg}_B^{\mathcal{A}}(\neg A \sqsubseteq D_2)$
$D_1 \sqcup D_2 \in (\mathbf{L}_B^{\mathcal{A}} \sqcup \mathbf{L}_B^{\mathcal{A}})$	$\mathsf{dlg}_B^{\mathcal{A}}(\neg X_1 \sqsubseteq D_1) \cup \mathsf{dlg}_B^{\mathcal{A}}(\neg X_2 \sqsubseteq D_2) \cup \{X_1(x) \wedge X_2(x) \to A(x)\}$
A, B concept names, R a role, $X_{(i)}$ fresh concept names	

Figure 7.2: Transforming axioms $\mathbf{A} \sqsubseteq \mathbf{L}_H^{\mathcal{A}}$ and $\neg\mathbf{A} \sqsubseteq \mathbf{L}_B^{\mathcal{A}}$ to datalog

restricted to a model of the corresponding DL axiom, and any model of the DL axiom can be extended to an interpretation that models the datalog program. We omit further details here. Examples of a very similar argument are found in the proofs of Lemma 7.4.1 and 7.4.2. □

Lemma 7.2.3 *Given a constant a and a concept $C \in \mathbf{L}_a^{\mathcal{A}}$, Fig. 7.3 recursively defines a datalog program $\mathsf{dlg}_a^{\mathcal{A}}(C(a), \bot)$ that semantically emulates $C(a)$.*

Proof. The construction of Fig. 7.3 uses a "guard" concept E that is used to defer the encoding of $\mathbf{L}_B^{\mathcal{A}}$ disjunctions. The induction claim thus is that, for every $E \in \mathbf{L}_B^{\mathcal{A}}$, $C \in \mathbf{L}_a^{\mathcal{A}}$, and $a \in \mathbf{I}$, the program $\mathsf{dlg}_H^{\mathcal{A}}(C(a), E)$ semantically emulates $(C \sqcup E)(a)$.

The concept E is processed in case $C \in \mathbf{L}_H^{\mathcal{A}}$ by using $\mathsf{dlg}_H^{\mathcal{A}}$. Another more interesting case is $C = \exists R.D$. The basic encoding works by standard Skolemisation, but the guard concept is also processed and a new guard $\neg Y$ is created for the Skolem constant d. It is not hard to show semantic emulation in all cases and we omit further details and refer to the full proofs given in Section 7.4. □

We summarise these results in the emulation theorem for $\mathcal{DLP}_{\mathcal{ALC}}$.

C	$\mathsf{dlg}_a^{\mathcal{A}}(C(a), E)$
$D \in \mathbf{L}_H^{\mathcal{A}}$	$\mathsf{dlg}_H^{\mathcal{A}}(X \sqsubseteq D \sqcup E) \cup \{X(a)\}$
$D_1 \sqcap D_2$	$\mathsf{dlg}_a^{\mathcal{A}}(D_1(a), E) \cup \mathsf{dlg}_a^{\mathcal{A}}(D_2(a), E)$
$D_1 \sqcup D_2 \in (\mathbf{L}_a^{\mathcal{A}} \sqcup \mathbf{L}_B^{\mathcal{A}})$	$\mathsf{dlg}_B^{\mathcal{A}}(\neg X \sqsubseteq D_2) \cup \mathsf{dlg}_a^{\mathcal{A}}(D_1(a), E \sqcup \neg X)$
$\exists R.D$	$\mathsf{dlg}_B^{\mathcal{A}}(\neg X \sqsubseteq E) \cup \{X(a) \to R(a,b), X(a) \to Y(b)\} \cup \mathsf{dlg}_a^{\mathcal{A}}(D(b), \neg Y)$
$E \in \mathbf{L}_B^{\mathcal{A}}$, X, Y fresh concept names, b a fresh constant	

Figure 7.3: Transforming axioms $C(a)$ with $C \in \mathbf{L}_a^{\mathcal{A}}$ to datalog

Theorem 7.2.4 *For every $\mathcal{DLP}_{\mathcal{ALC}}$ axiom α as in Definition 7.2.1, one can construct a datalog program* $\mathsf{datalog}(\alpha)$ *that semantically emulates α.*

Proof. If $\alpha = C \sqsubseteq D$ is a TBox axiom, define $\mathsf{datalog}(\alpha) := \mathsf{dlg}_H^{\mathcal{A}}(A \sqsubseteq \mathsf{NNF}(\neg C \sqcup D)) \cup \{A(x)\}$. If $\alpha = C(a)$ is an ABox axiom, define $\mathsf{datalog}(\alpha) := \mathsf{dlg}_a^{\mathcal{A}}(C(a), \bot)$. The result follows by Lemma 7.2.2 and 7.2.3. □

It remains to show that $\mathcal{DLP}_{\mathcal{ALC}}$ is indeed the largest DLP fragment of $\mathcal{ALC}$. We first define auxiliary datalog programs to entail that a concept's extension is empty for arbitrary concepts that are not in $\mathbf{L}_\top^{\mathcal{A}}$.

Definition 7.2.5 Given a name-separated concept $C \notin \mathbf{L}_\top^{\mathcal{A}}$, a datalog program $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}}$ is recursively defined as follows:

- If $C = \bot$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} := \{\}$.
- If $C \in \mathbf{A}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} := \{C(x) \to \bot\}$.
- If $C = \neg B \in \neg\mathbf{A}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} := \{B(x)\}$.
- If $C = \forall R.D$ with $D \notin \mathbf{L}_\top^{\mathcal{A}}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} := \{R(x,x)\} \cup [\![D \sqsubseteq \bot]\!]_{\mathcal{A}}$.
- If $C = \exists R.D$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} := \{R(x,y) \to \bot\}$.
- If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{L}_\top^{\mathcal{A}}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} := [\![D_1 \sqsubseteq \bot]\!]_{\mathcal{A}}$.
- If $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{L}_\top^{\mathcal{A}}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} := [\![D_1 \sqsubseteq \bot]\!]_{\mathcal{A}} \cup [\![D_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$.

Given a name-separated concept $C \notin \mathbf{L}_\bot^{\mathcal{A}}$, a datalog program $[\![\top \sqsubseteq C]\!]_{\mathcal{A}}$ is defined as $[\![\top \sqsubseteq C]\!]_{\mathcal{A}} := [\![\mathsf{NNF}(\neg C) \sqsubseteq \bot]\!]_{\mathcal{A}}$. ◇

Note that this definition is well. In particular, observe that $C \notin \mathbf{L}_\bot^{\mathcal{A}}$ implies $\mathsf{NNF}(\neg C) \notin \mathbf{L}_\top^{\mathcal{A}}$. Moreover, it is easy to see that $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}}$ ($[\![\top \sqsubseteq C]\!]_{\mathcal{A}}$) is satisfiable and entails $C \sqsubseteq \bot$ ($\top \sqsubseteq C$).

The next lemma shows that concepts that are not in $\mathbf{L}_B^{\mathcal{A}}$ can be forced to require certain positive entailments to hold in any model in which they have a non-empty extension.

Lemma 7.2.6 *If $C \notin \mathbf{L}^{\mathcal{A}}_{B}$ is name-separated then there is a datalog program $[\![C \sqsubseteq A]\!]_{\mathcal{A}}$ for a fresh concept name A such that*

- $[\![C \sqsubseteq A]\!]_{\mathcal{A}} \cup \{C(a)\}$ *is satisfiable for any individual name a, and*
- $[\![C \sqsubseteq A]\!]_{\mathcal{A}} \models C \sqsubseteq A$.

Proof. The result is shown by induction over the structure of C. If $C \in \mathbf{A}$ is a concept name, then $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := \{C(x) \to A(x)\}$ clearly satisfies the claim. If $C = \forall R.D$ with $D \notin \mathbf{L}^{\mathcal{A}}_{B}$ set $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := [\![D \sqsubseteq A]\!]_{\mathcal{A}} \cup \{R(x,x)\}$. The claim follows by induction. If $C = \exists R.D$ with $D \neq \bot$ then $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := \{R(x,y) \to A(x)\}$ clearly satisfies the claim. If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{L}^{\mathcal{A}}_{B}$, $D_1, D_2 \notin \mathbf{L}^{\mathcal{A}}_{\bot}$ then $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := [\![D_1 \sqsubseteq A]\!]_{\mathcal{A}}$ satisfies the claim by the induction hypothesis. For the case $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{L}^{\mathcal{A}}_{B}$ and $D_1, D_2 \notin \mathbf{L}^{\mathcal{A}}_{\top}$, we can define $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := [\![D_1 \sqsubseteq A]\!]_{\mathcal{A}} \cup [\![D_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$. The claim follows by induction. □

Note that the program $[\![C \sqsubseteq A]\!]_{\mathcal{A}}$ does not $\mathbf{FOL}_{\approx}$-emulate $C \sqsubseteq A$ since the subprogram $[\![D_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$ that is used for the $\sqcup$ case excludes a number of interpretations that satisfy C. But the previous result suffices for our subsequent arguments.

Theorem 7.2.7 *Consider a name-separated concept C, an individual name a, and a concept name A not occuring in C.*

(1) If $C \notin \mathbf{L}^{\mathcal{A}}_{a}$ then $C(a)$ cannot be $\mathbf{FOL}_{\approx}$*-emulated by any datalog program.*

(2) If $C \notin \mathbf{L}^{\mathcal{A}}_{H}$ then $A \sqsubseteq C$ and $\top \sqsubseteq C$ cannot be $\mathbf{FOL}_{\approx}$*-emulated by any datalog program, unless* P = PSpace.

In particular, no fragment of $\mathcal{ALC}$ that is larger than $\mathcal{DLP}_{\mathcal{ALC}}$ can be $\mathbf{FOL}_{\approx}$*-emulated by datalog, unless* P = PSpace.

Proof. The proof for both claims proceeds by an interleaved induction over the structure of C. Note that C cannot be atomic in either case. We begin with the induction steps for claim (1), assuming that the claims hold for all subformulae of C. Suppose for a contradiction that there is a datalog program $P_{C(a)}$ that $\mathbf{FOL}_{\approx}$-emulates $C(a)$.

If $C = \exists R.D$ with $D \notin \mathbf{L}^{\mathcal{A}}_{a}$ then $P_{C(a)} \cup \{R(a,y) \to y \approx b\}$ $\mathbf{FOL}_{\approx}$-emulates $D(b)$ for a fresh individual b, contradicting the induction hypothesis (1) for D. If $C = \forall R.D$ with $D \notin \mathbf{L}^{\mathcal{A}}_{H}$ then $P_{C(a)} \cup \{A(x) \to R(a,x)\}$ $\mathbf{FOL}_{\approx}$-emulates $A \sqsubseteq D$, contradicting the induction hypothesis (2) for D. If $C = C_1 \sqcap C_2$ with $C_1 \notin \mathbf{L}^{\mathcal{A}}_{a}$ and $C_1, C_2 \notin \mathbf{L}^{\mathcal{A}}_{\bot}$ then $P_{C(a)} \cup [\![\top \sqsubseteq C_2]\!]_{\mathcal{A}}$ $\mathbf{FOL}_{\approx}$-emulates $C_1(a)$, contradicting the induction hypothesis (1) for C_1.

Consider the case $C = C_1 \sqcup C_2$ where $C_1, C_2 \notin \mathbf{L}^{\mathcal{A}}_{\top}$. If $C_1 \notin \mathbf{L}^{\mathcal{A}}_{a}$ then $P_{C(a)} \cup [\![C_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$ $\mathbf{FOL}_{\approx}$-emulates $C_1(a)$, again contradicting the induction hypothesis

(1) for C_1. Otherwise, if $C_1, C_2 \in \mathbf{L}_a^{\mathcal{A}}$ then $C_1, C_2 \notin \mathbf{L}_B^{\mathcal{A}}$. Using fresh concept names A_1 and A_2, consider datalog programs $P_i \coloneqq \{A_i(x) \to \bot\} \cup [\![C_1 \sqsubseteq A_1]\!]_{\mathcal{A}} \cup [\![C_2 \sqsubseteq A_2]\!]_{\mathcal{A}}$ $(i = 1, 2)$. It is not hard to see that $\{C(a)\} \cup P_i$ is satisfiable, so the same is true for $P_{C(a)} \cup P_i$ by $\mathbf{FOL}_{\approx}$-emulation. Thus, $P_{C(a)} \cup [\![C_1 \sqsubseteq A_1]\!]_{\mathcal{A}} \cup [\![C_2 \sqsubseteq A_2]\!]_{\mathcal{A}}$ must have a model $\mathcal{I}_i$ such that $A_i^{\mathcal{I}_i} = \emptyset$ for $i = 1, 2$. By the least model property of datalog (see, e.g., [DEGV01]), this implies that $P_{C(a)} \cup [\![C_1 \sqsubseteq A_1]\!]_{\mathcal{A}} \cup [\![C_2 \sqsubseteq A_2]\!]_{\mathcal{A}}$ has a model $\mathcal{I}$ such that $A_1^{\mathcal{I}} = A_2^{\mathcal{I}} = \emptyset$. Thus $P_{C(a)} \cup P_1 \cup P_2$ is satisfiable. But clearly $P_i \models C_i \sqsubseteq \bot$ $(i = 1, 2)$ so $\{C(a)\} \cup P_1 \cup P_2$ is unsatisfiable, contradicting the supposed $\mathbf{FOL}_{\approx}$-emulation.

This finishes the induction steps for claim (1). For claim (2), suppose for a contradiction that $A \sqsubseteq C$ is $\mathbf{FOL}_{\approx}$-emulated by some datalog program $P_{A \sqsubseteq C}$. First consider the case that $C \notin \mathbf{L}_a^{\mathcal{A}}$. Then $P_{A \sqsubseteq C} \cup \{A(a)\}$ $\mathbf{FOL}_{\approx}$-emulates $C(a)$ for some fresh individual a, contradicting the induction hypothesis (1) for C. Thus, the remaining induction steps only need to cover the cases of $C \in \mathbf{L}_a^{\mathcal{A}} \setminus \mathbf{L}_H^{\mathcal{A}}$.

The case for $C = C_1 \sqcap C_2$ is similar to step (1). Likewise, the only remaining case of $C = C_1 \sqcup C_2$ is the case where, w.l.o.g., $C_1 \mathbf{L}_a^{\mathcal{A}} \setminus \mathbf{L}_H^{\mathcal{A}}$, which can also be treated as before. There are no remaining cases for $C = \forall R.D$.

Consider the case $C = \exists R.D$ with $D \notin \mathbf{L}_\bot^{\mathcal{A}}$. Then $P_{A \sqsubseteq C} \cup [\![\top \sqsubseteq D]\!]_{\mathcal{A}}$ $\mathbf{FOL}_{\approx}$-emulates $A \sqsubseteq \exists R.\top$. The logic obtained by extending $\mathcal{DLP}_{\mathcal{ALC}}$ with axioms of the form $A \sqsubseteq \exists R.\top$ is Horn-$\mathcal{FL}^-$ for which reasoning was shown to be PSpace-hard in Section 6.3.1. It is not hard to modify the proof of the according Lemma 6.3.5 to use only axioms of bounded size. Assuming that P $\neq$ PSpace the supposed $\mathbf{FOL}_{\approx}$-emulation contradicts Proposition 7.1.1. □

Our subsequent results for the maximal DLP fragment of $\mathcal{SROIQ}^{\text{free}}$ further strengthen the previous theorem so that the assumption P $\neq$ PSpace is no longer required. We thus do not invest any more effort to accomplish this for the above case. Another direct proof of this result is given in [KRS10].

7.3 Defining Description Logic Programs

In this section, we provide a direct definition of DLP as a fragment of $\mathcal{SROIQ}^{\text{free}}$, where we assume that the universal role U is *not* a special role operator but rather is introduced *a posteriori* by suitable axiomatisation. Adding U is not a problem in principle, but based on the discussions in the next sections it will become obvious that this would further complicate our presentation substantially. Our motivation for considering $\mathcal{SROIQ}$ is to cover the essential features that were considered for DLP and OWL 2 RL, neither of which includes the universal role.

We first summarise the characterisation of DLP as presented in Section 7.1.

Definition 7.3.1 A description logic $\mathcal{L}$ is a *DLP description logic* if the set of its knowledge bases adheres to the principles DLP 1–DLP 6 of Section 7.1. ◇

Our goal in this section thus is to define the maximal DLP description logic. Some further considerations are needed for this to become practically feasible. Namely, it turns out that the characterisation as given in the previous section leads to a prohibitively complex syntactic description of the language. Our first goal in this section therefore is to identify ways of simplifying its presentation. Note that it is not desirable to simply eliminate "syntactic sugar" in general, since the very goal of this work is to characterise which $\mathcal{SROIQ}$ knowledge bases can be considered as syntactic sugar for datalog.

A natural approach is to restrict attention to axioms in some normal form. DLP 4 requires closure under negation normal form, which seems to free us from the burden of explicitly considering negative occurrences of non-atomic concepts. But NNF does not allow for this simplification, since concepts of the form $\leqslant n\, R.D$ still contain D in negative polarity. Hence, the positive negation normal form of Definition 3.1.10 is more adequate for our purposes.

While pNNF effectively reduces the size of a DLP definition by half, the definition is still exceedingly complex. The construction of disjunctive normal forms is compatible with pNNF, so we can additionally require this form of normalisation. Another source of complexity is the fact that $\mathcal{SROIQ}$ features many concept expressions for which all possible renamings are necessarily equivalent to $\top$ or $\bot$. Simple examples such as $\top \sqcup C$ were already encountered in the definitions of $\mathbf{L}_{\top}^{\mathcal{A}}$ and $\mathbf{L}_{\bot}^{\mathcal{A}}$ in Section 7.2, but $\mathcal{SROIQ}$ also includes expressions like $\geqslant 0\, R.C$ or $\leqslant 3\, R.\{a\} \sqcup \{b\}$.

Definition 7.3.2 Let C be a $\mathcal{SROIQ}$ concept expression.

- C is *structurally valid* if $\top \sqsubseteq C'$ is valid for every renaming C' of C.
- C is *structurally unsatisfiable* if $C' \sqsubseteq \bot$ is valid for every renaming C' of C.
- C is *structurally refutable* if it is not structurally valid, i.e. if there is a renaming C' of C such that $\top \sqsubseteq C'$ is refutable.
- C is *structurally satisfiable* if it is not structurally unsatisfiable, i.e. if there is a renaming C' of C such that $C' \sqsubseteq \bot$ is refutable.

The renamings C' considered here refer to renamings over arbitrary signatures, and are not restricted to the signature of C. ◇

Many non-trivial examples for such concepts are based on the fact that some DL concepts do not allow for arbitrary interpretations but are in fact constrained to certain extensions. It is possible to provide a complete syntactic characterisation of these $\mathcal{SROIQ}$ concepts.

Concepts containing at most n elements in any interpretation, and their complements

$$\begin{aligned}
\mathbf{L}_{\bot} = \mathbf{L}_{\leq 0} &::= \bot \mid \mathbf{L}_{\bot} \sqcap \mathbf{C} \mid \mathbf{L}_{\bot} \sqcup \mathbf{L}_{\bot} \mid \geqslant n\,\mathbf{R}.\mathbf{L}_{\leq n-1}\ (n \geq 1) \\
\mathbf{L}_{\leq m+1} &::= \{\mathbf{I}\} \mid \mathbf{L}_{\leq m} \mid \mathbf{L}_{\leq m+1} \sqcap \mathbf{C} \mid \mathbf{L}_{\leq m'} \sqcup \mathbf{L}_{\leq m''}\ (m' + m'' = m + 1) \\
\overline{\mathbf{L}}_{\bot} = \overline{\mathbf{L}}_{\leq 0} &::= \top \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \overline{\mathbf{L}}_{\bot} \sqcap \overline{\mathbf{L}}_{\bot} \mid \overline{\mathbf{L}}_{\bot} \sqcup \mathbf{C} \mid \\
&\qquad \leqslant n\,\mathbf{R}.\neg\mathbf{C}\ (n \geq 0) \mid \geqslant 0\,\mathbf{R}.\mathbf{C} \mid \geqslant n\,\mathbf{R}.\overline{\mathbf{L}}_{\leq n-1}\ (n \geq 1) \\
\overline{\mathbf{L}}_{\leq m+1} &::= \top \mid \mathbf{A} \mid \exists\mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \\
&\qquad \overline{\mathbf{L}}_{\leq m+1} \sqcap \overline{\mathbf{L}}_{\leq m+1} \mid \overline{\mathbf{L}}_{\leq m+1} \sqcup \mathbf{C} \mid \overline{\mathbf{L}}_{\leq m'} \sqcup \overline{\mathbf{L}}_{\leq m''}\ (m' + m'' = m) \mid \\
&\qquad \leqslant n\,\mathbf{R}.\neg\mathbf{C}\ (n \geq 0) \mid \geqslant 0\,\mathbf{R}.\mathbf{C} \mid \geqslant n\,\mathbf{R}.\overline{\mathbf{L}}_{\leq n-1}\ (n \geq 1)
\end{aligned}$$

Concepts not containing at most n elements in any interpretation; their complements

$$\begin{aligned}
\mathbf{L}_{\top} = \mathbf{L}_{\geq \omega-0} &::= \top \mid \mathbf{L}_{\top} \sqcup \mathbf{C} \mid \mathbf{L}_{\top} \sqcap \mathbf{L}_{\top} \mid \geqslant 0\,\mathbf{R}.\mathbf{C} \mid \leqslant n\,\mathbf{R}.\neg\mathbf{L}_{\geq \omega-n}\ (n \geq 0) \\
\mathbf{L}_{\geq \omega-m-1} &::= \neg\{\mathbf{I}\} \mid \mathbf{L}_{\geq \omega-m} \mid \mathbf{L}_{\geq \omega-m-1} \sqcup \mathbf{C} \mid \mathbf{L}_{\geq \omega-m'} \sqcap \mathbf{L}_{\geq \omega-m''}\ (m' + m'' = m + 1) \\
\overline{\mathbf{L}}_{\top} = \overline{\mathbf{L}}_{\geq \omega-0} &::= \bot \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \overline{\mathbf{L}}_{\top} \sqcup \overline{\mathbf{L}}_{\top} \mid \overline{\mathbf{L}}_{\top} \sqcap \mathbf{C} \mid \\
&\qquad \geqslant n\,\mathbf{R}.\mathbf{C}\ (n \geq 1) \mid \leqslant n\,\mathbf{R}.\neg\overline{\mathbf{L}}_{\geq \omega-n}\ (n \geq 0) \\
\overline{\mathbf{L}}_{\geq \omega-m-1} &::= \bot \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \\
&\qquad \overline{\mathbf{L}}_{\geq \omega-m-1} \sqcup \overline{\mathbf{L}}_{\geq \omega-m-1} \mid \overline{\mathbf{L}}_{\geq \omega-m-1} \sqcap \mathbf{C} \mid \overline{\mathbf{L}}_{\geq \omega-m'} \sqcap \overline{\mathbf{L}}_{\geq \omega-m''}\ (m'+m'' = m) \mid \\
&\qquad \geqslant n\,\mathbf{R}.\mathbf{C}\ (n \geq 1) \mid \leqslant n\,\mathbf{R}.\neg\overline{\mathbf{L}}_{\geq \omega-n}\ (n \geq 0)
\end{aligned}$$

C: any $\mathcal{SROIQ}^{\text{free}}$ concept

Figure 7.4: Grammars of structurally valid, unsatisfiable, refutable, and satisfiable concepts

Lemma 7.3.3 *The grammars given in Fig. 7.4 characterise sets of $\mathcal{SROIQ}$ concept expressions as follows:*

- *$C \in \mathbf{L}_{\leq n}$ iff $C^{\mathcal{I}}$ contains at most n elements for any interpretation $\mathcal{I}$,*
- *$C \in \overline{\mathbf{L}}_{\leq n}$ iff $C^{\mathcal{I}}$ contains more than n elements for some interpretation $\mathcal{I}$,*
- *$C \in \mathbf{L}_{\geq \omega-n}$ iff $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ contains at most n elements for any interpretation $\mathcal{I}$,*
- *$C \in \overline{\mathbf{L}}_{\geq \omega-n}$ iff $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ contains more than n elements for some interpretation $\mathcal{I}$.*

In particular, $\mathbf{L}_{\top}$, $\mathbf{L}_{\bot}$, $\overline{\mathbf{L}}_{\top}$, and $\overline{\mathbf{L}}_{\bot}$ characterise the sets of structurally valid, unsatisfiable, refutable, or satisfiable concept expressions.

Proof. We first show the "only if" direction of $\mathbf{L}_{\leq n}$ by induction over the structure of the grammars. The base cases $\bot$ and $\{\mathbf{I}\}$ (where $n \geq 1$ is required) are obvious. The case $\mathbf{L}_{\leq n-1}$ (where $n \geq 1$) is immediate from the induction hypothesis. Note that the cases of $\sqcup$ and $\sqcap$ for $n = 0$ are simply special instances of the respective cases for $n \geq 1$. The cases for $\mathbf{L}_{\leq n} \sqcap \mathbf{C}$ and $\mathbf{L}_{\leq m'} \sqcap \mathbf{L}_{\leq m''}$ are again obvious from the induction hypothesis.

Considering the grammar for each operator, it can be seen that $\overline{\mathbf{L}}_{\leq n}$ is indeed the set complement of $\mathbf{L}_{\leq n}$ for each n. An easy induction over n is used to show this formally, where it suffices to compare the cases for each constructor to see that they are exhaustive and non-overlapping. Thus, to show the "if" direction of the claim for $\mathbf{L}_{\leq n}$, it suffices to show the "only if" direction of the claim for $\overline{\mathbf{L}}_{\leq n}$.

The "only if" direction of the claim for $\overline{\mathbf{L}}_{\leq n}$ if again established induction over the structure of concepts in $\mathbf{L}_{\leq n}$. Most cases are obvious. For the case of $C \sqcap D$, it is necessary to note that the extensions of C and D, in addition to containing more than n elements, can always be selected freely to ensure that the intersection of both extensions contains enough elements.

The proofs for the claims about $\mathbf{L}_{\geq \omega - n}$ and $\overline{\mathbf{L}}_{\geq \omega - n}$ are similar. □

The previous result shows that structural validity, satisfiability, unsatisfiability, and refutability of a concept expression can be recognised in polynomial time by using the given grammars.[2] For another simplification of our characterisation, we may thus assume that almost all occurrences of such concepts have been eliminated in the concepts that we consider. This completes the ingredients we need for defining the normal form that is used below.

Definition 7.3.4 A concept C is in *DLP normal form* if $C = \mathsf{DNF}(\mathsf{pNNF}(C))$ and

- if C has a structurally valid subconcept D, then $D = \top$ and either $C = D$ or D occurs in a subconcept of the form $\geqslant n\, R.D$,
- if C has a structurally unsatisfiable subconcept D, then $D = \bot$ and either $C = D$ or D occurs in a subconcept of the form $\leqslant n\, R.\neg D$.

The unique DLP normal form of a concept D is denoted by $\mathsf{DLPNF}(C)$. ◇

It is easy to see that $\mathsf{DLPNF}(C)$ can be computed in polynomial time. In particular, structurally valid and unsatisfiable subconcepts can be replaced by $\top$ and $\bot$, respectively, and expressions of the form $C \sqcup \bot$ and $C \sqcap \top$ can be reduced to C. Also note that the order of applying the single normalisation steps does not affect the DLP normal form. It therefore suffices to characterise concepts in DLP normal form that are in a DLP description logic. When convenient, we continue to use GCIs $C \sqsubseteq D$ to represent the unique DLP normal form of $\neg C \sqcup D$. Exploiting associativity and commutativity of $\sqcap$ and of $\sqcup$, we furthermore disregard order and nesting of multiple conjunctions or disjunctions.

Whereas structurally valid and invalid subconcepts are ignored in DLP normal forms, we still have reason to consider concepts with restricted extensions. We

[2]Note that the omission of the universal role allows us to ignore concepts such as $\leqslant 0\, U.\{a\}$ which would otherwise be structurally unsatisfiable; similar simplifications occur throughout this section.

thus use $\mathbf{D}_{\leq n}$ ($\mathbf{D}_{\geq \omega - n}$) to denote the sublanguage of concepts of $\mathbf{L}_{\leq n}$ ($\mathbf{L}_{\geq \omega - n}$) that are in DLP normal form.

Before giving the full definition of a large – actually, as we will show below, *the largest* – DLP description logic, we provide some examples to sketch the complexities of this endeavour (datalog emulations are provided in parentheses). DLP expressions of the form $A \sqcap \exists R.B \sqsubseteq \forall S.C$ ($A(x) \wedge R(x,y) \wedge B(y) \wedge S(x,z) \to C(z)$) are well-known. The same is true for $A \sqsubseteq \exists R.\{c\}$ ($A(x) \to R(x,c)$) but hardly for $A \sqsubseteq \geqslant 2\, R.(\{c\} \sqcup \{d\})$ ($A(x) \to R(x,c)$, $A(x) \to R(x,d)$, $c \approx d \to \bot$). Another unusual form of DLP axioms arises when Skolem constants (not functions) can be used, as in the case $\{c\} \sqsubseteq \geqslant 2\, R.A$ ($R(c,s)$, $R(c,s')$, $A(s)$, $A(s')$, $s \approx s' \to \bot$ with fresh s, s') and $A \sqsubseteq \exists R.(\{c\} \sqcap \exists S.\top)$ ($A(x) \to R(x,c)$, $A(x) \to S(c,s)$ with fresh s). Besides these simple cases, there are various DLP axioms for which the emulation in datalog is significantly more complicated, typically requiring an exponential number of rules. Examples are $\{c\} \sqsubseteq \geqslant 2\, R.(\neg\{a\} \sqcup A \sqcup B)$ and $\{c\} \sqsubseteq \geqslant 5\, R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leqslant 1\, S.(\{c\} \sqcup \{d\})))$. These cases are based on the complex semantic interactions between nominals and atleast-restrictions.

Definition 7.3.5 We define the description logic $\mathcal{DLP}$ to contain all knowledge bases consisting only of $\mathcal{SROIQ}^{\text{free}}$ axioms which are

- RBox axioms, or
- GCIs $C \sqsubseteq D$ such that the DLP normal form of $\neg C \sqcup D$ is a $\mathbf{D}_{\text{DLP}}$ concept as defined in the following grammar:

$$\mathbf{D}_{\text{DLP}} ::= \top \mid \bot \mid \mathbf{C}_H \mid \mathbf{D}^{=n}\ (n \geq 1) \mid \mathbf{C}_{\neq\top}$$

  where $\mathbf{C}_H$ is defined as in Fig. 7.5, and $\mathbf{D}^{=n}$ and $\mathbf{C}_{\neq\top}$ are defined as in Fig. 7.6, or
- ABox axioms $C(a)$ where the DLP normal form of C is $\top$, $\bot$, or a $\mathbf{D}_a$ concept as defined in Fig. 7.5.

As before, the simplifications of Section 3.1.3 are used to omit $\exists$ and $\forall$, and to specify concepts only up to associativity and commutativity of $\sqcap$ and $\sqcup$. ◇

In spite of the immense simplifications that DLP normal form provides, the definition of $\mathcal{DLP}$ still turns out to be extremely complex. We have not succeeded in simplifying the presentation any further without loosing substantial expressive features. Some intuitive explanations help to understand the underlying ideas. It is instructive to also compare these intuitions to the above examples.

The core language elements are in Fig. 7.5. Since all concepts are in DNF, each sublanguage consists of a conjunctive part $\mathbf{C}$ and a disjunctive part $\mathbf{D}$. Definitions of DLP typically distinguish between "head" and "body" concepts, and $\mathbf{C}_H$ and

Body concepts: for C in normal form, $C \in \mathbf{D}_B$ iff $C \sqcup A$ (or $\neg C \sqsubseteq A$) is in DLP

$$\begin{aligned}
\mathbf{C}_B &::= \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \leqslant 0\,\mathbf{R}.\neg(\mathbf{D}_B \cup \{\bot\}) \mid \mathbf{C}_B \sqcap \mathbf{C}_B \\
\mathbf{D}_B &::= \mathbf{C}_B \mid \mathbf{D}_B \sqcup \mathbf{D}_B
\end{aligned}$$

Head concepts: for C in normal form, $C \in \mathbf{D}_H$ iff $A \sqsubseteq C$ is in DLP

$$\begin{aligned}
\mathbf{C}_H &::= \mathbf{C}_B \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \geqslant n\,\mathbf{R}.\mathbf{D}_{n!} \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}_H \mid \leqslant 1\,\mathbf{R}.\neg(\mathbf{D}_B \cup \{\bot\}) \mid \\
&\qquad \mathbf{C}_H \sqcap \mathbf{C}_H \mid \mathbf{D}_{1!} \\
\mathbf{D}_H &::= \mathbf{C}_H \mid \mathbf{D}_H \sqcup \mathbf{D}_B \mid \mathbf{D}_a \sqcup \mathbf{C}_{\geq}
\end{aligned}$$

Assertional concepts: for C in normal form, $C \in \mathbf{D}_a$ iff $\{a\} \sqsubseteq C$ is in DLP

$$\begin{aligned}
\mathbf{C}_a &::= \mathbf{C}_H \mid \geqslant n\,\mathbf{R}.\mathbf{D}^{\geqslant n} \mid \mathbf{C}_a \sqcap \mathbf{C}_a \\
\mathbf{D}_a &::= \mathbf{C}_a \mid \mathbf{D}_a \sqcup \mathbf{D}_B
\end{aligned}$$

Disjunctions of nominal assertions of the form $\{\mathbf{I}\} \sqcap \mathbf{C}_a$

$$\begin{aligned}
\mathbf{D}_{1!} &::= \{\mathbf{I}\} \mid \{\mathbf{I}\} \sqcap \mathbf{C}_a \\
\mathbf{D}_{m+1!} &::= \mathbf{D}_{m!} \sqcup \mathbf{D}_{1!}
\end{aligned}$$

Conjunction of negated nominals, i.e. complements of some nominal disjunction

$$\begin{aligned}
\mathbf{C}_{\neg 1} &::= \neg\{\mathbf{I}\} \\
\mathbf{C}_{\neg(m+1)} &::= \mathbf{C}_{\neg m} \sqcap \neg\{\mathbf{I}\} \\
\mathbf{C}_{\geq} &::= \neg\{\mathbf{I}\} \mid \mathbf{C}_{\geq} \sqcap \mathbf{C}_{\geq}
\end{aligned}$$

Filler concepts for $\geqslant n$ in $\mathbf{D}_a$

$$\begin{aligned}
\mathbf{D}^{\geqslant n} &::= \top \mid \mathbf{C}_{\neg m} \sqcup \mathbf{D}^+_a \;\; (1 \leq m \leq n^2 - n) \mid \mathbf{D}_B \sqcup \mathbf{D}^+_{m!} \;\; (m < n) \mid \\
&\qquad \mathbf{D}_a \sqcup \mathbf{D}^+_{m!} \sqcup \mathbf{D}_{l!} \quad (\text{for } r := n - (m + l) \text{ we have } r > 0 \text{ and } r(r-1) \geq m) \\
&\qquad \text{where no disjuncts are added for expressions } \mathbf{D}^+_{0!} \text{ and } \mathbf{D}_{0!}
\end{aligned}$$

Extended concepts with restricted forms of ("local") disjunctions, used in $\mathbf{D}^{\geqslant n}$ only

$$\begin{aligned}
\mathbf{C}^+_B &::= \mathbf{C}_B \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}^+_B \mid \leqslant n\,\mathbf{R}.\neg(\mathbf{D}^+_a \cap \mathbf{D}_{\geq\omega-m}) \mid \mathbf{C}^+_B \sqcap \mathbf{C}^+_B \\
\mathbf{D}^+_B &::= \mathbf{C}^+_B \mid \mathbf{D}^+_B \sqcup \mathbf{D}^+_B \mid \mathbf{D}^+_a \sqcup \mathbf{C}_{\geq} \\
\mathbf{C}^+_H &::= \mathbf{C}_H \mid \geqslant n\,\mathbf{R}.\mathbf{D}^+_{n!} \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}^+_H \mid \leqslant 1\,\mathbf{R}.\neg\mathbf{D}^+_B \mid \leqslant n\,\mathbf{R}.\neg(\mathbf{D}^+_a \cap \mathbf{D}_{\geq\omega-m}) \mid \\
&\qquad \mathbf{C}^+_H \sqcap \mathbf{C}^+_H \mid \mathbf{D}^+_{1!} \\
\mathbf{D}^+_H &::= \mathbf{C}^+_H \mid \mathbf{D}^+_H \sqcup \mathbf{D}^+_B \mid \mathbf{D}^+_a \sqcup \mathbf{C}_{\geq} \\
\mathbf{C}^+_a &::= \mathbf{C}^+_H \mid \geqslant n\,\mathbf{R}.(\mathbf{D}^+_a \cup \{\top\}) \mid \mathbf{C}^+_a \sqcap \mathbf{C}^+_a \\
\mathbf{D}^+_a &::= \mathbf{C}^+_a \mid \mathbf{D}^+_a \sqcup \mathbf{D}^+_a \\
\mathbf{D}^+_{1!} &::= \{\mathbf{I}\} \sqcap \mathbf{C}^+_a \\
\mathbf{D}^+_{m+1!} &::= \mathbf{D}^+_{m!} \sqcup \mathbf{D}^+_{1!}
\end{aligned}$$

Figure 7.5: Grammars of DLP concepts in DLP normal form, simplified as discussed in Section 3.1.3

$\mathbf{C}_B$ play a similar role in our definition. $\mathbf{C}_H$ represents concepts that carry the full expressive power of a DLP GCI, and can serve as right-hand sides ("heads") of

Additional concepts based on global domain size restrictions

$$\begin{aligned}
\mathbf{D}^{=1} &::= \{\mathbf{I}\} \sqcap \mathbf{C}^p_H \\
\mathbf{D}^{=m+1} &::= \mathbf{D}^{=m} \sqcup (\{\mathbf{I}\} \sqcap \mathbf{C}^{=m+1}_\bot)
\end{aligned}$$

Additional concepts expressing $\top$ for unary domains (‘propositional’ case)

$$\begin{aligned}
\mathbf{C}^p_\top &::= \{\mathbf{I}\} \mid \mathbf{C}^p_\top \sqcap \mathbf{C}^p_\top \mid \leqslant 0\, \mathbf{R}.\neg(\mathbf{D}^p_\top) \mid \leqslant n\, \mathbf{R}.\neg\mathbf{D}\ (n \geq 1) \\
\mathbf{D}^p_\top &::= \mathbf{C}^p_\top \mid \mathbf{D}^p_\top \sqcup \mathbf{D}
\end{aligned}$$

Additional head and body concept expressions for unary domains (‘propositional’ case)

$$\begin{aligned}
\mathbf{C}^p_B &::= \mathbf{C}^{=1}_\bot \mid \mathbf{C}^p_\top \mid \neg\mathbf{A} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \mathbf{C}^p_B \sqcap \mathbf{C}^p_B \mid \leqslant 0\, \mathbf{R}.\neg(\mathbf{D}^p_B \cup \{\bot\}) \\
\mathbf{D}^p_B &::= \mathbf{D}^p_\top \mid \mathbf{D}^p_B \mid \mathbf{D}^p_B \sqcup \mathbf{D}^p_B \\
\mathbf{C}^p_H &::= \mathbf{C}^p_B \mid \mathbf{A} \mid \exists\mathbf{R}.\mathsf{Self} \mid \mathbf{C}^p_H \sqcap \mathbf{C}^p_H \mid \geqslant 1\, \mathbf{R}.\mathbf{D}^p_H \mid \leqslant 0\, \mathbf{R}.\neg\mathbf{D}^p_H \\
\mathbf{D}^p_H &::= \mathbf{D}^p_\top \mid \mathbf{C}^p_H \mid \mathbf{D}^p_H \sqcup \mathbf{D}^p_B
\end{aligned}$$

Additional structurally unsatisfiable concepts for domains of restricted size

$$\begin{aligned}
\mathbf{C}^{=1}_\bot &::= \neg\{\mathbf{I}\} \mid \mathbf{C}^{=1}_\bot \sqcap \mathbf{C} \mid \geqslant 1\, \mathbf{R}.\mathbf{D}^{=1}_\bot \mid \geqslant n\, \mathbf{R}.\mathbf{D}\ (n \geq 2) \\
\mathbf{C}^{=m+1}_\bot &::= \mathbf{C}^{=m+1}_\bot \sqcap \mathbf{C} \mid \geqslant n\, \mathbf{R}.\mathbf{D}^{=m+1}_\bot\ (n \geq 1) \mid \geqslant n\, \mathbf{R}.\mathbf{D}\ (n \geq m+2) \\
\mathbf{D}^{=m}_\bot &::= \mathbf{C}^{=m}_\bot \mid \mathbf{D}^{=m}_\bot \sqcup \mathbf{D}^{=m}_\bot
\end{aligned}$$

Concepts that can never hold for all individuals

$$\mathbf{C}_{\neq\top} ::= \neg\{\mathbf{I}\} \mid \mathbf{C}_{\neq\top} \sqcap \mathbf{C}$$

D: concepts in DLP normal form that are not structurally valid or unsatisfiable
C: concepts of **D** that are no disjunctions

Figure 7.6: Grammars of DLP concepts: special cases with restricted domain size

DLP GCIs. $\mathbf{C}_B$ concepts can be seen as negated generic left-hand sides (“bodies”) of GCIs. However, these basic classes are not sufficient for defining a maximal DLP. $\mathbf{C}_a$ characterises concept expressions which can be asserted for named individuals – these are even more expressive than $\mathbf{C}_H$ in that existential restrictions are allowed (intuitively, this is possible as in the context of known individuals the existentially asserted role neighbours can be expressed by Skolem constants). $\mathbf{D}_{m!}$ concepts then can be viewed as collections of individual assertions (e.g. $\{a\} \sqcap B$). Another way of stating such assertions is to use $\mathbf{C}_\geq$ in a disjunction (e.g. $\neg\{a\} \sqcup B$).

By far the most complex semantic interactions occur for atleast-restrictions in ABox assertions: $\mathbf{D}^{\geqslant n}$ and all subsequent definitions address this single case. For example, the $\mathcal{DLP}$ axiom $\{a\} \sqsubseteq \geqslant 2\, R.(\neg\{b\} \sqcup A \sqcup B)$ can be semantically emulated by the following set of datalog rules, where c_i are auxiliary constants:

$$R(a, c_1), \quad R(a, c_2), \quad b \approx c_1 \to A(b), \quad b \approx c_2 \to B(b), \quad c_1 \approx c_2 \to \bot.$$

This emulation uses internal symbols to resolve apparently disjunctive cases in a deterministic way. The datalog program does not represent disjunctive information: its least model simply contains two successors that are not equal to b. The nested disjunction only becomes relevant in the context of some disjunctive $\mathbf{FOL}_{\approx}$ formula, such as $\forall x.x \approx a \vee x \approx b$. The considered theory is no longer datalog in this case, and the program simply "re-uses" the disjunctive expressive power provided by the external theory. The fact that the actual program is far from being semantically equivalent to the original axiom illustrates the motive and utility of our definition of semantic emulation.

Many uses of nominals and atleast-restrictions lead to more complex interactions, some of which require completely different encodings. This is witnessed by the more complex arithmetic side condition used in $\mathbf{D}^{\geqslant n}$. Concepts in $\mathbf{D}_{\leq m} \cap \mathbf{D}^{+}_{a}$ correspond to disjunctions of m nominal classes, each of which is required to satisfy further disjunctive conditions, as e.g. $\{b\} \sqcap \geqslant 1\, R.(A \sqcup B)$. Now, as an example, a disjunction of an atomic class and four such "disjunctive nominals" is allowed as a filler for $\geqslant 7$ (since $3 \times 2 \geq 4$) but not for $\geqslant 6$ (since $2 \times 1 < 4$). Also note that the disjunctive concepts like $\mathbf{D}^{+}_{H}$ and $\mathbf{D}^{+}_{a}$ that are allowed in fillers do not allow all types of disjunctive information but only a finite amount of "local" disjunctions. For example, $\{a\} \sqcup B \sqcup C$ requires one "local" decision about a, whereas concepts like $\{a\} \sqcap \leqslant 0\, R.\neg(B \sqcup C)$ or $\{a\} \sqcap \leqslant 2\, R.\neg\bot$ require arbitrarily many decisions for all R successors.

The remaining grammars in Fig. 7.6 take care of less interesting special cases. Most importantly, $\mathbf{C}^{p}_{H}$ covers all concepts that can be emulated if the interpretation domain is restricted to contain just one individual. $\mathbf{C}_{\neq\top}$ contains axioms which make the knowledge base inconsistent as they deny the existence of a nominal. The auxiliary classes $\mathbf{C}^{=m}_{\bot}$ describe concepts that cannot be satisfied by an interpretation with at most m elements in their domain, as described in the following lemma.

Lemma 7.3.6 *A name-separated concept $C \neq \bot$ in DLP normal form is in $\mathbf{C}^{=m}_{\bot}$ as defined in Fig. 7.6 for some $m \geq 1$ iff, for all interpretations $\mathcal{I}$ with domain size $\#(\Delta^{\mathcal{I}}) \leq m$, we find $\mathcal{I} \models C \sqsubseteq \bot$.*

Proof. The "only if" direction can be shown by an easy induction, where the base cases are given by concepts $\geqslant n\, R.D$ with $n > m$, and – in the case $n = 1$ – negated nominals $\neg\{a\}$. The proof is straightforward and we omit further details.

For the "if" direction, assume that $C \notin \mathbf{C}^{=m}_{\bot} \cup \{\bot\}$, and let Δ be a domain of size m, i.e. $\#(\Delta) = m$. Then, for any $\delta \in \Delta$, we can find an interpretation $\mathcal{I}(\delta, C)$ such that $\Delta^{\mathcal{I}(\delta,C)} = \Delta$ and $\delta \in C^{\mathcal{I}(\delta,C)}$. The base cases with C of the form $\mathbf{C}$, $\exists\mathbf{R}.\mathsf{Self}$, $\{\mathbf{I}\}$, $\neg\mathbf{C}$, $\neg\exists\mathbf{R}.\mathsf{Self}$, and – if $n = 1$ – $\neg\{\mathbf{I}\}$ are obvious. If $C = D_1 \sqcup D_2$, then, without loss of generality, $D_1 \notin \mathbf{C}^{=m}_{\bot}$ and $\mathcal{I}(\delta, C) := \mathcal{I}(\delta, D_1)$ satisfies the claim.

Now assume that C is of the form $D_1 \sqcap D_2$. Then $D_1, D_2 \notin \mathbf{C}^{=m}_{\bot} \cup \{\bot\}$, and

C	$\mathsf{dlg}_B(\neg A \sqsubseteq C)$
$\bot$	$\{A(x)\} \cup P_{\mathrm{Inv}}$
$\neg B$	$\{B(x) \to A(x)\} \cup P_{\mathrm{Inv}}$
$\neg\{c\}$	$\{A(c)\} \cup P_{\mathrm{Inv}}$
$\neg\exists R.\mathsf{Self}$	$\{R(x,x) \to A(x)\} \cup P_{\mathrm{Inv}}$
$D_1 \sqcap D_2$	$\mathsf{dlg}_B(\neg A \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg A \sqsubseteq D_2)$
$D_1 \sqcup D_2$	$\mathsf{dlg}_B(\neg X_1 \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg X_2 \sqsubseteq D_2) \cup \{X_1(x) \land X_2(x) \to A(x)\}$
$\leqslant 0\, R.\neg D$	$\mathsf{dlg}_B(\neg X \sqsubseteq D) \cup \{R(x,y) \land X(y) \to A(x)\}$
A, B concept names, c an individual name, R a role, $X_{(i)}$ fresh concept names	

Figure 7.7: Transforming axioms $\neg\mathbf{A} \sqsubseteq (\mathbf{D}_B \cup \{\bot\})$ to datalog

we find interpretations $\mathcal{I}(\delta, D_1)$ and $\mathcal{I}(\delta, D_2)$ as in the hypothesis. Since C is name-separated, the hypothesis for D_1 is also satisfied by any variant $\mathcal{I}'(\delta, D_1)$ of $\mathcal{I}(\delta, D_1)$ which is obtained by changing the interpretation of symbols that occur in D_2. Thus we can assume without loss of generality that $\mathcal{I}(\delta, D_1)$ has been chosen such that it agrees with $\mathcal{I}(\delta, D_2)$ on all signature symbols that occur in D_2. By a symmetric argumentation for $\mathcal{I}(\delta, D_2)$, we find that such an $\mathcal{I}(\delta, D_1)$ would also satisfy the hypothesis for D_2, and hence we can set $\mathcal{I}(\delta, C) \coloneqq \mathcal{I}(\delta, D_1)$.

If $C = \leqslant n\, R.D$, then any interpretation $\mathcal{I}(\delta, C)$ with $R^{\mathcal{I}}(\delta, C) = \emptyset$ satisfies the claim. If $C = \geqslant n\, R.D$ with $n \leq m$, then consider distinct elements $\delta_1, \ldots, \delta_n \in \Delta$. Using structurality and the induction hypothesis again, we find a model $\mathcal{I}(\delta, C) = \mathcal{I}(\delta_1, D) = \ldots = \mathcal{I}(\delta_n, D)$ such that $R^{\mathcal{I}(\delta,C)} = \{\langle \delta, \delta_i \mid 1 \leq i \leq n\rangle\}$. □

7.4 Emulating $\mathcal{DLP}$ in Datalog

In this section, we show that knowledge bases of $\mathcal{DLP}$ as given in Definition 7.3.5 can indeed be emulated in datalog.

Emulations are generally established by means of recursively defined functions that translate $\mathcal{DLP}$ axioms to datalog. Relevant (auxiliary) transformations are required for each of the languages defined in Fig. 7.5 and 7.6. In all cases, the built-in semantics of inverse roles is explicitly needed in datalog. For this purpose, an auxiliary datalog program P_{Inv} is defined as $P_{\mathrm{Inv}} \coloneqq \{R(x,y) \to \mathrm{Inv}(R)(y,x) \mid R \in \mathbf{R}\}$, where $\mathbf{R}$ is the set of roles of the given signature. We begin with the rather simple case of $\mathbf{D}_B$.

Lemma 7.4.1 *Every $\mathcal{DLP}$ axiom $\neg A \sqsubseteq C$ with A a concept name and $C \in \mathbf{D}_B \cup \{\bot\}$ is semantically emulated by the datalog program $\mathsf{dlg}_B(\neg A \sqsubseteq C)$ as defined in Fig. 7.7.*

Proof. Note that the definition in Fig. 7.7 is well – especially all recursive uses of dlg_B refer to arguments in the domain of this function. The proof proceeds by induction over the structure of C, showing that the conditions of Definition 2.2.1 are satisfied. We show a single induction step to illustrate the easy argumentation.

Consider the case $C = D_1 \sqcup D_2$. For one direction of the claim, consider any model $\mathcal{I}$ of $\neg A \sqsubseteq C$. An interpretation $\mathcal{I}'$ over the extended signature is defined by setting $X_i^{\mathcal{I}'} := \Delta^{\mathcal{I}} \setminus D_i^{\mathcal{I}}$ for $i = 1, 2$. It is easy to see that $\mathcal{I}' \models \{\neg X_i \sqsubseteq D_i \mid i = 1, 2\} \cup \{X_1(x) \wedge X_2(x) \to A(x)\}$. By the induction hypothesis, we can find an interpretation $\mathcal{I}_1$ that extends $\mathcal{I}'$ and such that $\mathcal{I}_1 \models \mathsf{dlg}_B(\neg X_1 \sqsubseteq D_1)$. Another application of the hypothesis yields a model $\mathcal{I}_2 \models \mathsf{dlg}_B(\neg A \sqsubseteq C)$ as required to show the claim. The other direction requires us to show that every model of $\mathsf{dlg}_B(\neg A \sqsubseteq C)$ is also a model of $\neg A \sqsubseteq C$, which is obvious when applying the induction hypothesis. □

Now define, for a datalog program P and a ground literal $A(c)$, a datalog program $P|_{A(c)} := \{A(c) \wedge F \to H \mid F \to H \in P\}$. This way of manipulating datalog programs is convenient for our following definitions. Clearly, if P semantically emulates a formula φ, then $P|_{A(c)}$ semantically emulates $\varphi \vee \neg A(c)$.

The remaining language definitions of Fig. 7.5 are interdependent, so the corresponding translation needs to be established in a single recursion for which semantic emulation is shown in a single structural induction. We still separate the relevant claims for clarity, so the following lemmata can be considered as induction steps in the overall proof. The following lemma illustrates a first, simple induction step:

Lemma 7.4.2 *Consider a concept $C \in \mathbf{D}_H$ such that, for every proper subconcept $D \in \mathbf{D}_a$ of C and individual name d, the program $\mathsf{dlg}_a(\{d\} \sqsubseteq D)$ semantically emulates $\{d\} \sqsubseteq D$. Then, given a concept name A, the datalog program $\mathsf{dlg}_H(A \sqsubseteq C)$ as defined in Fig. 7.8 semantically emulates $A \sqsubseteq C$, where we use $\mathsf{ind}(E)$ to denote the set of individual names that occur in a concept E.*

Proof. Note that the definition is well, and especially that all uses of programs $\mathsf{dlg}_a(\{d\} \sqsubseteq D)$ do indeed refer to proper subconcepts D of C. The proof proceeds by induction, using similar arguments as in Lemma 7.4.1. We illustrate a single case which uses some features that did not occur before.

Consider the case $C = \{c\} \sqcap D \in \mathbf{D}_{1!}$. For the one direction, let $\mathcal{I}$ be a model of $A \sqsubseteq C$. If $\pi(\{c\} \sqsubseteq D)$ is a first-order formula that corresponds to $\{c\} \sqsubseteq D$, then $\mathcal{I} \models \neg A(c) \vee \pi(\{c\} \sqsubseteq D)$. Moreover, $\mathcal{I} \models A \sqsubseteq \{c\}$. By our assumptions and

C	$\mathsf{dlg}_H(A \sqsubseteq C)$
$D \in \mathbf{D}_B$	$\mathsf{dlg}_B(\neg X \sqsubseteq D) \cup \{A(x) \land X(x) \to \bot\}$
B	$\{A(x) \to B(x)\} \cup P_{\mathrm{Inv}}$
$\{c\}$	$\{A(x) \to c \approx x\} \cup P_{\mathrm{Inv}}$
$\exists R.\mathsf{Self}$	$\{A(x) \to R(x,x)\} \cup P_{\mathrm{Inv}}$
$D_1 \sqcap D_2 \in (\mathbf{D}_H \sqcap \mathbf{D}_H)$	$\mathsf{dlg}_H(A \sqsubseteq D_1) \cup \mathsf{dlg}_H(A \sqsubseteq D_2)$
$\{c\} \sqcap D \ \in \mathbf{D}_{1!}$	$\mathsf{dlg}_a(\{c\} \sqsubseteq D)\vert_{A(c)} \cup \mathsf{dlg}_H(A \sqsubseteq \{c\})$
$D_1 \sqcup D_2 \in (\mathbf{D}_H \sqcup \mathbf{D}_B)$	$\mathsf{dlg}_H(X_2 \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg X_1 \sqsubseteq D_2) \cup \{A(x) \land X_1(x) \to X_2(x)\}$
$D_1 \sqcup D_2 \in (\mathbf{D}_a \sqcup \mathbf{C}_{\geq})$	$\bigcup_{c \in \mathsf{ind}(D_2)} \mathsf{dlg}_a(\{c\} \sqsubseteq D_1)\vert_{A(c)}$
$\geqslant n\, R.D \ \in (\geqslant n\, \mathbf{R}.\mathbf{D}_{n!})$	$\bigcup_{c \in \mathsf{ind}(D)} \Big(\{A(x) \to R(x,c)\} \cup \bigcup_{d \in \mathsf{ind}(D) \setminus \{c\}} \{A(x) \land c \approx d \to \bot\} \cup$ $\mathsf{dlg}_a(\{c\} \sqsubseteq D_c)\Big)$ where D_c is such that $D = D'_c \sqcup (D_c \sqcap \{c\})$ for some $D'_c \in \mathbf{D}_{n-1!}$
$\leqslant 0\, R.\neg D$	$\mathsf{dlg}_H(X \sqsubseteq D) \cup \{A(x) \land R(x,y) \to X(y)\}$
$\leqslant 1\, R.\neg D$	$\mathsf{dlg}_B(\neg X \sqsubseteq D) \cup \{A(x) \land R(x,y) \land X(y) \land R(x,z) \land X(z) \to y \approx z\}$
A, B concept names, c, d individual names, R a role, $X_{(i)}$ fresh concept names, $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ as defined in Fig. 7.9 below	

Figure 7.8: Transforming axioms $\mathbf{A} \sqsubseteq \mathbf{D}_H$ to datalog

the induction hypothesis, $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$ semantically emulates $\{c\} \sqsubseteq D$ – hence $\mathsf{dlg}_a(\{c\} \sqsubseteq D)|_{A(c)}$ semantically emulates $\neg A(c) \lor \pi(\{c\} \sqsubseteq D)$ –, and $\mathsf{dlg}_H(A \sqsubseteq \{c\})$ semantically emulates $A \sqsubseteq \{c\}$. Since the auxiliary symbols that may occur in both datalog programs are distinct, semantic emulation yields a single extended interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models \mathsf{dlg}_a(\{c\} \sqsubseteq D)|_{A(c)}$ and $\mathcal{I}' \models \mathsf{dlg}_H(A \sqsubseteq \{c\})$, as required. The other direction is shown in a similar fashion by applying the induction hypothesis and assumptions of the lemma. □

The induction steps for defining $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ are rather more complex, and some preparation is needed first. Concepts of the forms $\mathbf{D}_a^+$, $\mathbf{D}_H^+$, and $\mathbf{D}_B^+$ allow for restricted forms of "local" disjunction. To make this notion explicit, we first elaborate how such concepts can be expressed as disjunctions of finitely many $\mathcal{DLP}$ knowledge bases.

Definition 7.4.3 Consider concept expressions C and D such that:

- $C \in \neg\mathbf{C}$ and $D \in \mathbf{D}_B^+$, or
- $C \in \mathbf{C}$ and $D \in \mathbf{D}_H^+$, or
- $C \in \{\mathbf{I}\}$ and $D \in \mathbf{D}_a^+$.

A set of knowledge bases $\mathcal{K}_{C\sqsubseteq D}$ is defined recursively as follows:

(1) If $D \in \mathbf{D}_a$ then $\mathcal{K}_{C\sqsubseteq D} := \{\{C \sqsubseteq D\}\}$.
Assume $D \notin \mathbf{D}_a$ for the remaining cases.

(2) If $D = D_1 \sqcap D_2$ then $\mathcal{K}_{C\sqsubseteq D} := \{\mathrm{KB}_1 \cup \mathrm{KB}_2 \mid \mathrm{KB}_1 \in \mathcal{K}_{C\sqsubseteq D_1}, \mathrm{KB}_2 \in \mathcal{K}_{C\sqsubseteq D_2}\}$.

(3) If $D = D_1 \sqcup D_2$ then:

(3a) If $D_1 \in \mathbf{C}_{\geq}$, define auxiliary sets of knowledge bases $\mathcal{K}_M$ for $M \subseteq \mathsf{ind}(D_1)$ as follows: $\mathcal{K}_M := \Big\{\{C \sqsubseteq \bigsqcap_{d\in M} \neg\{d\}\} \cup \bigcup_{d\in\mathsf{ind}(D_1)\setminus M} \mathrm{KB}_d \mid \mathrm{KB}_d \in \mathcal{K}_{\{d\}\sqsubseteq D_2}\Big\}$. Then set $\mathcal{K}_{C\sqsubseteq D} := \bigcup_{M\subseteq\mathsf{ind}(D_1)} \mathcal{K}_M$.

(3b) If $D_1 \in \mathbf{D}_B^+ \setminus \mathbf{C}_{\geq}$, then consider fresh concept names B_1 and B_2, and define $\mathcal{K}_{C\sqsubseteq D} := \Big\{\{C \sqsubseteq \neg B_1 \sqcup B_2\} \cup \mathrm{KB}_1 \cup \mathrm{KB}_2 \mid \mathrm{KB}_1 \in \mathcal{K}_{\neg B_1 \sqsubseteq D_1}, \mathrm{KB}_2 \in \mathcal{K}_{B_2 \sqsubseteq D_2}\Big\}$.

(3c) If $D_1, D_2 \notin \mathbf{D}_B^+$, then $\mathcal{K}_{C\sqsubseteq D} := \mathcal{K}_{C\sqsubseteq D_1} \cup \mathcal{K}_{C\sqsubseteq D_2}$.

(4) If $D = \geqslant n\, R.D'$ then:

(4a) If $D' \in \mathbf{D}_{n!}^+$ then w.l.o.g. $D' = D_1 \sqcup \ldots \sqcup D_n$ with $D_i = \{d_i\} \sqcap D_i'$ and $D_i' \in \mathbf{C}_a^+$. Define $\mathcal{K}_{C\sqsubseteq D} := \Big\{\{C \sqsubseteq \geqslant n\, R. \bigsqcup_{i=1}^n \{d_i\}\} \cup \bigcup_{i=1}^n \mathrm{KB}_i \mid \mathrm{KB}_i \in \mathcal{K}_{\{d_i\}\sqsubseteq D_i'}\Big\}$.

(4b) If $D' \notin \mathbf{D}_{n!}^+$ then consider a fresh individual name d and assume that $\mathcal{K}_{\{d\}\sqsubseteq D'} = \{\mathrm{KB}_1, \ldots, \mathrm{KB}_s\}$. Let d_i ($i = 1, \ldots, n$) be fresh individuals, and let KB_j^i denote the knowledge base KB_j with all occurrences of d replaced by d_i. Then define $\mathcal{K}_{C\sqsubseteq D} := \Big\{\{\{d_i\} \sqcap \{d_j\} \sqsubseteq \bot \mid 1 \leq i < j \leq n\} \cup \{C \sqsubseteq \geqslant 1\, R.\{d_i\} \mid 1 \leq i \leq n\} \cup \bigcup_{1\leq i\leq n} \mathrm{KB}_{k_i}^i \mid k_1, \ldots, k_n \in \{1, \ldots, s\}\Big\}$.

(5) If $D = \leqslant n\, R.\neg D'$ then:

(5a) If $D' \in \mathbf{C}_{\geq}$ then a $\geq n$-partitioning $\mathcal{M}$ of $\mathsf{ind}(D')$ is a set $\mathcal{M} = \{M_1, \ldots, M_m\}$ of $m \geq n$ mutually disjoint non-empty sets $M_i \subseteq \mathsf{ind}(D')$. Given such a $\geq n$-partitioning, define $\mathrm{KB}_{\mathcal{M}} := \{\{c\} \sqsubseteq \{d\} \mid c, d \in M_i \text{ for some } i \in \{1, \ldots, m\}\} \cup \{C \sqcap \bigsqcap_{c\in S} \geqslant 1\, R.\{c\} \sqsubseteq \bot \mid S \subseteq \mathsf{ind}(D'), \#\{M_i \mid M_i \cap S \neq \emptyset\} > n\}$. Then define $\mathcal{K}_{C\sqsubseteq D} := \{\mathrm{KB}_{\mathcal{M}} \mid \mathcal{M} \text{ a } \geq n\text{-partitioning of } \mathsf{ind}(D')\}$.

(5b) If $D' = D_1 \sqcup D_2$ where $D_1 \in \mathbf{C}_{\geq}$ and $D_2 \in \mathbf{D}_a^+$, then define a set of knowledge bases $\mathcal{K}_M$ for a set $M \subseteq \mathsf{ind}(D_1)$ as follows: $\mathcal{K}_M := \Big\{\mathrm{KB} \cup \bigcup_{d\in M} \mathrm{KB}_d \mid \mathrm{KB} \in \mathcal{K}_{C\sqsubseteq D''} \text{ with } D'' = \leqslant n\, R.\neg \bigsqcap_{d\in\mathsf{ind}(D_1)\setminus M} \neg\{d\}, \mathrm{KB}_d \in \mathcal{K}_{\{d\}\sqsubseteq D_2}\Big\}$. Then define $\mathcal{K}_{C\sqsubseteq D} := \bigcup_{M\subseteq\mathsf{ind}(D_1)} \mathcal{K}_M$.

(5c) If $n \leq 1$ and $D' \in \mathbf{D}_H^+$ then consider a fresh concept name B, and set $C' := \neg B$ if $D' \in \mathbf{D}_B^+$ and $C' := B$ otherwise. Define $\mathcal{K}_{C\sqsubseteq D} := \Big\{\{C \sqsubseteq \leqslant n\, R.\neg C'\} \cup \mathrm{KB} \mid \mathrm{KB} \in \mathcal{K}_{C'\sqsubseteq D'}\Big\}$.

As usual, empty conjunctions are treated as $\top$. In cases (3a) and (5b), the construction may lead to axioms in $\mathbf{L}_\top$; these axioms are omitted from $\mathcal{K}_{C\sqsubseteq D}$. $\diamond$

Observe that, without loss of generality, the cases in the previous definition are indeed exhaustive and mutually exclusive for $D \in \mathbf{D}_a^+$. In particular, cases (5a) and (5b) cover all situations where $D \in (\mathbf{D}_{\geq\omega-m} \cap \mathbf{D}_a^+)$, where we find $\#\mathsf{ind}(D') > n$ and $\#\mathsf{ind}(D_1) > n$, respectively, since we assume that $D \notin \mathbf{D}_a$. It is easy to verify that all recursive uses of $\mathcal{K}_{C\sqsubseteq D}$ satisfy the definition's conditions on C and D, and that all axioms in knowledge bases of $\mathcal{K}_{C\sqsubseteq D}$ are in DLP normal form. Note that case (4b) can only occur if $D \in \mathbf{D}_a^+ \setminus \mathbf{D}_H^+$, so C must be a nominal in these cases. Similar observations for the other cases allow us to state the following lemma.

Lemma 7.4.4 *Consider concept expressions C and D as in Definition 7.4.3. If D is in $\mathbf{D}_a^+$ ($\mathbf{D}_H^+$, $\mathbf{D}_B^+$) then all axioms of the form $C \sqsubseteq E$ in knowledge bases of $\mathcal{K}_{C\sqsubseteq D}$ are such that E is in $\mathbf{D}_a$ ($\mathbf{D}_H$, $\mathbf{D}_B$).*

In particular, the knowledge bases in $\mathcal{K}_{C\sqsubseteq D}$ are in $\mathcal{DLP}$.

Proof. The claim can be verified by considering all axioms that are created in the cases of Definition 7.4.3. The claims for $\mathbf{D}_a^+$, $\mathbf{D}_H^+$, and $\mathbf{D}_B^+$ are interdependent and must be proven together.

The claim clearly holds for the base case (1). Case (2) immediately follows from the induction hypothesis. Case (3a) is trivial since additional axioms of the form $C \sqsubseteq E$ do not occur in knowledge bases of $\mathcal{K}_{\{d\}\sqsubseteq D_2}$. Case (3b) and (3c) are again immediate from the induction hypothesis, where we note for (3b) that $D_1 \sqcup D_2$ is in $\mathbf{D}_a$ ($\mathbf{D}_H$, $\mathbf{D}_B$) for $D_1 \in \mathbf{D}_B \setminus \mathbf{C}_\geq$ whenever D_2 is in $\mathbf{D}_a$ ($\mathbf{D}_H$, $\mathbf{D}_B$).

Case (4a) can only occur if $D \in \mathbf{D}_H^+ \setminus \mathbf{D}_B^+$ so it suffices to note that the concept $\geqslant n\, R. \bigsqcup_{i=1}^n \{d_i\}$ is in $\mathbf{D}_H$. Case (4b) in turn requires that $D \in \mathbf{D}_a^+ \setminus \mathbf{D}_H^+$, and clearly $\geqslant 1\, R.\{d_i\} \in \mathbf{D}_a$.

Cases (5a) is immediate, since $C \sqcap \bigsqcap_{c\in S} \geqslant 1\, R.\{c\} \sqsubseteq \bot$ is equivalently expressed as $C \sqsubseteq \bigsqcup_{c\in S} \leqslant 0\, R.\neg\neg\{c\}$, the conclusion of which is in $\mathbf{D}_B$. Case (5b) follows directly by induction. Case (5c) comprises three relevant cases: $n = 0$ and $D' \in \mathbf{D}_B^+$ ($D \in \mathbf{D}_B^+$), $n = 0$ and $D' \in \mathbf{D}_H^+$ ($D \in \mathbf{D}_H^+$), $n = 1$ and $D' \in \mathbf{D}_B^+$ ($D \in \mathbf{D}_H^+$). We find that C' is in $\mathbf{D}_B$ ($\mathbf{D}_H$) whenever D' is in $\mathbf{D}_B^+$ ($\mathbf{D}_H^+$), so that the claim holds in each case.

It remains to show the second part of the claim. Using the first part of the claim, the preconditions on C and D imply that all axioms $C \sqsubseteq E$ that are constructed for $\mathcal{K}_{C\sqsubseteq D}$ are in $\mathcal{DLP}$. Axioms $C' \sqsubseteq E$ in $\mathcal{K}_{C\sqsubseteq D}$ with $C' \neq C$ must be obtained from some $\mathcal{K}_{C'\sqsubseteq D'}$ that was used in the construction of $\mathcal{K}_{C\sqsubseteq D}$. But such recursive constructions only occur in cases where the preconditions of the definition are satisfied, so the claim follows by induction. $\square$

The next proposition shows that $C \sqsubseteq D$ is emulated by the disjunction of the knowledge bases in $\mathcal{K}_{C\sqsubseteq D}$, thus establishing the correctness of the decomposition.

DL does not provide a syntax for knowledge base disjunctions, and we do not want to move to first-order logic here, so we use a slightly different formulation that follows Definition 2.2.1.

Proposition 7.4.5 *Consider concept expressions C and D as in Definition 7.4.3, both based on some signature $\mathscr{S}$. Let $\mathscr{S}'$ be the extended signature of $\mathcal{K}_{C\sqsubseteq D}$.*

- *Every interpretation $\mathcal{I}$ over $\mathscr{S}$ with $\mathcal{I} \models C \sqsubseteq D$ can be extended to an interpretation $\mathcal{I}'$ over $\mathscr{S}'$ such that $\mathcal{I}' \models \mathrm{KB}$ for some $\mathrm{KB} \in \mathcal{K}_{C\sqsubseteq D}$.*
- *For every interpretation $\mathcal{I}'$ over $\mathscr{S}'$ such that $\mathcal{I}' \models \mathrm{KB}$ for some $\mathrm{KB} \in \mathcal{K}_{C\sqsubseteq D}$, we find that $\mathcal{I}' \models C \sqsubseteq D$.*

Proof. We proceed by induction. Case (1) is obvious. Cases (2) is immediate from the induction hypothesis. For case (3a), let M be the largest set of individuals such that $\mathcal{I} \models C \sqsubseteq \bigsqcap_{d\in M} \neg\{d\}$. Using the induction hypothesis, it is easy to see that $\mathcal{I} \models C \sqsubseteq D$ implies that there is an extension $\mathcal{I}'$ of $\mathcal{I}$ such that $\mathcal{I}' \models \mathrm{KB}$ for some $\mathrm{KB} \in \mathcal{K}_M$. The converse is similar.

For case (3b), consider an interpretation $\mathcal{I}$ over $\mathscr{S}$ with $\mathcal{I} \models C \sqsubseteq D$. Consider the extended signature $\mathscr{S}'$ with the fresh concept names B_1 and B_2, and define an extension $\mathcal{I}''$ of $\mathcal{I}$ over $\mathscr{S}'$ by setting $B_1^{\mathcal{I}''} := \neg D_1^{\mathcal{I}}$ and $B_2^{\mathcal{I}''} := D_2^{\mathcal{I}}$. Then $\mathcal{I}'' \models \neg B_1 \sqsubseteq D_1$ and $\mathcal{I}'' \models B_2 \sqsubseteq D_2$, and we can apply the induction hypothesis for $\mathcal{K}_{\neg B_1\sqsubseteq D_1}$ and $\mathcal{K}_{B_2\sqsubseteq D_2}$ to obtain models $\mathcal{I}''_i$ (over some extended signature $\mathscr{S}'''$) such that $\mathcal{I}''_1 \models \mathrm{KB}_1$ for some $\mathrm{KB}_1 \in \mathcal{K}_{\neg B_1\sqsubseteq D_1}$ and $\mathcal{I}''_2 \models \mathrm{KB}_2$ for some $\mathrm{KB}_2 \in \mathcal{K}_{B_2\sqsubseteq D_2}$. Since $\mathcal{I}''_1$ and $\mathcal{I}''_2$ agree on B_1, B_2, and all symbols of $C \sqsubseteq D$, there is an interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models \mathrm{KB}_1 \cup \mathrm{KB}_2$. Since $C^{\mathcal{I}} = \neg B_1^{\mathcal{I}'} \cup B_2^{\mathcal{I}'}$, it is easy to see that $\mathcal{I}'$ satisfies the conditions of the claim. The other direction of the claim for (3b) is an easy consequence of the induction hypothesis.

Case (3c) can only occur if $C \in \{\mathbf{I}\}$, and it is easy to see that the claim holds in this case.

Case (4a) is again not hard to see when using the induction hypothesis. For case (4b), first note that C must be a nominal since D is cannot be in $\mathbf{D}_H$. The required semantic emulation then is an easy consequence of standard Skolemisation, where each successor d_i may satisfy any of the sufficient subconditions that are captured by $\mathrm{KB}^i_1, \ldots, \mathrm{KB}^i_s$.

The reasoning for case (5a) is similar to case (3a): given an interpretation $\mathcal{I}$, we find a $\geq n$-partitioning $\mathcal{M}$ such that $c, d \in M_i$ iff $c^{\mathcal{I}} = d^{\mathcal{I}}$. It is easy to see that $\mathcal{I} \models C \sqsubseteq D$ implies $\mathcal{I} \models \mathrm{KB}_{\mathcal{M}}$; no induction is required. The other direction is again obvious.

Case (5b) is a simple extension of case (5a) where a subset M of individuals is selected in each knowledge base to ensure that all individuals of M are instances of D_2, thus reducing the requirement to a maximal number of R-successors that do not belong to M. To express this more formally, we use expressions $\geqslant 1\, U.(\{d\} \sqcap E)$ where U is the universal role that can be semantically emulated in $\mathcal{DLP}$ – this allows us to embed ABox assertions into GCIs. With this notation, we observe that $C \sqsubseteq \leqslant n\, R.\neg(D_1 \sqcup D_2)$ is semantically emulated by the disjunction of all the axioms $C \sqsubseteq \leqslant n\, R.\neg \bigsqcap_{d \in \mathsf{ind}(D_1)\setminus M} \neg\{d\} \sqcap \bigsqcap_{d \in M} \geqslant 1\, U.(\{d\} \sqcap C_2)$ for all $M \subseteq \mathsf{ind}(D_1)$. It is easy to see that the construction in (5b) corresponds to this disjunction, where conjunction is modelled as in case (2), and individual assertions are encoded using the recursive constructions $\mathcal{K}_{\{d\} \sqsubseteq D_2}$ that are valid by the induction hypothesis. The converse is easily obtained by similar considerations.

Case (5c) uses a similar argument as case (3b). Consider an interpretation $\mathcal{I}$ over $\mathcal{S}$ with $\mathcal{I} \models C \sqsubseteq D$. For the extended signature $\mathcal{S}'$ with fresh concept name B, an extension $\mathcal{I}''$ of $\mathcal{I}$ is defined by setting $C'^{\mathcal{I}''} := D^{\mathcal{I}}$. By the induction hypothesis for $\mathcal{K}_{C' \sqsubseteq D'}$, we find a model $\mathcal{I}'$ (over some extended signature $\mathcal{S}''$) such that $\mathcal{I}' \models$ KB for some KB $\in \mathcal{K}_{C' \sqsubseteq D'}$. But then there is a corresponding knowledge base KB$' = \{C \sqsubseteq \leqslant n\, R.\neg C'\} \cup$ KB in $\mathcal{K}_{C \sqsubseteq D}$ such that $\mathcal{I}' \models$ KB$'$. Thus $\mathcal{I}'$ satisfies the conditions of the claim when restricted to $\mathcal{S}'$. The other direction is again easy. □

We can now define datalog programs for semantically emulating axioms of the form $\{c\} \sqsubseteq \geqslant n\, R.\mathbf{D}^{\geqslant n}$. We consider all three main cases – $\mathbf{C}_{\neg m} \sqcup \mathbf{D}_a^+$, $\mathbf{D}_B \sqcup \mathbf{D}_{m!}^+$, $\mathbf{D}_a \sqcup \mathbf{D}_{m!}^+ \sqcup \mathbf{D}_{l!}$ – individually, before combining these cases with the remaining forms of $\mathbf{D}_a$ to complete the induction.

Lemma 7.4.6 *Consider a constant c, and a concept $C = \geqslant n\, R.D_1 \sqcup D_2$ such that $D_1 \in \mathbf{C}_{\neg m}$, $D_2 \in \mathbf{D}_a^+$, and $(1 \leq m \leq n^2 - n)$.*

Assume that, for every individual symbol d and every knowledge base KB $\in \mathcal{K}_{\{d\} \sqsubseteq D_2}$*, there is a datalog program* $\mathsf{datalog}$(KB) *that semantically emulates* KB.

Then we can effectively construct a datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ that semantically emulates $\{c\} \sqsubseteq C$.

Proof. Let h be the smallest number such that $2^h \geq (\#\mathcal{K}_{\{d\} \sqsubseteq D_2})^n$, where d is an arbitrary constant (clearly, the cardinality $\#\mathcal{K}_{\{d\} \sqsubseteq D_2}$ does not depend on the choice of d). Now let $S := \{c_{ijk} \mid i, j \in \{1, \ldots, n\}, k \in \{1, \ldots, h\}\}$ be a set of $n \times n \times h$ fresh constants. It is convenient to consider the indices of constants in S to be coordinates, so that S consists of the elements of a three dimensional matrix with n rows, n columns, and h layers. Now given any $k = 1, \ldots, h$, we define sets $A_i^k, B_i^k \subseteq S$ for all $i = 1, \ldots, n$ by setting:

$$A_i^k := \{c_{i1k}, c_{i2k}, \ldots, c_{ink}\} \quad \text{and} \quad B_i^k := \{c_{1ik}, c_{2ik}, \ldots, c_{nik}\}.$$

In other words, A_i^k (B_i^k) is the ith row (column) in layer h of S. Now given a set $O \subseteq S$, define $O(k) := \{c_{ijk} \in O \mid i, j \in \{1, \ldots, n\}\}$ – the intersection of O with layer h in S. Now for every h-tuple $v = \langle X_1, \ldots, X_h \rangle$ with $X_k \in \{A, B\}$ for all

$k = 1, \ldots, h$, there is a unique partitioning $P_v = \{O_1, \ldots, O_n\}$ of S into n disjoint subsets $O_i \subseteq S$ $(1 \leq i \leq n)$ for which the following holds: for every $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, h\}$, we find that $O_i(k) = (X_k)^k_i$. Observe that the 2^h partitions P_v that can be constructed in this way are indeed mutually distinct. Intuitively, the partitions P_v thus encode binary numbers of h digits.

Given partitionings $P = \{O_1, \ldots, O_p\}$ and $P' = \{O'_1, \ldots, O'_{p'}\}$ of S, we say that P is *finer than* P' if, for every $i \in \{1, \ldots, p'\}$, we find that O'_i is a union of parts $O_j \in P$. Note that every part O_j can be contained in at most one part O'_i, and thus $p' \leq p$. Partitions of the form P_v have the following important property: for every partition $P = \{O_1, \ldots, O_p\}$ of S with $p \in \{n, \ldots, n + m - 1\}$, there is at most one partition of the form P_v such that P is finer than P_v. To show this, consider two h-tuples $v, w \subseteq \{A, B\}^h$ that differ in (at least) the kth component ($k \in \{1, \ldots, h\}$), i.e. (w.l.o.g.) the kth component of v is A, and the kth component of w is B. Now for any partition P that is finer than P_v and P_w, for every $i \in \{1, \ldots, n\}$ there are parts $O_1, \ldots, O_j \in P$ such that $A^k_i = O_1(k) \cup \ldots \cup O_j(k)$, and parts $O'_1, \ldots, O'_{j'} \in P$ such that $B^k_i = O'_1(k) \cup \ldots \cup O'_{j'}(k)$. This implies that P cannot contain a part O such that $\#O(k) > 1$ since no two sets A^k_i and $B^k_{i'}$ share more than one constant. Hence P must have at least n^2 parts to cover all elements in layer k. Now the precondition $m \leq n^2 - n$ implies that $n + m - 1 < n^2$, which establishes the claim.

To establish the required datalog program, partitions of constants are considered as equality classes, and rules are created to check for particular equalities. To this end, define a conjunction $[\![O]\!] := c_1 \wedge \ldots \wedge c_j$ for every set $O = \{c_1, \ldots, c_n\} \subseteq S$. This notation is extended to partitions $P = \{O_1, \ldots, O_i\}$ of S by setting $[\![P]\!] := [\![O_1]\!] \wedge \ldots \wedge [\![O_i]\!]$.

Consider a fresh constant d. For every h-tuple $v \in \{A, B\}^h$, let $\phi_v : P_v \to \mathcal{K}_{\{d\} \sqsubseteq D_2}$ be a mapping of parts of P_v to knowledge bases in $\mathcal{K}_{\{d\} \sqsubseteq D_2}$ such that, for every n-tuple $K = \langle \mathrm{KB}_1, \ldots, \mathrm{KB}_n \rangle \in \mathcal{K}^n_{\{d\} \sqsubseteq D_2}$ of knowledge bases, there is an h-tuple $w \in \{A, B\}^h$ with partition $P_v = \{O_1, \ldots, O_n\}$ as defined above, and $\phi_w(O_i) = \mathrm{KB}_i$ for all $i = 1, \ldots, n$. This choice of the functions ϕ_v is possible due to our initial choice of h, since there are 2^h such functions but only $\#\mathcal{K}^n_{\{d\} \sqsubseteq D_2}$ different n-tuples of knowledge bases from $\mathcal{K}_{\{d\} \sqsubseteq D_2}$.

For every partition P of S into $i \in \{1, \ldots, n + m - 1\}$ parts, datalog rules are constructed as follows. If P is not finer than any partition of the form P_v, then only the rule $[\![P]\!] \to \bot$ is added (this includes the case of P having less than n parts). Otherwise, let P_v be the unique partition of this form that is finer than P. For every part O of P_v, select one part $\pi(O)$ of P such that $\pi(O) \subseteq O$, so that there are n distinct *selected parts* in P. Now let $d_1, \ldots, d_m$ denote the m constants of D_1. For every $e = d_1, \ldots, d_m$ and for every part $O \in P_v$, let A be a fresh concept name and construct the following datalog:

(i) $[\![P]\!] \wedge e \approx f \to A(e)$, where $f \in \pi(O)$ is arbitrary,

(ii) $\mathsf{datalog}(\mathrm{KB}')|_{A(e)}$ where KB' is obtained from $\phi_v(O)$ by replacing all occurrences of $\{d\}$ with $\{e\}$.

Now $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined to be the union of P_{Inv} and all datalog rules constructed above, and the datalog facts $R(c, c_{ijk})$ for all $i, j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, h\}$.

It remains to show that $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ semantically emulates $\{c\} \sqsubseteq C$. For the one direction, consider a model $\mathcal{I}$ of $\{c\} \sqsubseteq C$. We need to show that it can be extended to a model of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. Select n distinct R-successors $\delta_1, \ldots, \delta_n$ of $c^{\mathcal{I}}$ such that $\delta_i \in (D_1 \sqcup D_2)^{\mathcal{I}}$ for all $i = 1, \ldots, n$. By Proposition 7.4.5, for all $e \in \{d_1, \ldots, d_m\}$, if $e^{\mathcal{I}} \in D_2^{\mathcal{I}}$ then there is an extended interpretation $\mathcal{I}_e$ such that $\mathcal{I}_e \models \mathrm{KB}_e$ for some $\mathrm{KB}_e \in \mathcal{K}_{\{e\} \sqsubseteq D_2}$. Since $\mathcal{I}_e$ extends $\mathcal{I}$ only over fresh symbols that occur in one $\mathcal{K}_{\{e\} \sqsubseteq D_2}$, all interpretations $\mathcal{I}_e$ can be combined into a single extension $\mathcal{I}'$ of $\mathcal{I}$.

Now let $\mathrm{KB}'_e \in \mathcal{K}_{\{d\} \sqsubseteq D_2}$ denote the knowledge base from which KB_e is obtained by replacing all axioms of the form $\{d\} \sqsubseteq F$ by $\{e\} \sqsubseteq F$, where d is the constant used when constructing $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. By the construction of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$, there is a tuple $v \in \{A, B\}^h$ and a partition $P_v = \{O_1, \ldots, O_n\}$ such that $\phi_v(O_i) = \mathrm{KB}'_{d_j}$ for all $i = 1, \ldots, n$ for which $d_j^{\mathcal{I}} = \delta_i$ and $d_l^{\mathcal{I}} \neq \delta_i$ for all $l < j$.

Consider any $e \in \{d_1, \ldots, d_m\}$ with $e^{\mathcal{I}} \in D_2^{\mathcal{I}}$. The model $\mathcal{I}'$ above was constructed such that $\mathcal{I}' \models \mathrm{KB}_e$, and thus, by the assumption of the lemma, there is an extension $\mathcal{J}'$ of $\mathcal{I}'$ such that $\mathcal{J}' \models \mathsf{datalog}(\mathrm{KB}_e)$. We define a model $\mathcal{J}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ by further extending $\mathcal{J}'$. For all constants $f \in S$, define $f^{\mathcal{J}} := \delta_i$ for the unique $i \in \{1, \ldots, n\}$ such that $f \in O_i$. Moreover, for each of the fresh concept name A introduced in (i) above, let $A^{\mathcal{J}}$ be the smallest extension for which all rules of (i) are satisfied by $\mathcal{J}$.

Now it is easy to see that $\mathcal{J}$ satisfies the facts $R(c, c_{ijk})$ for all $i, j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, h\}$. To see that it also satisfies the rules constructed in (ii) above, note that the rules (ii) for some particular $e \in \{c_1, \ldots, c_m\}$ are always satisfied if $\mathcal{J} \not\models A(e)$. Assume $\mathcal{J} \models A(e)$. By minimality of $A^{\mathcal{J}}$, this implies that $\mathcal{J} \models e \approx f$ for some $f \in S$ that belongs to a part O_i of P_v, and thus $e^{\mathcal{J}} = \delta_i$ for some $i \in \{1, \ldots, n\}$. By construction, $\phi_v(O_i)$ is of the form KB'_{d_j} (where e might be unequal to d_j, but with $e^{\mathcal{J}} = d_j^{\mathcal{J}} = \delta_i$). Since $\delta_i \in (D_1 \sqcup D_2)^{\mathcal{I}}$, we find $\delta_i \in D_2^{\mathcal{I}}$ and thus $\mathcal{J} \models \mathsf{dlg}_a(\{b'\} \sqsubseteq F)$ for all $\{b\} \sqsubseteq F \in \mathrm{KB}'_{d_j}$, where $b' = e$ if $b = d$ and $b' = b$ otherwise. This shows that the rules (ii) are indeed satisfied by $\mathcal{J}$.

For the other direction, consider a model $\mathcal{I}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. We need to show that it is also a model of $\{c\} \sqsubseteq C$. Let P be the partition of S that corresponds to the $\approx$ equivalence classes on S induced by $\mathcal{I}$. By the construction of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$, the partition P is finer than some partition of the form P_v, and thus has at least n parts. Moreover, n of the parts of P are selected parts of the form $\pi(O)$ for some

$O \in P_v$. It is not hard to see that the n domain elements of $\mathcal{I}$ that correspond to the selected parts are R-successors of c that belong to $(D_1 \sqcup D_2)^{\mathcal{I}}$, which is an easy consequence of rules (i) and (ii) together with the assumed model-theoretic correspondences for axioms in $\mathcal{K}_{\{d\} \sqsubseteq D_2}$. □

Lemma 7.4.7 *Consider a constant c, and a concept $C = \geqslant n\, R.D_1 \sqcup D_2$ such that $D_1 \in \mathbf{D}_B$, $D_2 \in \mathbf{D}^+_{m!}$ with $m < n$ of the form $D_2 = (\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_m\} \sqcap C_m)$.*

Assume that, for every $i \in \{1, \ldots, m\}$ and every knowledge base KB $\in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$, *there is a datalog program* datalog(KB) *that semantically emulates* KB.

Then we can effectively construct a datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ *that semantically emulates $\{c\} \sqsubseteq C$.*

Proof. For each $i = 1, \ldots, m$, let $l_i \geq 1$ be the least number such that $2^{l_i} \geq \#\mathcal{K}_{\{c_i\} \sqsubseteq C_i}$, and consider a set S_i of fresh constants $S_i := \{a_{i1}, b_{i1}, \ldots, a_{il_i}, b_{il_i}\}$. Let V_i denote the set of all sets of the form $\{x_1, x_2, \ldots, x_{l_i}\}$ with $x_h \in \{a_{ih}, b_{ih}\}$ for all $h \in \{1, \ldots, l_i\}$. Let $\phi_i : V_i \to \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$ be an arbitrary surjective function (which exists due to the choice of a sufficiently large l_i).

Consider fresh constants $d_1, \ldots, d_{n-m}$ (note that $n - m \geq 1$) and a fresh concept name B. We construct the following datalog rules and programs:

(i) $\mathsf{dlg}_B(\neg B \sqsubseteq D_1)$

(ii) for every $i \in \{1, \ldots, n-m\}$:
$R(c, d_i)$,
$B(d_i) \to \bot$ for a fresh concept name A,

(iii) for every $i, j \in \{1, \ldots, n-m\}$, $i \neq j$:
$d_i \approx d_j \to \bot$,

(iv) for every $i \in \{1, \ldots, n-m\}$ and $j \in \{1, \ldots, m\}$:
$d_i \approx c_j \to \bot$,

(v) for every $i \in \{1, \ldots, m\}$ and $h \in \{1, \ldots, l_i\}$:
$R(c, a_{ih})$,
$R(c, b_{ih})$,
$a_{ih} \approx b_{ih} \to \bot$,

(vi) for every $i \in \{1, \ldots, m\}$ and $v = \{x_{i1}, x_{i2}, \ldots, x_{il_i}\} \in V_i$:
$B(x_{i1}) \wedge \ldots \wedge B(x_{il_i}) \to A(c_i)$ for a fresh concept name A,
$A(c_i) \to c_i \approx x_{i1}$,
$\mathsf{datalog}(\phi_i(v))|_{A(c_i)}$,

(vii) for every $i, j \in \{1, \ldots, m\}$, $i \neq j$, for every $e \in S_i$ and $f \in S_j \cup \{d_1, \ldots, d_{n-m}\}$:
$e \approx f \to e \approx d_1$.

Now $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined as the union of P_{Inv} and all rules and programs constructed above.

It remains to show that $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ semantically emulates $\{c\} \sqsubseteq C$. For the one direction, consider any model $\mathcal{I}$ of $\{c\} \sqsubseteq C$. Select n distinct R-successors $\delta_1, \ldots, \delta_n$ of $c^{\mathcal{I}}$ such that $\delta_i \in (D_1 \sqcup D_2)^{\mathcal{I}}$ for all $i = 1, \ldots, n$. By Proposition 7.4.5, for all $i \in \{1, \ldots, m\}$, if $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$ then there is an extended interpretation $\mathcal{I}_i$ such that $\mathcal{I}_i \models \mathrm{KB}_i$ for some $\mathrm{KB}_i \in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$. Since $\mathcal{I}_i$ extends $\mathcal{I}$ only over fresh symbols that occur in one $\mathcal{K}_{\{c_i\} \sqsubseteq C_i}$, all interpretations $\mathcal{I}_i$ can be combined into a single extension $\mathcal{I}'$ of $\mathcal{I}$. By the assumption of the lemma, we find an extension $\mathcal{J}'$ of $\mathcal{I}'$ such that $\mathcal{J}' \models \mathsf{datalog}(\mathrm{KB}_i)$.

A model $\mathcal{J}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined by further extending $\mathcal{J}'$. For the auxiliary concept B of (i), define $B^{\mathcal{J}} := (\neg D_1)^{\mathcal{I}}$ and let $\mathcal{J}$ be such that $\mathcal{J} \models \mathsf{dlg}_B(\neg B \sqsubseteq D_1)$ (which is possible by Lemma 7.4.1). For each $i \in \{1, \ldots, n-m\}$, select $d_i^{\mathcal{J}} \in \{\delta_1, \ldots, \delta_n\}$ such that rules (ii)–(iv) above are satisfied. This is always possible since at most m elements of $\{\delta_1, \ldots, \delta_n\}$ can be in $(\neg D_1)^{\mathcal{I}}$. Without loss of generality, we assume that $d_i^{\mathcal{J}} = \delta_i$.

Now select an injective function $\psi : \{1, \ldots, m\} \to \{2, \ldots, n\}$ such that $\psi(i) = j$ if $c_i^{\mathcal{I}} = \delta_j$ for some $j \in \{1, \ldots, n\}$ and there is no $i' < i$ such that $c_{i'}^{\mathcal{I}} = \delta_j$; and $\psi(i) \in D_1^{\mathcal{I}}$ otherwise. Again, it is not hard to see that this is always possible. Now for each $i \in \{1, \ldots, m\}$, interpretations for constants in S_i are defined as follows. If $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$, then let $v \in V_i$ be such that $\mathrm{KB}_i = \phi_i(v)$. Otherwise, let $v \in V_i$ be arbitrary. For all $h \in \{1, \ldots, l_i\}$ and $x \in \{a_{ih}, b_{ih}\}$, define $x^{\mathcal{J}} := \delta_{\psi(i)}$ if $x \in v$, and $x^{\mathcal{J}} := \delta_1$ otherwise. It is not hard to see that $\mathcal{J}$ satisfies rules (v) and (vii). For the auxiliary concepts A introduced in (vi) for some set $w \in V_i$, set $A^{\mathcal{J}} := \{c_i^{\mathcal{I}}\}$ if $w = v$ and $\delta_{\psi(i)} \in (\neg D_1)^{\mathcal{I}}$ (which also implies $c_i^{\mathcal{I}} = \delta_{\psi(i)}$), and set $A^{\mathcal{J}} := \emptyset$ otherwise. Thus, there is at most one such auxiliary concept for i that is non-empty, corresponding to the set $v \in V_i$ for which $\mathrm{KB}_i = \phi_i(v)$. The construction of $\mathcal{J}'$ ensures that the remaining rules of (vi) are satisfied as required. It should be observed that this construction also works in the case that $c_i^{\mathcal{I}} = c_j^{\mathcal{I}}$ for some $i \neq j$.

For the other direction, consider any model $\mathcal{I}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. The rules of (i)–(iv) obviously establish $n - m$ distinct R-successors $d_1, \ldots, d_{n-m}$ of c that are in D_1. According to rules (vii), for every $i \in \{1, \ldots, m\}$ and every $k \in \{1, \ldots, l_i\}$, some $x_{ik} \in \{a_{ik}, b_{ik}\}$ is unequal to all constants in $S_j \cup \{d_1, \ldots, d_{m-n}\}$ for all $j \neq i$ with $j \in \{1, \ldots, m\}$. Hence, if the premise of the first rule of (vi) is false for all $v \in V_i$, then there must be some $k \in \{1, \ldots, l_i\}$ such that $x_{ik}^{\mathcal{I}} \notin B^{\mathcal{I}}$ and hence, by (i), $x_{ik}^{\mathcal{I}} \in D_1^{\mathcal{I}}$, yielding the required distinct R-successor for i. Otherwise, if the premise of the first rule of (vi) is true for some $v \in V_i$, then $c_i^{\mathcal{I}} \approx x_{i1}$ is the required successor, since $c_i^{\mathcal{I}} \in D_2^{\mathcal{I}}$ is ensured by the rules of (vi) together with the assumptions of the lemma. □

Lemma 7.4.8 *Consider a constant c, and a concept $C = \geqslant n\, R.D_1 \sqcup D_2 \sqcup D_3$ such that $D_1 \in \mathbf{D}_a$, $D_2 \in \mathbf{D}^+_{m!}$ of the form $D_2 = (\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_m\} \sqcap C_m)$, $D_3 \in \mathbf{D}^+_{l!}$ of the form $D_3 = (\{c_{m+1}\} \sqcap C_{m+1}) \sqcup \ldots \sqcup (\{c_{m+l}\} \sqcap C_{m+l})$, and for $r \coloneqq n - (m + l)$ we have $r > 0$ and $r(r - 1) \geq m$.*

Assume that, for every constant e, $\mathsf{dlg}_a(\{e\} \sqsubseteq D_1)$ semantically emulates $\{e\} \sqsubseteq D_1$, and that, for every KB $\in \mathcal{K}_{\{c_i\}\sqsubseteq C_i}$ ($i \in \{1, \ldots, m+l\}$), $\mathsf{datalog}(\mathrm{KB})$ semantically emulates KB.

Then we can effectively construct a datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ that semantically emulates $\{c\} \sqsubseteq C$.

Proof. Let $s \geq 1$ be such that $2^s \geq \prod_{i=1}^{m} \#\mathcal{K}_{\{c_i\}\sqsubseteq C_i}$. Consider the following sets of fresh constants:

- $\{d_i \mid i = 1, \ldots, r\}$,
- $\{e_{ij} \mid i = 1, \ldots, m, j = 1, \ldots, s\}$,
- $\{f_i \mid i = 1, \ldots, l\}$.

Now, for each $i = 1, \ldots, m$, let $\phi_i : \{1, 2\}^s \to \mathcal{K}_{\{c_i\}\sqsubseteq C_i}$ be a surjective function from s-ary binary numbers to $\mathcal{K}_{\{c_i\}\sqsubseteq C_i}$, which exists due to our choice of s. Moreover, for each $i = 1, \ldots, m$, let $\psi_i = \langle h, k\rangle$ be a pair of distinct numbers $h, k \in \{1, \ldots, r\}, h \neq k$ such that $\psi_i \neq \psi_j$ whenever $i \neq j$. This choice is possible since there are $r(r - 1)$ such pairs and $r(r - 1) \geq m$ was assumed. Given any j-ary tuple θ, we use $\theta(k)$ to denote the kth component of θ for $k = 1, \ldots, j$. In particular, we use the notation $\psi_i(\nu(j))$ ($i = 1, \ldots, m$, $j = 1, \ldots, s$) with tuples $\nu \in \{1, 2\}^s$ below.

Let B be a fresh concept name – we will use it to mark certain distinct R-successors that the datalog program must ensure to exist. We construct the following datalog rules and programs:

(i) for all $e, f \in \{d_1, \ldots, d_r, c_1, \ldots, c_{m+l}\}$ with $e \neq f$:
$B(e) \wedge B(f) \wedge e \approx f \to \bot$,
$B(e) \to R(c, e)$,

(ii) for all $i \in \{1, \ldots, r\}$:
$B(d_i)$,
$\mathsf{dlg}_a(\{d_i\} \sqsubseteq D_1)$,

(iii) for all $i \in \{1, \ldots, m\}$, $\nu \in \{1, 2\}^s$, $h \in \{1, \ldots, s\}$:
$R(c, e_{ih})$,
$\mathsf{dlg}_a(\{e_{ih}\} \sqsubseteq D_1)$,
for all $j \in \{1, \ldots, r\}$, $j \neq \psi_i(1)$, $j \neq \psi_i(2)$: $e_{ih} \approx d_j \to \bot$,
$e_{i1} \approx d_{\psi_i(\nu(1))} \wedge \ldots \wedge e_{is} \approx d_{\psi_i(\nu(s))} \to A(c_i)$ for a fresh concept name A,
$A(c_i) \to B(c_i)$,
$\mathsf{datalog}(\phi_i(\nu))|_{A(c_i)}$,

(iv) for all $i, j \in \{1, \ldots, m\}$ with $i \neq j$, for all $h \in \{1, \ldots, s\}$:
if there is $k \in \{1, 2\}$ such that $\psi_i(k) = \psi_j(k)$: $e_{ih} \approx e_{jh} \to e_{ih} \approx d_{\psi_i(k)}$,
otherwise: $e_{ih} \approx e_{jh} \to \bot$,

(v) for all $i \in \{1, \ldots, l\}$, $j \in \{1, \ldots, r\}$:
$R(c, f_i)$,
$\mathsf{dlg}_a(\{f_i\} \sqsubseteq D_1)$,
$f_i \approx d_j \to A(c_{m+i})$ for a fresh concept name A,
$A(c_{m+i}) \to B(c_{m+i})$,
$\mathsf{dlg}_a(\{c_{m+i}\} \sqsubseteq C_{m+i})|_{A(c_{m+i})}$,

(vi) for all $e \in \{f_1, \ldots, f_l, e_{11}, \ldots, e_{1s}, \ldots, e_{m1}, \ldots, e_{ms}\}$:
for all $f \in \{f_1, \ldots, f_l\}$ with $e \neq f$: $f \approx e \to f \approx d_1$,
for all $f \in \{c_1, \ldots, c_{m+l}\}$: $B(f) \wedge f \approx e \to \bot$.

Now $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined as the union of P_{Inv} and all rules and programs constructed above.

It remains to show that $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ that semantically emulates $\{c\} \sqsubseteq C$. For the one direction, consider any model $\mathcal{I}$ of $\{c\} \sqsubseteq C$. Select n distinct R-successors $\delta_1, \ldots, \delta_n$ of $c^{\mathcal{I}}$ such that $\delta_i \in (D_1 \sqcup D_2 \sqcup D_3)^{\mathcal{I}}$ for all $i = 1, \ldots, n$. By Proposition 7.4.5, for all $i \in \{1, \ldots, m\}$, if $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$ then there is an extended interpretation $\mathcal{I}_i$ such that $\mathcal{I}_i \models \text{KB}_i$ for some $\text{KB}_i \in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$. As in the proof of Lemma 7.4.7 above, we can find an extended interpretation $\mathcal{J}'$ such that $\mathcal{J}' \models \text{KB}_i$. Using a similar argument, we can chose $\mathcal{J}'$ such that $\mathcal{J}' \models \mathsf{dlg}_a(\{c_j\} \sqsubseteq C_j)$ for each $j \in \{m+1, \ldots, m+l\}$ for which $c_j^{\mathcal{I}} \in C_j^{\mathcal{I}}$.

A model $\mathcal{J}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined by further extending $\mathcal{J}'$. At least r elements $\delta \in \{\delta_1, \ldots, \delta_n\}$ must satisfy $\delta \in D_1^{\mathcal{I}}$ – w.l.o.g. we assume that this is the case for $\delta_1, \ldots, \delta_r$. Then set $d_i^{\mathcal{J}} := \delta_i$ for all $i \in \{1, \ldots, r\}$.

Now select an injective function $\sigma : \{1, \ldots, m+l\} \to \{1, \ldots, n\}$ such that $\sigma(i) = j$ if $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$, $c_i^{\mathcal{I}} = \delta_j$ for some $j \in \{1, \ldots, n\}$ and there is no $i' < i$ such that $c_{i'}^{\mathcal{I}} = \delta_j$; and $\sigma(i) \in D_1^{\mathcal{I}}$ otherwise. Such a function clearly exists. Consider some $i \in \{1, \ldots, m\}$. If $\delta_{\sigma(i)}^{\mathcal{I}} \in D_1^{\mathcal{I}}$, then set $e_{ih}^{\mathcal{J}} := \sigma(i)$ for each $h \in \{1, \ldots, s\}$. Otherwise, $\delta_{\sigma(i)} = c_i^{\mathcal{I}}$ and $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$. In this case, let $\nu \in \{1, 2\}^s$ be such that $\text{KB}_i = \phi_i(\nu)$, and define $e_{ih}^{\mathcal{J}} := d_{\psi_i(\nu(h))}^{\mathcal{J}}$ for each $h \in \{1, \ldots, s\}$. Finally, for $i \in \{1, \ldots, l\}$, define $f_i^{\mathcal{J}} := \delta_{\sigma(m+i)}$.

By the assumption of the lemma, for each program of the form $\mathsf{dlg}_a(\{e\} \sqsubseteq D)$ that is constructed in rules (ii), (iii), and (v), we can extend $\mathcal{J}$ to symbols of $\mathsf{dlg}_a(\{e\} \sqsubseteq D)$ so that the respective programs are satisfied. For B we select the smallest extensions $B^{\mathcal{J}}$ for which the rules of (ii), (iii), and (v) that use B are satisfied. It is easy to check that the rules of (i) are satisfied. Similarly, we assign minimal extensions to all auxiliary concept names A introduced in (iii) and (v). Now it is not hard to check that $\mathcal{J}$ satisfies all rules of (i)–(vi) as required.

C	$\mathsf{dlg}_a(\{c\} \sqsubseteq C)$
$D \in \mathbf{D}_H$	$\mathsf{dlg}_H(X \sqsubseteq D) \cup \{X(c)\}$
$D_1 \sqcap D_2$	$\mathsf{dlg}_a(\{c\} \sqsubseteq D_1) \cup \mathsf{dlg}_a(\{c\} \sqsubseteq D_2)$
$D_1 \sqcup D_2 \in (\mathbf{D}_a \sqcup \mathbf{D}_B)$	$\mathsf{dlg}_B(\neg X \sqsubseteq D_2) \cup \mathsf{dlg}_a(\{c\} \sqsubseteq D_1)\vert_{X(c)}$
$\geqslant n\, R.\top$	$\{R(c, a_1), \ldots, R(c, a_n)\} \cup P_{\text{Inv}}$
$\geqslant n\, R.D \quad (D \neq \top)$	$\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ as defined in Lemma 7.4.6, 7.4.7, and 7.4.8
X a fresh concept name, a_i fresh constants	

Figure 7.9: Transforming axioms $\{\mathbf{I}\} \sqsubseteq \mathbf{D}_a$ to datalog

For the other direction, consider any model $\mathcal{I}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. The rules of (ii) establish r distinct R-successors $d_1, \ldots, d_r$ of c that are in D_1. For any $i \in \{1, \ldots, l\}$, the rules of (iv) ensure that f_i is not equal to any c_j in B. The rules of (v) leave two possibilities. Either f_i is equal to some constant d_j, in which case c_{m+i} is an R-successor of c that is in C_{m+i}, and that is distinct from all other c_h and d_h by (i). Or f_i is not equal to any constant d_j or f_h $(h \neq i)$, and thus not equal to any e_{hk} either (vi); so f_i constitutes a new R-successor of c that is in D_1.

For any $i \in \{1, \ldots, m\}$, if some e_{ih} is not equal to $d_{\psi_i(1)}$ or $d_{\psi_i(2)}$, then the rules of (iii) and (iv) ensure that e_{ih} is not equal to any other constant of the form d_j or e_{jk}. Rules (iv) ensure that e_{ih} is also not equal to any constant of the form f_j, and thus e_{ih} constitutes an additional R-successor of c that is in D_1. If no such e_{ih} exists, then a rule of (iii) applies for some $\nu \in \{1, 2\}^s$, implying that $c_i^{\mathcal{I}} \in A^{\mathcal{I}}$ for the respective fresh concept name A. But then the rules of (iii) together with the assumptions of the lemma imply that $\mathcal{I} \models \phi_i(\nu) \in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$. By Proposition 7.4.5, we find that $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$. Rules (i) and (iv) ensure that c_i is distinct from the remaining R-successors. Overall, we thus obtain $r + m + l = n$ distinct R-successors of c that belong to $D_1 \sqcup D_2 \sqcup D_3$. □

Lemma 7.4.9 *Consider a concept $C \in \mathbf{D}_a$ and constant c such that every datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$ ($\mathsf{dlg}_H(X \sqsubseteq D)$) on the right-hand side of Fig. 7.9 semantically emulates $\{c\} \sqsubseteq D$ ($X \sqsubseteq D$). Then the datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ as defined in Fig. 7.9 semantically emulates $\{c\} \sqsubseteq C$.*

Proof. The proof proceeds by induction. The complex cases have already been established in Lemma 7.4.6, 7.4.7, and 7.4.8. The remaining induction steps are very similar to the steps in Lemma 7.4.1 and 7.4.2. □

We can now complete our induction by summarising the previous lemmata.

Proposition 7.4.10 *Consider concepts $C \in \mathbf{D}_H$, $D \in \mathbf{D}_a$, a concept name A, and a constant symbol c. Lemma 7.4.1, 7.4.2, 7.4.6, 7.4.7, 7.4.8, and 7.4.9 together* *define a recursive construction procedure for datalog programs* $\mathsf{dlg}_H(A \sqsubseteq C)$ *and* $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$ *that semantically emulate* $A \sqsubseteq C$ *and* $\{c\} \sqsubseteq D$*, respectively.*

Proof. The mentioned results are the basis for establishing an inductive argument to proof the claim. Lemma 7.4.6, 7.4.7 and 7.4.8 require the existence of certain datalog programs $\mathsf{datalog}(\mathrm{KB})$. For this proof, we define $\mathsf{datalog}(\mathrm{KB}) := \{\mathsf{dlg}_B(\neg A \sqsubseteq E) \mid \neg A \sqsubseteq E \in \mathrm{KB}\} \cup \{\mathsf{dlg}_H(A \sqsubseteq E) \mid A \sqsubseteq E \in \mathrm{KB}\} \cup \{\mathsf{dlg}_a(\{f\} \sqsubseteq E) \mid \{f\} \sqsubseteq E \in \mathrm{KB}\}$ (we provide a more general definition of $\mathsf{datalog}(\mathrm{KB})$ for other forms knowledge bases at the end of this section). According to Lemma 7.4.4 this definition is well and covers all axioms that can occur in KB.

It remains to show that the preconditions of each induction step are indeed satisfied by applying the induction hypothesis that the claim hold for proper subconcepts of the considered concepts. This is obvious whenever preconditions require the claim to hold for programs of the form $\mathsf{dlg}_H(A' \sqsubseteq C')$ or $\mathsf{dlg}_a(\{c'\} \sqsubseteq D')$ where C' and D' are proper subconcepts of C and D, respectively.

The induction steps for $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$, however, need to use Lemma 7.4.6, 7.4.7 which 7.4.8 additionally require that, for a proper subconcept D' of D and some $\mathrm{KB} \in \mathcal{K}_{\{c'\} \sqsubseteq D'}$, the claim holds for all programs $\mathsf{dlg}_H(A \sqsubseteq E)$ with $A \sqsubseteq E \in \mathrm{KB}$ and for all programs $\mathsf{dlg}_a(\{f\} \sqsubseteq E)$ with $\{f\} \sqsubseteq E \in \mathrm{KB}$ (the translations $\mathsf{dlg}_B(\neg A \sqsubseteq E)$ are always given by Lemma 7.4.1). Inspecting Definition 7.4.3, we find that most axioms in knowledge bases of $\mathcal{K}_{\{c'\} \sqsubseteq D'}$ are of the form $C \sqsubseteq D''$ with D'' a proper subconcept of D', so that the induction hypothesis applies. However, all cases other than (1), (2), and (3c) also introduce additional axioms that are not referring to subconcepts. By checking the recursive definitions of these axioms, it is easy to see that the claim holds for all axioms of this form. □

We still need to show that the "propositional" concepts in $\mathbf{D}^{=n}$ can also be emulated in datalog.

Lemma 7.4.11 *For every concept* $C \in \mathbf{D}^{=n}$ *for some* $n \geq 1$*, one can construct a datalog program* $\mathsf{datalog}(C)$ *that semantically emulates* C.

Proof. C is of the form $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_n\} \sqcap C_n)$ with $C_1 \in \mathbf{C}^p_H$ and $C_i \in \mathbf{C}^{=i}_\bot$ for $i = 2, \ldots, n$. It is not hard to see that C is semantically equivalent to $\{c_1\} \sqcap C_1$. This is shown by induction over n. Clearly, all models of C have domains with at most n elements. By Lemma 7.3.6, for all $n > 2$, $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_n\} \sqcap C_n)$ is semantically equivalent to $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_{n-1}\} \sqcap C_{n-1})$, as required.

All models of $\{c_1\} \sqcap C_1$ have a unary domain, so that further simplifications are possible. Given any concept D in DLP normal form, let $\phi(D)$ be the concept that is obtained by exhaustively applying the following rules:

- If D has a subconcept $\geqslant 1\, R.E$, replace this subconcept by $E \sqcap \exists R.\mathsf{Self}$.
- If D has a subconcept $\geqslant m\, R.E$ with $m > 1$, replace this subconcept by $\bot$.

α	$\mathsf{dlg}(\alpha) \setminus P_{\text{Inv}}$
$\mathsf{Ref}(R)$	$\{R(x,x)\}$
$\mathsf{Irr}(R)$	$\{R(x,x) \to \bot\}$
$\mathsf{Sym}(R)$	$\{R(x,y) \to R(y,x)\}$
$\mathsf{Asy}(R)$	$\{R(x,y) \wedge R(y,x) \to \bot\}$
$\mathsf{Dis}(R_1,R_2)$	$\{R_1(x,y) \wedge R_2(x,y) \to \bot\}$
$\mathsf{Tra}(R)$	$\{R(x,y) \wedge R(y,z) \to R(x,z)\}$
$R_1 \circ R_2 \circ \ldots \circ R_n \sqsubseteq R$	$\{R_1(x_0,x_1) \wedge \ldots \wedge R_n(x_{n-1},x_n) \to R(x_0,x_n)\}$

Figure 7.10: Transforming $\mathcal{SROIQ}$ RBox axioms to datalog

– If D has a subconcept $\leqslant m\, R.\neg E$ with $m > 1$, replace this subconcept by $\top$.

It is easy to check that $D \in \mathbf{C}_B^p$ implies $\mathsf{DLPNF}(\phi(D)) \in \mathbf{D}_B$, and that $D \in \mathbf{C}_H^p$ implies $\mathsf{DLPNF}(\phi(D)) \in \mathbf{D}_H$. Clearly, $\{c_1\} \sqcap C_1$ is semantically equivalent to $\{c_1\} \sqcap \phi(C_1)$, which is in turn equivalent to the knowledge base $\{\top \sqsubseteq \{c_1\}, \{c_1\} \sqsubseteq \phi(C_1)\}$. Thus, by Proposition 7.4.10, C is semantically emulated by $\mathsf{datalog}(C) := \{x \approx c_1\} \cup \mathsf{dlg}_a(\{c_1\} \sqsubseteq \mathsf{DLPNF}(\phi(C_1)))$ as long as $\mathsf{DLPNF}(\phi(C_1)) \notin \{\top, \bot\}$. If $\mathsf{DLPNF}(\phi(C_1)) = \top$ set $\mathsf{datalog}(C) := \{\}$. If $\mathsf{DLPNF}(\phi(C_1)) = \bot$ set $\mathsf{datalog}(C) := \{\top \to \bot\}$ (the unsatisfiable rule with empty body and head). □

To obtain the main result of this section, it remains to show that RBox and ABox axioms in $\mathcal{DLP}$ can also be emulated in datalog.

Theorem 7.4.12 *For every $\mathcal{DLP}$ axiom α as in Definition 7.3.5, one can construct a datalog program* $\mathsf{datalog}(\alpha)$ *that semantically emulates α.*

Proof. If α is a TBox axiom of the form $C \sqsubseteq D$, then set $E := \mathsf{DLPNF}(\neg C \sqcup D)$. If $E = \top$ then $\mathsf{datalog}(\alpha) := \{\}$. If $E = \bot$ of $E \in \mathbf{C}_{\neq\top}$ then $\mathsf{datalog}(\alpha) := \{\top \to \bot\}$ (the unsatisfiable rule with empty body and head). It is easy to see, that concepts of the form $\mathbf{C}_{\neq\top}$ are indeed unsatisfiable when used as axioms. If $E \in \mathbf{D}^{=n}$ for some $n \geq 1$ then set $\mathsf{datalog}(\alpha) := \mathsf{datalog}(E)$ as defined in Lemma 7.4.11. Finally, if $E \in \mathbf{D}_H$ then set $\mathsf{datalog}(\alpha) := \mathsf{dlg}_H(A \sqsubseteq E) \cup \{A(x)\}$ as defined in Proposition 7.4.10, where A is a fresh concept name.

If α is an ABox axiom of the form $C(a)$ with $\mathsf{DLPNF}(C) \in \mathbf{D}_a$ then define $\mathsf{datalog}(\alpha) := \mathsf{dlg}_a(\{a\} \sqsubseteq \mathsf{DLPNF}(C))$ as given in Proposition 7.4.10.

If α is an RBox axiom then $\mathsf{dlg}_R(\alpha)$ is obtained as the union of P_{Inv} and the rules given in Fig. 7.10. Set $\mathsf{datalog}(\alpha) := \mathsf{dlg}_R(\alpha)$. It is easy to see that this datalog program satisfies the claim. □

7.5 Model Constructions for Datalog

In this section, we introduce constructions on first-order logic interpretations that will be essential for showing that certain formulae cannot be in DLP. The general approach is to find operations that preserve models for datalog programs, i.e. operations under which the set of models of any datalog program must be closed. A well-known model construction in logic programming is the intersection of two Herbrand models, and it is well-known that Horn logic is closed under such intersections. The next definition generalises intersections in two ways: on the one hand, it uses functions to allow for interpretations with different (non-Herbrand) domains; on the other hand, it allows us to construct additional domain elements as feature combinations of existing elements.

Definition 7.5.1 Consider a datalog signature $\mathscr{S}$ and two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ over that signature. Consider a set Δ and functions $\mu : \Delta \to \Delta^{\mathcal{I}_1}$ and $\nu : \Delta \to \Delta^{\mathcal{I}_2}$ such that, for each constant c in $\mathscr{S}$, there is exactly one element $\delta_c \in \Delta$ for which $\mu(\delta_c) = c^{\mathcal{I}_1}$ and $\nu(\delta_c) = c^{\mathcal{I}_2}$. The *product interpretation* $\mathcal{J} = \mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$ is defined as follows:

- $\Delta^{\mathcal{J}} := \Delta$,
- for each constant c in $\mathscr{S}$, set $c^{\mathcal{J}} := \delta_c$,
- for each n-ary predicate symbol p and n-tuple $\bar{\delta} \in \Delta^n$, set $\bar{\delta} \in p^{\mathcal{J}}$ iff $\mu(\bar{\delta}) \in p^{\mathcal{I}_1}$ and $\nu(\bar{\delta}) \in p^{\mathcal{I}_2}$, where $\mu(\bar{\delta})$ and $\nu(\bar{\delta})$ denote the tuples obtained by applying μ and ν to each component of $\bar{\delta}$. ◇

The previous definition does not imply that constants have distinct interpretations: $\delta_c = \delta_d$ if and only if $c^{\mathcal{I}_1} = d^{\mathcal{I}_1}$ and $c^{\mathcal{I}_2} = d^{\mathcal{I}_2}$. As the definition of equality in product models is similar to the definition of predicate extensions, it is convenient to formulate Definition 7.5.1 for first-order logic without equality, assuming that $\approx$ is introduced by the well-known axiomatisation of its properties as discussed in Section 4.1.3. A direct definition for $\mathbf{FOL}_{\approx}$ is straightforward.

The construction of product interpretation can be considered as a combination of *direct product* and *sub-model* constructions known in model theory [CK90]. The essential property of product interpretations is the following:

Proposition 7.5.2 *Consider a signature $\mathscr{S}$, interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$, and functions $\mu : \Delta \to \Delta^{\mathcal{I}_1}$ and $\nu : \Delta \to \Delta^{\mathcal{I}_2}$ as in Definition 7.5.1. Then, for every datalog program P over $\mathscr{S}$, we find that $\mathcal{I}_1 \models P$ and $\mathcal{I}_2 \models P$ implies $\mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2 \models P$.*

Proof. Let $\mathcal{J} := \mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$. Consider any rule $B \to H$ in P, and a variable assignment $\mathcal{Z}$ for $\mathcal{J}$ such that $\mathcal{J}, \mathcal{Z} \models B$. Define a variable assignment $\mathcal{Z}_1$ for $\mathcal{I}_1$ by setting $\mathcal{Z}_1(x) := \mu(\mathcal{Z}(x))$. By Definition 7.5.1, it is easy to see that $\mathcal{I}_1, \mathcal{Z}_1 \models B$, and thus $\mathcal{I}_1, \mathcal{Z}_1 \models H$. Analogously, we construct a variable assignment $\mathcal{Z}_2$ such that $\mathcal{I}_2, \mathcal{Z}_2 \models B$ and $\mathcal{I}_2, \mathcal{Z}_2 \models H$. It is easy to see that this implies $\mathcal{J}, \mathcal{Z} \models H$ as required. □

A well-known special case of the above product construction is obtained for $\Delta = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ with μ and ν being the projections to the first and second component of each pair in Δ. It turns out that this canonical product construction is not sufficient to detect all cases of knowledge bases that cannot be $\mathbf{FOL}_{\approx}$-emulated in datalog. For example, the set of models of the non-DLP axiom $\{a\} \sqsubseteq \geqslant 2\, R.(\neg\{b\} \sqcup \leqslant 1\, S.\neg A)$ is closed under canonical products. The more general construction above is needed to address such cases.

When using Proposition 7.5.2 to show that a knowledge base cannot be $\mathbf{FOL}_{\approx}$-emulated in datalog, it must be taken into account that $\mathbf{FOL}_{\approx}$-emulation is not as strong as semantic equivalence. It is not sufficient to show that the models of a knowledge base are not closed under products. For example, the DLP axiom $\{a\} \sqsubseteq \geqslant 1\, R.\top$ has a model $\mathcal{I}$ with domain $\Delta^{\mathcal{I}} := \{a, x\}$, $a^{\mathcal{I}} := a$, and $R^{\mathcal{I}} := \langle a, x\rangle$. Yet, the function $\mu : \{a\} \to \{a, x\}$ with $\mu(a) = a$ can be used to construct an interpretation $\mathcal{I} \times_{\mu,\mu} \mathcal{I}$ that is not a model of the axiom. Note that all preconditions of Definition 7.5.1 are satisfied. Proposition 7.5.2 allows us to conclude that there is no datalog program that is semantically equivalent to $\{a\} \sqsubseteq \geqslant 1\, R.\top$, but not that there is no such program $\mathbf{FOL}_{\approx}$*-emulating* the axiom. To show that a knowledge base cannot even be emulated in datalog, we therefore use the following observation.

Lemma 7.5.3 *Consider a knowledge base* KB *over some signature $\mathscr{S}$. If there are $\mathbf{FOL}_{\approx}$ theories T_1 and T_2 over $\mathscr{S}$ such that:*

- KB $\cup\, T_1$ *and* KB $\cup\, T_2$ *are satisfiable, and*
- *for every pair of models $\mathcal{I}_1 \models$ KB$\cup T_1$ and $\mathcal{I}_2 \models$ KB$\cup T_2$, possibly based on an extended signature $\mathscr{S}'$, there are functions μ and ν such that $\mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2 \not\models$* KB,

then KB *cannot be $\mathbf{FOL}_{\approx}$-emulated in datalog.*

If $T_1 = T_2$ then this conclusion can also be obtained if the precondition only holds for pairs of equal models $\mathcal{I}_1 = \mathcal{I}_2$.

Proof. For a contradiction, suppose that the preconditions of the lemma hold and there is a datalog program P that $\mathbf{FOL}_{\approx}$-emulates KB. Then $P \cup$ KB $\cup\, T_i$ is satisfied by some model $\mathcal{I}_i$ of P for each $i = 1, 2$, where the relevant signature of P may be larger than the signature of KB. Let $\mathcal{J} = \mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$ denote the product interpretation from the second condition. Applying Proposition 7.5.2, we find that $\mathcal{J}$ is a model of P that is not a model of KB. But then the union of P with a $\mathbf{FOL}_{\approx}$ formula of $\mathscr{S}$ that is semantically equivalent to the negation of the conjunction of all axioms in KB is satisfiable, contradicting the supposed $\mathbf{FOL}_{\approx}$-emulation. The last part of the claim is obvious. □

The optional extension of the signature in the previous lemma can be important since the preconditions of Definition 7.5.1 require that the domain of the constructed model contains elements for all constant symbols.

As a simple example for this approach, we show that $\mathrm{KB} = \{\top \sqsubseteq A \sqcup B\}$ cannot be $\mathbf{FOL}_{\approx}$-emulated in datalog. Define auxiliary knowledge bases $\mathrm{KB}_1 = \{A \sqsubseteq \bot\}$ and $\mathrm{KB}_2 = \{B \sqsubseteq \bot\}$. Clearly, $\mathrm{KB} \cup \mathrm{KB}_1$ and $\mathrm{KB} \cup \mathrm{KB}_2$ are satisfied by some models $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. However, it is easy to see that no product of $\mathcal{I}_1$ and $\mathcal{I}_2$ can be a model of KB – independent of the choice of μ and ν – since the extensions of A and B must always be empty in such a product.

Of course there are other examples for which μ and ν must be chosen more carefully. In particular, it is sometimes necessary to restrict the amount of new elements that are introduced by the product. The following definition provides a useful notation for such a restricted form of products that will be sufficient for most applications:

Definition 7.5.4 Consider interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ over a signature $\mathscr{S}$, and let $\mathbf{I}$ be the set of constants in $\mathscr{S}$. Given a set $S \subseteq \mathbf{I} \times \mathbf{I}$, functions $\mu : \Delta \to \Delta^{\mathcal{I}_1}$ and $\nu : \Delta \to \Delta^{\mathcal{I}_2}$ are defined as follows:

- $\Delta := S \cup \{\langle c, c\rangle \mid c \in \mathbf{I}\}$,
- $\mu(\langle c, d\rangle) := c^{\mathcal{I}_1}$,
- $\nu(\langle c, d\rangle) := d^{\mathcal{I}_2}$.

$\mathcal{I}_1 \times_S \mathcal{I}_2$ denotes the product interpretation $\mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$ for these functions. ◇

A special aspect of the previous definition is that it restricts attention to named elements – elements that are represented by some individual name – in the original models. It is an easy corollary of Proposition 7.5.2 that all other elements are indeed irrelevant for satisfying a datalog program.

7.6 Showing Structural Maximality of $\mathcal{DLP}$

In this section, we show that the earlier definition of $\mathcal{DLP}$ is indeed maximal for the underlying principles. The proof mainly uses the principle of structurality (DLP 6) due to which it suffices to show that structural concept expressions that are not in $\mathcal{DLP}$ cannot be $\mathbf{FOL}_{\approx}$-emulated in datalog. To this end, we generally use the strategy suggested by Lemma 7.5.3. The below discussions often use datalog rules or DL axioms in the context of first-order logic to conveniently denote an arbitrary $\mathbf{FOL}_{\approx}$ theory of the same semantics, as obtained by any of the standard translations. Especially, this abbreviated form never refers to the more complex datalog transformation of $\mathcal{DLP}$ concepts, and it is only used when syntactic details are not relevant. Moreover, we assume that $\approx$ always denotes the equality predicate, and do not explicitly provide an axiomatisation for it.

The outline of the proof is as follows. We start by specifying some useful kinds of auxiliary datalog programs in Definition 7.6.1 and 7.6.2. The first major class of concept expressions is excluded by Proposition 7.6.6 which shows that concepts that are not in $\mathbf{D}_a^+$ can usually not be emulated in datalog. This result is prepared by Lemma 7.6.3, Lemma 7.6.4, and Lemma 7.6.5. These lemmata also are of some utility later on, since they can be used to exclude most forms of existential statements from DLP.

The second main ingredient of the maximality proof is Corollary 7.6.9. It extends Proposition 7.6.6 by establishing that concepts can typically not be emulated in datalog if they are not in $\mathbf{D}_H$. The chief insight that leads to this result is formulated in Lemma 7.6.8 which sports the most complex proof of this section. After this, it is comparatively easy to establish Lemma 7.6.10 to treat some pathological cases that had been excluded from the earlier considerations. In particular, it includes the "propositional" case where a DL concept enforces a unary interpretation domain.

The outcomes of Proposition 7.6.6, Corollary 7.6.9, and Lemma 7.6.10 are finally summarised in the main Theorem 7.6.11.

To pursue the proof strategy outlined by Lemma 7.5.3, our main work consists in specifying suitable auxiliary theories T_1 and T_2. To simplify this task, we first define some auxiliary theories that will be used frequently. Many of these constructions have the additional advantage of being in datalog – with the important consequence that they are still satisfied by product interpretations (Proposition 7.5.2). Often this is relevant for showing that said product interpretations cannot satisfy a given non-DLP concept.

Whereas many concept expressions C cannot be $\mathbf{FOL}_{\approx}$-emulated in datalog, it is usually possible to specify a datalog program that entails $\{c\} \sqsubseteq C$ for a given constant c by specifying sufficient properties that c must satisfy for this to be true. This only fails if C is structurally unsatisfiable. The below construction generalises this idea to any number of constants, and to the dual case where $\{c\} \sqsubseteq \neg C$ is entailed. The constructions in Definition 7.6.1 and 7.6.2 should be compared to the simpler cases discussed in Definition 7.2.5 which serve essentially the same purpose for $\mathcal{ALC}$.

Definition 7.6.1 Consider a name-separated concept C in positive normal form, and individual names $c_0, \ldots, c_n$ for $n \geq 0$.
If $C \notin \mathbf{L}_{\leq n}$ the datalog program $[\![c_0, \ldots, c_n \in C]\!]$ is defined recursively as follows:

- If $C = \top$ or $C = \geqslant 0\, R.D$ then $[\![c_0, \ldots, c_n \in C]\!] := \emptyset$.
- If $C = \{d\}$ then $n = 0$ and $[\![c_0 \in C]\!] := \{c_0 \approx d\}$.
- If C is of the form $\mathbf{A}$, $\neg\mathbf{A}$, $\neg\{\mathbf{I}\}$, $\exists\mathbf{R}.\mathsf{Self}$, or $\neg\exists\mathbf{R}.\mathsf{Self}$, then $[\![c_0, \ldots, c_n \in C]\!] := \bigcup_{0 \leq i \leq n} \mathsf{datalog}(\{c_i\} \sqsubseteq C)$.
- $C = D_1 \sqcap D_2$, then $D_i \notin \mathbf{L}_{\leq n}$ for $i = 1, 2$, and $[\![c_0, \ldots, c_n \in C]\!] := [\![c_0, \ldots, c_n \in D_1]\!] \cup [\![c_0, \ldots, c_n \in D_2]\!]$.
- If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{L}_{\leq n}$, then $[\![c_0, \ldots, c_n \in C]\!] := [\![c_0, \ldots, c_n \in D_1]\!]$.
- If $C = D_1 \sqcup D_2$ with $D_1 \in \overline{\mathbf{L}}_{\leq m'}$ and $D_1 \in \overline{\mathbf{L}}_{\leq m''}$ such that $m' + m'' = n - 1$, then $[\![c_0, \ldots, c_n \in C]\!] := [\![c_0, \ldots, c_{m'} \in D_1]\!] \cup [\![c_{m'+1}, \ldots, c_{m'+m''+1} \in D_2]\!]$.
- If $C = \geqslant m\, R.D$ with $m \geq 1$, consider fresh constants $d_0, \ldots, d_m$, and define $[\![c_0, \ldots, c_n \in C]\!] := [\![d_0, \ldots, d_m \in D]\!] \cup \{R(c_i, d_j) \mid 0 \leq i \leq n,\ 0 \leq j \leq m\} \cup \{d_i \approx d_j \to \bot \mid 0 \leq i < j \leq m\}$.
- If $C = \leqslant m\, R.\neg D$, then $[\![c_0, \ldots, c_n \in C]\!] := \{x \approx c_i \wedge R(x, y) \to \bot \mid 1 \leq i \leq n\}$.

If $C \notin \mathbf{L}_{\geq \omega - n}$ then define a datalog program $[\![c_0, \ldots, c_n \notin C]\!] := [\![c_0, \ldots, c_n \in \mathsf{pNNF}(\neg C)]\!]$. ◇

Note that the given cases directly follow the definition of $\overline{\mathbf{L}}_{\leq n}$ in Fig. 7.4. Also note that $[\![c_0, \ldots, c_n \in C]\!]$ and $[\![c_0, \ldots, c_n \notin C]\!]$ are satisfiable, even if we additionally require that all constants c_i are mutually unequal (which is not implied by the datalog programs).

Definition 7.6.1 can be viewed as a way to entail statements of the form $\{c_0\} \sqcup \ldots \sqcup \{c_n\} \sqsubseteq C$ if $C \notin \mathbf{L}_{\leq n}$. For cases where C is not in $\mathbf{L}_{\leq n}$ for any $n \geq 0$ this approach can be generalised to entail statements of the form $D \sqsubseteq C$ for a more general class of concepts D. The necessary construction is provided by the following definition which is very similar to Definition 7.6.1. We provide an alternative perspective and specify the dual case – entailing $C \sqsubseteq D$ in cases where $C \notin \mathbf{L}_{\geq \omega - n}$ for all $m \geq 0$ – which is the only case that is needed in our subsequent arguments.

Definition 7.6.2 Consider a name-separated concept C in positive normal form, and a concept $D \in \mathbf{D}_H$.

If $C \notin \mathbf{L}_{\geq \omega - m}$ for any $m \geq 0$, the datalog program $[\![C \sqsubseteq D]\!]_{\leq}$ is defined recursively as follows:

- If $C = \bot$ then $[\![C \sqsubseteq D]\!]_{\leq} := \emptyset$.
- If C is of the form $\mathbf{A}$, $\{\mathbf{I}\}$, or $\exists\mathbf{R}.\mathsf{Self}$, then $[\![C \sqsubseteq D]\!]_{\leq} := \mathsf{datalog}(C \sqsubseteq D)$.
- If C is of the form $\neg\mathbf{A}$, or $\neg\exists\mathbf{R}.\mathsf{Self}$, then $[\![C \sqsubseteq D]\!]_{\leq} := \mathsf{datalog}(C \sqsubseteq \bot)$.
- If $C = D_1 \sqcup D_2$, then $D_i \notin \mathbf{L}_{\geq \omega - m}$ for any $m \geq 0$ $(i = 1, 2)$, and $[\![C \sqsubseteq D]\!]_{\leq} := [\![D_1 \sqsubseteq D]\!]_{\leq} \cup [\![D_2 \sqsubseteq D]\!]_{\leq}$.

- If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{L}_{\geq\omega-m}$ for any $m \geq 0$, then $[\![C \sqsubseteq D]\!]_{\leq} := [\![D_1 \sqsubseteq D]\!]_{\leq}$.
- If $C = \leqslant m\, R.\neg E$, consider fresh constants $d_0, \ldots, d_m$, and set $[\![C \sqsubseteq D]\!]_{\leq} := [\![d_0, \ldots, d_m \notin E]\!] \cup \{d_i \approx d_j \to \bot \mid 0 \leq i < j \leq m\} \cup \bigcup_{0 \leq i \leq m} \mathsf{datalog}(\geqslant 1\, R.\{d_i\})$.
- If $C = \geqslant m\, R.E$, then $[\![C \sqsubseteq D]\!]_{\leq} := \mathsf{datalog}(\geqslant 1\, R.\top \sqsubseteq D)$. ◇

It should be noted that the cases of the definition are indeed exhaustive. Also observe that $[\![C \sqsubseteq D]\!]_{\leq}$ is always satisfiable, where $D \neq \bot$ is important to ensure that this is actually true for cases like $[\![\{c\} \sqsubseteq D]\!]_{\leq}$. This also shows that $[\![C \sqsubseteq A]\!]_{\leq} \cup \{A \sqsubseteq \bot\}$ cannot be assumed to be satisfiable in general.

Some further observations should be made in order to understand how Definitions 7.6.1 and 7.6.2 can be used when discussing datalog emulation. The constructions in both cases do certainly not $\mathbf{FOL}_{\approx}$-emulate the statement that they entail. For example, $[\![c \in C]\!]$ enforces one particular case for which $\{c\} \sqsubseteq C$; it does in general not describe all such cases. Moreover, the program $[\![C \sqsubseteq D]\!]_{\leq}$ may enforce a much stronger condition such as $C \sqsubseteq \bot$ as in the case of $C = \leqslant m\, R.\neg E$. This illustrates that the extension of C can be constrained by $[\![C \sqsubseteq D]\!]_{\leq}$. Conversely, a knowledge base $[\![A \sqcup B \sqsubseteq D]\!]_{\leq}$ might entail the stronger statement $A \sqsubseteq D$.

Luckily, as long as structurality is assumed, the knowledge bases of Definition 7.6.1 and 7.6.2 hardly semantically interact with concept expressions other than those that they are constructed from. Yet, it must be noted that $[\![c_0, \ldots, c_n \in C]\!]$ may introduce mutually unequal individuals d_i for the case $C = \geqslant m\, R.D$, and that two distinct individuals are already required if $C = \neg\{d\}$. This effect can occur for all of the above constructions. Logical theories in $\mathbf{FOL}_{\approx}$ can restrict the maximum size of the domain, and the same is accomplished by DL axioms that correspond to concept expressions in $\mathbf{L}_{\leq m}$ for some $m \geq 0$. We need to exclude this possibility when using the above definitions.

The previous discussion shows that it is important to carefully check all uses of Definitions 7.6.1 and 7.6.2 to avoid undesired semantic ramifications. A useful intuition is that the constructed theories enforce a simplification upon C that allows us to disregard the concept's internal structure. As an example of a typical usage of these constructions, consider the axiom $\alpha = \{a\} \sqsubseteq C_1 \sqcup C_2$ with $C_2 \notin \mathbf{L}_{\top}$. Then $\alpha \cup [\![a \notin C_2]\!]$ implies $\{\{a\} \sqsubseteq C_1\}$.[3] So $[\![a \notin C_2]\!]$ allowed us to dismiss an "uninteresting" C_2 to focus on the impact of C_1.

The following lemmata use the product construction to create elements that are not in a given concept's extension, where we usually use the abbreviated product construction of Definition 7.5.4. In the weakest case, elements outside the extension must be provided to achieve this (Lemma 7.6.3). With stronger side

[3]This implication is not quite a $\mathbf{FOL}_{\approx}$-emulation since $[\![a \notin C_2]\!]$ can require a minimal domain cardinality, as discussed above.

conditions, some or even all of the elements can be part of the concept extension (Lemma 7.6.4 and 7.6.5). The lemmata are essential ingredients for showing that subconcepts that are not in $\mathbf{D}_a^+$ cannot occur in any DLP concept that is in normal form, and the assumptions of the lemma are therefore motivated by the definition of $\mathbf{D}_a^+$.

Lemma 7.6.3 *Consider a name-separated concept C in DLP normal form such that $C \neq \bot$ and $C \notin \mathbf{D}_{\geq\omega-n}$ for all $n \geq 0$ (in particular $C \neq \top$). Let $c_0, \dots, c_n$ be fresh constants. There is a consistent datalog program $[\![c_0, \dots, c_n \notin C]\!]_\times$ such that*

- $[\![c_0, \dots, c_n \notin C]\!]_\times \models \neg(c_i \approx c_j)$ *for all $i, j \in \{0, \dots, n\}$ with $i \neq j$,*
- $[\![c_0, \dots, c_n \notin C]\!]_\times \models \{c_i\} \sqsubseteq \neg C$ *for all $i = 0, \dots, n$,*
- *for all models $\mathcal{I}_1$, $\mathcal{I}_2$ of $[\![c_0, \dots, c_n \notin C]\!]_\times$, and any set of constants $N \subseteq \mathbf{I}$ with $\{c_0, \dots, c_n\} \subseteq N$, the product $\mathcal{J} = \mathcal{I}_1 \times_{(N\times N)} \mathcal{I}_2$ is such that $\langle c_i, c_j\rangle \notin C^{\mathcal{J}}$ for all $i, j \in \{0, \dots, n\}$.*

Proof. Using a fresh concept name A, we define $[\![c_0, \dots, c_n \notin C]\!]_\times := [\![C \sqsubseteq \neg A]\!]_\leq \cup \{A(c_i) \mid 0 \leq i \leq n\} \cup \{c_i \approx c_j \to \bot \mid 0 \leq i < j \leq n\}$. Given models $\mathcal{I}_1$ and $\mathcal{I}_2$ of $[\![c_0, \dots, c_n \notin C]\!]_\times$, and $\mathcal{J} = \mathcal{I}_1 \times_{(N\times N)} \mathcal{I}_2$, we find that $\langle c_i, c_j\rangle \in A^{\mathcal{J}}$ for all $i, j \in \{0, \dots, n\}$. Since $[\![c_0, \dots, c_n \notin C]\!]_\times$ is in datalog, it is satisfied by $\mathcal{J}$, and thus we conclude $\langle c_i, c_j\rangle \notin C^{\mathcal{J}}$ for all $i, j \in \{0, \dots, n\}$ as required. □

The next lemma considers concepts $C \notin \mathbf{D}_B^+$. The lemma is also stated for sets of individuals, and additional care is now needed to ensure that it is possible for C to have a set of (distinct) instances. It is not enough to assume $C \notin \mathbf{D}_{\leq n}$ for some or all $n \geq 0$ since this pre-condition cannot be preserved by all recursive constructions. Namely, the recursion in the case $C = D_1 \sqcup D_2$ must be based on the one subconcept D_i for which we have $D_i \notin \mathbf{D}_B^+$, but there is no reason for $D_i \notin \mathbf{D}_{\leq n}$ to hold for any $n \geq 1$ (only $n = 0$ is excluded since C is in DLP normal form). This explains why the lemma considers multiple individuals $c_0, \dots, c_n$ only in cases where this problem can be avoided.

Lemma 7.6.4 *Consider a name-separated concept C in DLP normal form such that $C \notin \mathbf{D}_B^+$, and C does not have a subconcept $D \notin \mathbf{D}_a^+$. Let $n \geq 0$ be such that $n = 0$ if C is a disjunction or $C \in \mathbf{D}_{\leq k}$ for some $k \geq 0$, and consider fresh constants $c_0, \dots, c_n, d_0, \dots, d_m$. There is a consistent datalog program $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times$ and according set $M := \{c_0, \dots, c_n, d_0, \dots, d_m\} \cup \{c \in \mathbf{I} \mid c$ occurs in $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times\}$ such that*

- $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \models \neg(e \approx f)$ *for all $e, f \in \{c_0, \dots, c_n, d_0, \dots, d_m\}$ with $e \neq f$,*
- $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \models \{c_i\} \sqsubseteq C$ *for all $i = 0, \dots, n$,*

- $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times \models \{d_i\} \sqsubseteq \neg C$ *for all* $i = 0, \ldots, m$,
- *for all models* $\mathcal{I}_1$, $\mathcal{I}_2$ *of* $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times$, *and any set of constants* $N \subseteq \mathbf{I}$ *with* $M \subseteq N$, *the product* $\mathcal{J} = \mathcal{I}_1 \times_{(N \times N)} \mathcal{I}_2$ *is such that* $\langle c_i, d_j \rangle \notin C^{\mathcal{J}}$ *for all* $i \in \{0, \ldots, n\}$ *and* $j \in \{0, \ldots, m\}$.

Proof. Note that the conditions imply that $C \in \mathbf{D}_a^+$, and hence $C \notin \{\top, \bot\}$. Set $P := \{e \approx f \to \bot \mid e, f \in \{c_0, \ldots, c_n, d_0, \ldots, d_m\}, e \neq f\}$. We define $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times$ recursively based on the structure of C, and we inductively show that it has the required properties. Both parts can conveniently be interleaved. Thus, in each of the following cases, let $\mathcal{I}_1$ and $\mathcal{I}_2$ be models of the $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times$ just defined, and let $\mathcal{J}$ be the product interpretation as in the claim:

- If C has the form $\mathbf{A}$, $\{\mathbf{I}\}$ or $\exists \mathbf{R}.\mathsf{Self}$, then $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times := P \cup [\![c_0, \ldots, c_n \in C]\!] \cup [\![d_0, \ldots, d_m \notin C]\!]$.

  It is easy to see that $\mathcal{J}$ satisfies the claim. Note that the pre-conditions of the lemma imply $n = 0$ whenever $C \in \{\mathbf{I}\}$.

- If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_B^+$, then $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times := [\![c_0, \ldots, c_n \in D_1, d_0, \ldots, d_m \notin D_1]\!]_\times$.

  Since $\mathcal{I}_1$ and $\mathcal{I}_2$ are models of $[\![c_0, \ldots, c_n \in D_1, d_0, \ldots, d_m \notin D_1]\!]_\times$, the claim follows immediately by induction.

- If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{C}_B^+$ and $D_2 \notin \mathbf{D}_{\geq \omega - k}$ for all $k \geq 0$, then $n = 0$ is required. Define $[\![c_0 \in C, d_0, \ldots, d_m \notin C]\!]_\times := [\![c_0 \in D_1, d_0, \ldots, d_m \notin D_1]\!]_\times \cup [\![D_2 \sqsubseteq \{c_0\}]\!]_\leq$.

  $\mathcal{I}_1$ and $\mathcal{I}_2$ are models of $[\![c_0, \ldots, c_n \in D_1, d_0, \ldots, d_m \notin D_1]\!]_\times$ and we can apply the induction hypothesis. The desired result follows since the product $\mathcal{J}$ also satisfies the datalog program $[\![D_2 \sqsubseteq \{c_0\}]\!]_\leq$.

- If $C = \geqslant k\, R.D$ with $k \geq 1$, then $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times := P \cup \{R(c_i, e_j) \mid 0 \leq i \leq n, 1 \leq j \leq k\} \cup [\![e_1, \ldots, e_k \in D]\!] \cup \{R(d_i, x) \to \bot \mid 0 \leq j \leq m\}$ for fresh individual names $e_1, \ldots, e_k$.

  It is again easy to see that $\mathcal{J}$ satisfies the claim.

- If $C = \leqslant 0\, R.\neg D$ with $D \notin \mathbf{D}_B^+$, then, for a fresh constant e, define $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times := P \cup \{R(c_i, x) \to x \approx e, R(c_i, e) \mid 0 \leq i \leq n\} \cup \{R(d_i, f) \mid 0 \leq i \leq m\} \cup [\![e \in D, f \notin D]\!]_\times$.

  We find that $\langle \langle c_i, d_j \rangle, \langle e, f \rangle \rangle \in R^{\mathcal{J}}$ for all $i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, m\}$. The claim follows from the induction hypothesis.

– If $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_{\geq\omega-k}$ for all $k \geq 0$, then consider fresh individuals e, f, g. Define $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times := P \cup \{R(c_i, x) \to x \approx e, R(c_i, e) \mid 0 \leq i \leq n\} \cup \{R(d_i, f), R(d_i, g) \mid 0 \leq i \leq m\} \cup [\![e, f, g \notin D]\!]_\times$. Note that the last component of this union also requires that the individuals denoted by e, f, g are mutually distinct.

  We find that $\langle\langle c_i, d_j\rangle, \langle e, f\rangle\rangle \in R^{\mathcal{J}}$ and $\langle\langle c_i, d_j\rangle, \langle e, g\rangle\rangle \in R^{\mathcal{J}}$ for all $i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, m\}$. The claim follows from Lemma 7.6.3.

It should be verified that the given cases are exhaustive. Especially, $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_{\geq\omega-k}$ for all $k \geq 0$ is the only case where $C = \leqslant k\, R.\neg D$ for some $k \geq 1$ – all other forms are either in $\mathbf{D}^+_B$ or not in $\mathbf{D}^+_a$. Moreover, all recursive applications of the construction satisfy the necessary pre-conditions, especially the requirements for $n \geq 1$ are preserved. □

The third and final lemma in this series is only needed for two individuals so that we can simplify our presentation slightly. However, the construction now becomes more complex since we can no longer use an auxiliary datalog theory, and since more care is needed in selecting a suitable product interpretation.

Lemma 7.6.5 *Consider a name-separated concept C in DLP normal form such that $C \notin \mathbf{D}^+_H$, and C does not have a subconcept $D \notin \mathbf{D}^+_a$. Let c_0, c_1 be fresh constants. There is a consistent first-order theory $[\![c_0, c_1 \in C]\!]_\times$ and a set of constants $N \subseteq \mathbf{I}$ such that*

– $[\![c_0, c_1 \in C]\!]_\times \models \neg(c_0 \approx c_1)$,
– $[\![c_0, c_1 \in C]\!]_\times \models \{c_i\} \sqsubseteq C$ *for* $i = 0, 1$,
– *for all models $\mathcal{I}$ of $[\![c_0, c_1 \in C]\!]_\times$, the product $\mathcal{J} = \mathcal{I} \times_{(N\times N)} \mathcal{I}$ is such that $\langle c_0, c_1\rangle \notin C^{\mathcal{J}}$.*

Proof. The conditions again imply that $C \in \mathbf{D}^+_a$, and hence $C \notin \{\top, \bot\}$. Moreover, $C \notin \mathbf{D}^+_H$ and $C \in \mathbf{D}^+_a$ implies that $C \notin \mathbf{D}_{\leq 1}$. Indeed, $C \notin \mathbf{D}_{\leq 0}$ since C is in DLP normal form, and thus $C \in \mathbf{D}_{\leq 1}$ would imply that C is of the form $\{\mathbf{I}\} \sqcap \mathbf{C}^+_a \subseteq \mathbf{D}^+_{1!} \subseteq \mathbf{D}^+_H$. This property is inherited by subconcepts D of C as long as $D \notin \mathbf{D}^+_H$.

We define $[\![c_0, c_1 \in C]\!]_\times$ recursively based on the structure of C, and we inductively show that it has the required properties. Both parts can conveniently be interleaved. In addition, we also specify a suitable set N of constant symbols to use in the product construction in the recursion. Thus, in each of the following cases, let $\mathcal{I}$ be a model of the $[\![c_0, \ldots, c_n \in C]\!]_\times$ just defined, and let $\mathcal{J}$ be the product interpretation as in the claim.

– If $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}^+_B$ then $[\![c_0, c_1 \in C]\!]_\times := [\![c_0 \in D_1, c_1 \notin D_1]\!]_\times \cup [\![c_1 \in D_2, c_0 \notin D_2]\!]_\times$ and the set N is defined as in Lemma 7.6.4.

  Using Lemma 7.6.4, it is easy to see that $\mathcal{J}$ satisfies the claim.

- If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}^+_H$ and $D_2 \in \mathbf{D}^+_B$ then consider a fresh concept name A. Since $C \notin \mathbf{D}_{\geq\omega-n}$ for all $n \geq 0$, the same holds for D_1 and D_2. Moreover, $D_1 \notin \mathbf{D}_{\leq 1}$ as discussed initially. We thus can define $[\![c_0, c_1 \in C]\!]_\times := [\![c_0, c_1 \in D_1]\!]_\times \cup [\![D_2 \sqsubseteq \neg A]\!]_\leq \cup \{A(c_0), A(c_1)\}$. The set N is defined to be the same as for $[\![c_0, c_1 \in D_1]\!]_\times$.

  $\mathcal{I}$ is a model of $[\![c_0, c_1 \in D_1]\!]_\times$ and we can apply the induction hypothesis. The desired result follows since the product $\mathcal{J}$ also satisfies the datalog program $[\![D_2 \sqsubseteq \neg A]\!]_\leq \cup \{A(c_0), A(c_1)\}$ (Proposition 7.5.2).

- If $C = D_1 \sqcap D_2$ then we can assume $D_1 \notin \mathbf{D}^+_H$. Clearly, $C \notin \mathbf{D}_{\leq 1}$ implies $D_1, D_2 \notin \mathbf{D}_{\leq 1}$. Thus we can set $[\![c_0, c_1 \in C]\!]_\times := [\![c_0, c_1 \in D_1]\!]_\times \cup [\![c_0, c_1 \in D_2]\!]$, where N is again taken to be the set of constants as defined for $[\![c_0, c_1 \in D_1]\!]_\times$.

  We can again apply the induction hypothesis since $\mathcal{I} \models [\![c_0, c_1 \in D_1]\!]_\times$, and use the fact that $\mathcal{J} \models [\![c_0, c_1 \in D_2]\!]$.

- If $C = \geqslant n\, R.D$ then $D \notin \mathbf{D}^+_{n!} \cup \mathbf{D}_{\leq n-1} \cup \{\bot\}$. Since all subconcepts of C are assumed to be in $\mathbf{D}^+_a$, we conclude that $D \notin \mathbf{D}_{\leq n}$. Thus we can introduce fresh individual symbols $d_0, \ldots, d_n$ and set $[\![c_0, c_1 \in C]\!]_\times := [\![d_0, \ldots, d_n \in D]\!] \cup \{\neg(e \approx f) \mid e, f \in \{c_0, c_1, d_0, \ldots, d_n\}, e \neq f\} \cup \{\forall x.R(c_0, x) \leftrightarrow \bigvee_{0 \leq i < n} x \approx d_i\} \cup \{\forall x.R(c_1, x) \leftrightarrow \bigvee_{0 < i \leq n} x \approx d_i\}$. Define $N := \{c_0, c_1\}$.

  We claim that $\langle c_0, c_1 \rangle \in \Delta^{\mathcal{J}}$ is such that $\langle c_0, c_1 \rangle \notin C^{\mathcal{J}}$. Consider any element $\langle e, f \rangle \in \Delta^{\mathcal{J}}$ such that $\langle \langle c_0, c_1 \rangle, \langle e, f \rangle \rangle \in R^{\mathcal{J}}$. By the construction of $\mathcal{J}$, we have that $\langle c_0^{\mathcal{I}}, e^{\mathcal{I}} \rangle, \langle c_1^{\mathcal{I}}, f^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, and thus $e^{\mathcal{I}} = d_i^{\mathcal{I}}$ and $f^{\mathcal{I}} = d_j^{\mathcal{I}}$ for some $i \in \{0, \ldots, n-1\}, j \in \{1, \ldots, n\}$. Since the constants d_i are unequal to c_0, c_1, this implies that $e, f \notin N$, and thus $e = f = d_i = d_j$. Therefore, $\langle e, f \rangle$ is equal to $d_i^{\mathcal{J}}$ for some $i \in \{1, \ldots, n-1\}$ whenever $\langle \langle c_0, c_1 \rangle, \langle e, f \rangle \rangle \in R^{\mathcal{J}}$, as required for $\langle c_0, c_1 \rangle \notin C^{\mathcal{J}}$.

- If $C = \leqslant 0\, R.\neg D$ with $D \notin \mathbf{D}^+_H$ then define $[\![c_0, c_1 \in C]\!]_\times := [\![c_0, c_1 \in D]\!]_\times \cup \{R(c_0, c_0), R(c_1, c_1)\}$, where N is defined as for $[\![c_0, c_1 \in D]\!]_\times$.

  The claim follows by induction as before.

- If $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_B \cup \{\bot\}$ then $[\![c_1, c_0 \in C]\!]_\times := [\![c_0 \in D, c_1 \notin D]\!]_\times \cup \{R(c_0, c_0), R(c_0, c_1), R(c_1, c_0), R(c_1, c_1)\}$, where N is defined to be the set M as given in Lemma 7.6.4.

  The claim is a consequence of Lemma 7.6.4.

- If $C = \leqslant n\, R.\neg D$ with $n \geq 2$ then consider fresh individual symbols $c_2, \ldots, c_n$ and define $[\![c_0, c_1 \in C]\!]_\times := [\![c_0, c_1 \notin D]\!]_\times \cup [\![c_2, \ldots, c_n \notin D]\!] \cup \{R(c_i, c_j) \mid i \in \{0, 1\}, j \in \{0, \ldots, n\}, i \neq j\} \cup \{\neg(c_i \approx c_j) \mid 0 \leq i < j \leq n\}$, where N is defined to be the set M as given in Lemma 7.6.3.

It is easy to see that $\langle c_0, c_1\rangle$ in $\mathcal{J}$ has at least n distinct R-successors $\langle c_i, c_i\rangle$ ($i = 2, \ldots, n$) and $\langle c_1, c_0\rangle$. The former are not in D since $\mathcal{J}$ satisfies the datalog program $[\![c_2, \ldots, c_n \notin D]\!]$. The latter are not in D by Lemma 7.6.3.

Atomic concepts, nominals, Self restrictions, and their negations do not occur since $C \notin \mathbf{D}^+_H$. □

The previous result is used in the following proposition to show that certain kinds of atmost-concepts are generally excluded from DLP, even if they occur as subconcepts only.

Proposition 7.6.6 *Given a name-separated concept $C \notin \{\top, \bot\}$ in DLP normal form, the following three statements are equivalent:*

- *$C \notin \mathbf{D}^+_a$,*
- *C has a subconcept $D \notin \mathbf{D}^+_a$,*
- *C contains a subconcept $\leqslant k\, S.\neg F$ such that $F \in \mathbf{D}^+_a$ and $F \notin \mathbf{D}_{\geq\omega-l}$ for all $l \geq 0$ and:*

  (a) $k = 0$ and $F \notin \mathbf{D}^+_H \cup \{\bot\}$, or
  (b) $k = 1$ and $F \notin \mathbf{D}^+_B \cup \{\bot\}$, or
  (c) $k \geq 2$.

If these statements hold and, in addition, $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$, and $C \notin \mathbf{C}_{\neq\top}$, then C cannot be $\mathbf{FOL}_\approx$-emulated in datalog.

Proof. Note that the preconditions on C imply that $\{C\}$ is satisfiable. The claimed equivalence is easily verified by considering the grammar for $\mathbf{D}^+_a$ given in Fig. 7.5, where it should be noted that some cases are inherited from $\mathbf{D}^+_H$ and $\mathbf{D}^+_B$. Also observe that $F \in \mathbf{D}^+_a$ is thus equivalent to saying that F has no subconcept $E \notin \mathbf{D}^+_a$.

First, we define an auxiliary theory that requires $\leqslant k\, S.\neg F$ to be non-empty in order for C to be satisfied. As before, we sometimes mix first-order logic and DL to denote an arbitrary $\mathbf{FOL}_\approx$ theory that represents the first-order semantics of this combination. Given a constant symbol c, and a subconcept D of C such that $\leqslant k\, S.\neg F$ is a subconcept of D, we recursively construct a $\mathbf{FOL}_\approx$ theory $T(c, D)$:

- If $D = \leqslant k\, S.\neg F$, then $T(c, D) := \emptyset$.
- If $D = D_1 \sqcap D_2$ with $\leqslant k\, S.\neg F$ a subconcept of D_1, then $T(c, D) := T(c, D_1)$.
- If $D = D_1 \sqcup D_2$ with $\leqslant k\, S.\neg F$ a subconcept of D_1, then $T(c, D) := T(c, D_1) \cup [\![c \notin D_2]\!]$.
- If $D = \geqslant n\, R.D'$, then consider fresh constants $c_1, \ldots, c_n$ and define $T(c, D) := \{\forall x.R(c, x) \to \bigvee_{1\leq i\leq n} c_i \approx x\} \cup T(c_0, D')$.

– If $D = \leqslant n\, R.\neg D'$ (with $R \neq S$), then consider fresh constants $c_0, \ldots, c_n$ and set $T(c, D) \coloneqq \{\bigwedge_{0 \leq i \leq n} R(c, c_i) \wedge \bigwedge_{0 \leq i < j \leq n} \neg(c_i \approx c_j)\} \cup [\![c_1, \ldots, c_n \notin D]\!] \cup T(c_0, D')$.

Note that $T(c, D)$ is satisfiable, due to structurality of C and the fact that the subconcept $\leqslant k\, S.\neg F$ cannot be part of a subconcept of the form $\mathbf{L}_\top$ or $\mathbf{L}_\bot$ since C is in DLP normal form. Now the theory T is defined as $T \coloneqq T(c, C)$ for some fresh constant c. It is easy to see that $T \cup \{C\}$ is satisfiable, and that $T \cup \{C\} \cup \{\leqslant k\, S.\neg F \sqsubseteq \bot\}$ is unsatisfiable.

Consider the case $k = 0$. Let a and b be fresh constants. We use the construction of Lemma 7.6.5 to ensure that every element in the respective product interpretations has an S-successor $\langle a, b\rangle$ in $\neg F$, and N denotes the according set of constant symbols as in the definition of $[\![a, b \in F]\!]_\times$. Some care is needed to ensure that the auxiliary theory T remains true in any such product interpretation. Thus define $T' \coloneqq T \cup \{\neg(c \approx d) \mid c \in N, d \text{ occurs in } T\} \cup \{\forall x.S(x, a) \wedge S(x, b)\} \cup [\![a, b \in F]\!]_\times$. It is not hard to see that $T' \cup \{C\}$ is satisfiable. For an arbitrary model $\mathcal{I}$ of $T' \cup \{C\}$, consider the product interpretation $\mathcal{J} \coloneqq \mathcal{I} \times_{(N \times N)} \mathcal{I}$. Since $\mathcal{J}$ satisfies $\forall x.S(x, a) \wedge S(x, b)$ (by Proposition 7.5.2), we find $\langle \delta, \langle a, b\rangle\rangle \in S^\mathcal{J}$ for all $\delta \in \Delta^\mathcal{J}$. Thus Lemma 7.6.5 entails $\mathcal{J} \models \leqslant 0\, S.\neg F \sqsubseteq \bot$.

Moreover, $\mathcal{J}$ satisfies T. This is a consequence of Proposition 7.5.2 for all axioms of T that are in datalog. The only axioms for which this is not the case are of the form $\forall x.R(c, x) \to \bigvee_{1 \leq i \leq n} c_i \approx x$. Consider any element $\langle e, f\rangle \in \Delta^\mathcal{J}$ such that $\langle c^\mathcal{J}, \langle e, f\rangle\rangle \in R^\mathcal{J}$. By the construction of $\mathcal{J}$, we have that $\langle c^\mathcal{I}, e^\mathcal{I}\rangle, \langle c^\mathcal{I}, f^\mathcal{I}\rangle \in R^\mathcal{I}$, and thus $e^\mathcal{I} = c_i^\mathcal{I}$ and $f^\mathcal{I} = c_j^\mathcal{I}$ for some $i, j \in \{1, \ldots, n\}$. Since all constants in N must be unequal to constants c_i, this implies that $e, f \notin N$, and thus $e = f = c_i = c_j$. Therefore, $\langle e, f\rangle$ is equal to $c_i^\mathcal{J}$ for some $i \in \{1, \ldots, n\}$ whenever $\langle c^\mathcal{J}, \langle e, f\rangle\rangle \in R^\mathcal{J}$, so that the considered axiom of T is indeed satisfied.

Since $T \cup \{C\} \cup \{\leqslant k\, S.\neg F \sqsubseteq \bot\}$ is unsatisfiable, this implies $\mathcal{J} \not\models \{C\}$. This establishes the preconditions for Lemma 7.5.3 (for the case $T_1 = T_2$) and thus shows the claim.

The other cases $k = 1$ and $k \geq 2$ are very similar, using constructions $[\![a \in F, b \notin F]\!]_\times$ and $[\![c_1, \ldots, c_k \notin F]\!]_\times$ of Lemma 7.6.4 and 7.6.3. For $k = 1$, it is admissible that $a^\mathcal{I} \notin F^\mathcal{I}$ is an S-successor of all elements. For $k \geq 2$, k such S-successors $c_1^\mathcal{I}, \ldots, c_k^\mathcal{I} \notin F^\mathcal{I}$ are allowed. In either case, the product construction generates further S-successors that require $\leqslant k\, S.\neg F$ to be empty. □

Observe how the previous proof depends on using the second pre-condition of Lemma 7.5.3 where a single model is multiplied with itself. This is essential to ensure that the auxiliary theory T is satisfied in the product, even though it contains non-datalog axioms. The above result also marks a case where we really need product constructions that are different from the canonical product that uses all pairs of (named) individuals as the new interpretation domain. The auxiliary theory T in the above case would not generally be satisfied in a canonical product:

the non-datalog axioms introduced for atleast-restrictions require a fixed set of successor individuals, whereas a canonical product contains additional successors that correspond to pairs of the original individuals.

For the remaining steps of the proof, we use some additional auxiliary constructions. The datalog programs of Definitions 7.6.1 and 7.6.2 are not suitable to isolate properties that exclude a concept from DLP: to the contrary, they simply enforce certain entailments to override any complex semantic effects. The following definition therefore provides us with knowledge bases that can be used to "measure" information about the extension of a concept C without enforcing $C \sqsubseteq \bot$. The underlying intuition is that non-emptiness of some concepts can be ensured to entail *positive* information. The construction thus can be viewed as a generalisation of the construction in Lemma 7.2.6 to the more complex case of $\mathcal{SROIQ}$.

We provide two cases: $[\![c \in C \rightsquigarrow A]\!]_B$ is used to detect whether a constant c is in C, while $[\![C \rightsquigarrow A]\!]_{B\leq}$ is used to detect if C is generally non-empty. Both constructions can only work (in $\mathcal{DLP}$) if C "contains" positive information, i.e. if it is not in $\mathbf{D}_B$. Note that the constructions can be considered as specialisations of $[\![a \notin C]\!]$ and $[\![C \sqsubseteq A]\!]_{\leq}$.

Definition 7.6.7 Consider a name-separated concept C in DLP normal form such that $C \notin \mathbf{D}_B \cup \{\bot, \top\} \cup \mathbf{D}_{\geq\omega-k}$ for some $k \geq 0$. For individual names $c_0, \ldots, c_k$ and concepts $A_0, \ldots, A_k \in \mathbf{D}_H$, a datalog program $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B$ is defined recursively as follows:

- If C has the form $\mathbf{A}$, $\{\mathbf{I}\}$ or $\exists\mathbf{R}.\mathsf{Self}$, then $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B :=$ $\bigcup_{0\leq i\leq k} \mathsf{datalog}(\{c_i\} \sqcap C \sqsubseteq A_i)$.
- If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_B$, then w.l.o.g. D_1 is not a conjunction and thus $D_1 \notin \mathbf{D}_{\geq\omega-m}$ for all $m \geq 0$. Define $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B :=$ $[\![c_0, \ldots, c_k \in D_1 \rightsquigarrow A_0, \ldots, A_k]\!]_B$.
- If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}_B$, then $D_1, D_2 \notin \mathbf{D}_{\geq\omega-k}$. Set $[\![c_0, \ldots, c_k \in C \rightsquigarrow$ $A_0, \ldots, A_k]\!]_B := [\![c_0, \ldots, c_k \in D_1 \rightsquigarrow A_0, \ldots, A_k]\!]_B \cup [\![c_0, \ldots, c_k \notin D_2]\!]$.
- If $C = \geqslant n\, R.D$ with $n \geq 1$, then $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B := \{R(c_i, x) \rightarrow$ $A_i(c_i) \mid 0 \leq i \leq k\}$.
- If $C = \leqslant 0\, R.\neg D$, then, for a fresh constant d and fresh concept name B, define $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B := [\![d \in D \rightsquigarrow B]\!]_B \cup \{R(c_i, d), B(d) \rightarrow A_i(c_i) \mid$ $0 \leq i \leq k\}$.
- If $C = \leqslant n\, R.\neg D$ with $n \geq 1$, then consider fresh constants d_i ($i = 0, \ldots, n$). Define $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B := \{R(c_i, d_j) \mid 0 \leq i \leq k, 0 \leq j \leq$ $n\} \cup \{d_j \approx d_l \rightarrow A_i(c_i) \mid 0 \leq j < l \leq n, 0 \leq i \leq k\} \cup [\![d_0, \ldots, d_n \notin D]\!]$.

Moreover, if $C \notin \mathbf{D}_{\geq\omega-k}$ for all $k \geq 0$, then a datalog program $[\![C \rightsquigarrow A]\!]_{B\leq}$ is defined recursively as follows:

- If C has the form $\mathbf{A}$, $\{\mathbf{I}\}$ or $\exists\mathbf{R}.\mathsf{Self}$, then $[\![C \rightsquigarrow A]\!]_{B\leq} := \mathsf{datalog}(C \sqsubseteq A)$.
- If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_B$ and $D_1 \notin \mathbf{D}_{\geq\omega-n}$ for all $n \geq 0$, then $[\![C \rightsquigarrow A]\!]_{B\leq} := [\![D_1 \rightsquigarrow A]\!]_{B\leq}$.
- If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}_B$, then $[\![C \rightsquigarrow A]\!]_{B\leq} := [\![D_1 \rightsquigarrow A]\!]_{B\leq} \cup [\![D_2 \sqsubseteq A]\!]_{\leq}$.
- If $C = \geqslant n\, R.D$ with $n \geq 1$, then $[\![C \rightsquigarrow A]\!]_{B\leq} := \{R(x,y) \to A(x)\}$.
- If $C = \leqslant 0\, R.\neg D$, then, for a fresh constant c and fresh concept name B, define $[\![C \rightsquigarrow A]\!]_{B\leq} := [\![c \in D \rightsquigarrow B]\!]_B \cup \{R(x,c), B(c) \to A(x)\}$.
- If $C = \leqslant n\, R.\neg D$ with $n \geq 1$, then consider fresh constants c_i $(i = 0, \ldots, n)$. Define $[\![C \rightsquigarrow A]\!]_{B\leq} := \{R(x,c_i) \mid 0 \leq i \leq n\} \cup \{c_i \approx c_j \to A(x) \mid 0 \leq i < j \leq n\} \cup [\![c_0, \ldots, c_n \notin D]\!]$. $\diamond$

It should be noted that the cases in the previous definition are indeed exhaustive: side conditions usually are only provided to specify a particular situation that can be assumed without loss of generality. Conditions that follow from the assumptions are omitted. Observe that the necessary conditions for recursion are satisfied in all cases of the definition. The choice of D_1 in the cases for $C = D_1 \sqcap D_2$ is possible since we disregard the nesting order of $\sqcup$: if there is some $D_1 \notin \mathbf{D}_B$, then there is some such D_1 that does not have a $\mathbf{C}_\geq$ disjunct (which is in $\mathbf{D}_B$) while still $D_1 \notin \mathbf{D}_B$. But then this $D_1 \notin \mathbf{D}_{\geq\omega-m}$ for all $m \geq 0$ as required.

It is not hard to see that, given the preconditions of Definition 7.6.7, we find that $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B \models \bigcup_{0\leq i\leq k} \{C \sqcap \{c_i\} \sqsubseteq A_i\}$ and $[\![C \rightsquigarrow A]\!]_{B\leq} \models C \sqsubseteq A$. Notably, the case $C = \leqslant n\, R.\neg D$ uses a different approach than the other cases: the positive information used to entail non-emptiness of A is found in the equality relations that are implied between auxiliary constants d_i.

Observe that the datalog programs of Definition 7.6.7 again may significantly constrain the extension of C. For example, if $C = \leqslant 1\, R.\neg\bot$ then $[\![C \rightsquigarrow A]\!]_{B\leq}$ is only satisfied by interpretations that entail either $C \sqsubseteq \bot$ or $\top \sqsubseteq C$. This may entail $\top \sqsubseteq A$, so we will only use $[\![C \rightsquigarrow A]\!]_{B\leq}$ if $\top \sqsubseteq A$ or $C \sqsubseteq \bot$ is satisfiable. Non-emptiness of C might also be unavoidable, so one cannot assume that $[\![C \rightsquigarrow A]\!]_{B\leq} \cup \{A \sqsubseteq \bot\}$ is satisfiable. Yet, the remaining freedom will generally suffice for our purposes.

Another noteworthy fact is that $[\![c_0, c_1 \in C \rightsquigarrow A_0, A_1]\!]_B$ is not the same as $[\![c_0 \in C \rightsquigarrow A_0]\!]_B \cup [\![c_1 \in C \rightsquigarrow A_1]\!]_B$, which is the reason why the definition must explicitly include cases with $k > 0$. To see this, consider $C = (\neg\{a\} \sqcap \neg\{b\}) \sqcup B$. Then $[\![c_0, c_1 \in C \rightsquigarrow A_0, A_1]\!]_B = \{B(c_0) \to A_0(c_0), B(c_1) \to A_0(c_1), c_0 \approx a, c_1 \approx b\}$ but $[\![c_0 \in C \rightsquigarrow A_0]\!]_B \cup [\![c_1 \in C \rightsquigarrow A_1]\!]_B = \{B(c_0) \to A_0(c_0), B(c_1) \to A_0(c_1), c_0 \approx a, c_1 \approx a\}$. The latter entails the unwanted consequence $c_0 \approx c_1$ since the auxiliary programs $[\![c_i \notin \neg\{a\} \sqcap \neg\{b\}]\!]$ are constructed independently for $i = 0, 1$ instead of using $[\![c_0, c_1 \notin \neg\{a\} \sqcap \neg\{b\}]\!]$.

The following lemma provides some important ingredients for showing maximality of $\mathcal{DLP}$, since it establishes the pre-conditions of Lemma 7.5.3 for broad classes of concepts.

Lemma 7.6.8 *Let $C \in \mathbf{D}_a^+$ be a name-separated concept expression in DLP normal form, let $\mathbf{I}$ be the set of constants of the given signature, and let $a, b, c \in \mathbf{I}$ be arbitrary constants not occurring in C.*

(1) If $C \notin \mathbf{D}_H$, then one of the following is true:

- *There is a theory T and a set of constants $N \subseteq \mathbf{I}$ with $a, b \in N$ such that: given an arbitrary model $\mathcal{I}$ of $\{\{a\} \sqcup \{b\} \sqsubseteq C\} \cup T$, we find that $\mathcal{J} = \mathcal{I} \times_{(N\times N)} \mathcal{I}$ is such that $\langle a, b\rangle \notin C^{\mathcal{J}}$.*
- *There are theories T_1, T_2 such that: given arbitrary models $\mathcal{I}_i$ of $\{\{a\} \sqcup \{b\} \sqsubseteq C\} \cup T_i$ $(i = 1, 2)$, we find that $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I}\times\mathbf{I})} \mathcal{I}_2$ is such that $\langle a, b\rangle \notin C^{\mathcal{J}}$.*

(2) If $C \notin \mathbf{D}_a$, then there are theories T_1, T_2 such that: given arbitrary models $\mathcal{I}_i$ of $\{\{c\} \sqsubseteq C\} \cup T_i$ $(i = 1, 2)$, we find that $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I}\times\mathbf{I})} \mathcal{I}_2$ is such that $c^{\mathcal{J}} = \langle c, c\rangle \notin C^{\mathcal{J}}$.

In all cases, models $\mathcal{I}$, $\mathcal{I}_1$ and $\mathcal{I}_2$ as described in the claims exist.

Proof. By Proposition 7.6.6, $C \in \mathbf{D}_a^+$ implies $D \in \mathbf{D}_a^+$ for all subconcepts D of C.

We start by considering claim (1). Claim (2) is shown independently below, so if $C \notin \mathbf{D}_a$ then we obtain theories T_1 and T_2 as in claim (2) for some fresh constant c. It is easy to see that the theories $T_i' := T_i \cup \{a \approx c, b \approx c\}$ $(i = 1, 2)$ suffice for establishing claim (1). It remains to show claim (1) for cases where $C \in \mathbf{D}_a$. An easy induction can be used to show that $\mathbf{D}_H^+ \cap \mathbf{D}_a \subseteq \mathbf{D}_H$. Hence, using our assumption that $C \notin \mathbf{D}_H$, we can also conclude $C \notin \mathbf{D}_H^+$.

The only remaining cases for claim (1) therefore are such that $C \notin \mathbf{D}_H^+$, so that Lemma 7.6.5 can be applied. Define $T := [\![a, b \in C]\!]_\times$, and define N as in the lemma. The claim follows from Lemma 7.6.5.

For claim (2), we construct theories $T_1 = T_1(c, C)$ and $T_2 = T_2(c, C)$ for a fresh constant c as in the claim. The proof proceeds by induction over the structure of C. Note that C cannot be an atomic class, nominal, Self restriction, or the negation thereof.

Consider the case $C = D_1 \sqcap D_2$. Without loss of generality, we find that $D_1 \notin \mathbf{D}_a$. Applying the induction hypothesis, we obtain theories $T_i(c, C) := T_i(c, D_1)$ $(i = 1, 2)$ that satisfy the claim.

Consider the case $C = D_1 \sqcup D_2$. As a first case, assume that $D_1 \notin \mathbf{D}_a$. Then we can define theories $T_i(c, C) \coloneqq T_i(c, D_1) \cup [\![c \notin D_2]\!]$ $(i = 1, 2)$. The claim then follows from the induction hypothesis together with the fact that every product interpretation constructed from models of $T_i(c, C)$ $(i = 1, 2)$ must also satisfy $[\![c \notin D_2]\!]$ by Proposition 7.5.2. The case $D_2 \notin \mathbf{D}_a$ is similar.

Now assume that $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}_B$. Using fresh concept names A_1, A_2 and the construction of Definition 7.6.7, define $T_i(c, C) \coloneqq \{A_i(c) \to \bot\} \cup \bigcup_{j=1,2}[\![c \in D_j \rightsquigarrow A_j]\!]_B$ for $i = 1, 2$. Then any product interpretation $\mathcal{J}$ of any two models of $T_i(c, C)$ $(i = 1, 2)$ satisfies $\bigcup_{j=1,2}[\![c \in D_j \rightsquigarrow A_j]\!]_B \cup \{A_j(c) \to \bot\}$, and hence $\mathcal{J} \not\models \{c\} \sqcup D_i$ $(i = 1, 2)$ as required.

Consider the case $C = \leqslant 0\, R.\neg D$ with $D \notin \mathbf{D}_H$. Since $C \in \mathbf{D}_a^+$ we find $D \in \mathbf{D}_H^+$. Using $\mathbf{D}_H^+ \cap \mathbf{D}_a \subseteq \mathbf{D}_H$ as above, we conclude that $D \notin \mathbf{D}_a$, which allows us to apply the induction hypothesis. Consider a fresh individual name d and define $T_i(c, C) \coloneqq T_i(d, D) \cup \{R(c, d)\}$ $(i = 1, 2)$. Given models $\mathcal{I}_i$ of $T_i(c, C)$ $(i = 1, 2)$, the induction hypothesis implies that $\mathcal{J} \coloneqq \mathcal{I}_1 \times_{(\mathbf{I} \times \mathbf{I})} \mathcal{I}_2$ does not satisfy $\{d\} \sqsubseteq D$. Since $\mathcal{J} \models R(c, d)$ we conclude $\mathcal{J} \not\models \{c\} \sqsubseteq C$.

Consider the case $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_B$ and $D \notin \mathbf{D}_{\geq\omega-1}$. Using fresh symbols c_1, c_2, A_1, A_2, we define $T_i(c, C) \coloneqq \{A_i(c_i) \to \bot\} \cup [\![c_1, c_2 \in D \rightsquigarrow A_1, A_2]\!]_B \cup \{R(c, c_1), R(c, c_2)\}$ for $i = 1, 2$. Using similar arguments as in the last case of $C = D_1 \sqcup D_2$, we find that no product interpretation of models of $T_i(c, C)$ $(i = 1, 2)$ can satisfy $\{c\} \sqsubseteq C$.

Consider the case $C = \leqslant n\, R.\neg D$ with $n \geq 2$ and $D \notin \mathbf{D}_{\geq\omega-n}$. Using fresh individuals symbols $c_0, \ldots, c_n$, set $T \coloneqq [\![c_0, \ldots, c_n \notin D]\!] \cup \{R(c, c_i) \mid 0 \leq i \leq n\}$. We define $T_1(c, C) \coloneqq T \cup \{c_i \approx c_j \to \bot \mid 1 \leq i < j \leq n\}$ and $T_2(c, C) \coloneqq T \cup \{c_i \approx c_j \to \bot \mid 0 \leq i < j \leq n - 1\}$. Thus, any model of $\{\{c\} \sqsubseteq C\} \cup T_1(c, C)$ $(\{\{c\} \sqsubseteq C\} \cup T_2(c, C))$ entails $c_0 \approx c_1$ $(c_{n-1} \approx c_n)$, but this entailment is lost in every product interpretation. This shows the desired result since product interpretations satisfy T by Proposition 7.5.2.

Consider the case $C = \geqslant 1\, R.D$ with $D \notin \mathbf{D}^{\geqslant 1}$. Then $D \in \mathbf{D}_a^+$ and $D \notin \mathbf{D}_a$. For a fresh constant d, define $T_i(c, C) \coloneqq T_i(d, D) \cup \{R(c, x) \to d \approx x\}$ for $i = 1, 2$. The claim follows from the induction hypothesis and the fact that every considered product interpretation also satisfies $\{R(c, x) \to d \approx x\}$.

Consider the case $C = \geqslant n\, R.D$ with $n \geq 2$ and $D \notin \mathbf{D}^{\geqslant n}$. Without loss of generality, we can assume that D is of the form $C_1 \sqcup \ldots \sqcup C_p \sqcup E$ $(p \geq 1)$ where no C_i is a disjunction, $C_i \notin \mathbf{C}_B$ for $i = 1, \ldots, p$, and $E \in \mathbf{D}_B \cup \{\bot\}$. For the following argument, we use $E = \bot$ to cover the case where no such E is given in the original DLP normal form. Note that E might be a disjunction but cannot be $\top$.

First assume that there is some $F \in \{E, C_1, \ldots, C_p\}$ such that $F \in \mathbf{D}_{\geq\omega-k}$ for some $k \geq 0$. Since F is in DLP normal form, it is a disjunction that contains some disjunct in $\mathbf{C}_{\neg m}$ $(m \geq 1)$. All subconcepts of D are assumed to be in $\mathbf{D}_a^+$, so if $m \leq n^2 - n$ then $D \in \mathbf{D}^{\geqslant n}$; a contradiction. Thus D is of the form $D_1 \sqcup D_2$ with

$D_1 \in \mathbf{C}_{\neg m}$ and $m > n^2 - n$. Moreover, $D_2 \notin \mathbf{D}_a$ since otherwise we would find $D \in \mathbf{D}_a \subset \mathbf{D}^{\geqslant n}$.

Let $\mathsf{ind}(D_1) = \{c_1, \ldots, c_m\}$ be the set of constants in D_1. Let $p_1, p_2, \ldots, p_{n^2-n}$ denote a sequence of all pairs $p_i = \langle d_1, d_2\rangle$ of constants $d_1, d_2 \in \{c_1, \ldots, c_n\}$ with $d_1 \neq d_2$. The order is inessential, but some order is needed for notational purposes. Define auxiliary theories $T_i(c, C) := \left\{\forall x.R(c, x) \to \bigvee_{1\leq j\leq n} c_j \approx x\right\} \cup \bigcup_{1\leq j\leq m} T_i(c_j, D_2) \cup \{c_j \approx d_i \mid n < j \leq m, p_{j-n} = \langle d_1, d_2\rangle\}$. Observe that the first component in this definition refers only to the first n constants $c_1, \ldots, c_n$, the second part is specified for all m constants, and the third component refers to the last $m - n$ constants $c_{n+1}, \ldots, c_m$ only.

To see that these theories satisfy the claim, consider models $\mathcal{I}_i$ of $\{\{c\} \sqsubseteq C\} \cup T_i(c, C)$ $(i = 1, 2)$, and let $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I}\times\mathbf{I})} \mathcal{I}_2$ denote their product. Observe that, by the construction of $T_i(c, C)$, the constants c_j $(1 \leq j \leq m)$ are mutually unequal in $\mathcal{J}$. Now consider an arbitrary element $\delta \in \Delta^{\mathcal{J}}$ such that $\langle c^{\mathcal{J}}, \delta\rangle \in R^{\mathcal{J}}$. By definition of the product, there must be a constant symbol d – possibly an auxiliary constant that did not occur in C – such that $\delta = \langle d, d\rangle$ and $\langle c^{\mathcal{I}_i}, d^{\mathcal{I}_i}\rangle \in R^{\mathcal{I}_i}$ for $i = 1, 2$. Since the models $\mathcal{I}_i$ satisfy $\forall x.R(c, x) \to \bigvee_{1\leq j\leq n} c_j \approx x$, we conclude that $\mathcal{I}_1 \models d \approx c_j$ and $\mathcal{I}_2 \models d \approx c_k$ for some (possibly distinct!) $j, k \in \{1, \ldots, n\}$. Thus, there are at most n^2 elements $\delta \in \Delta^{\mathcal{J}}$ such that $\langle c^{\mathcal{J}}, \delta\rangle \in R^{\mathcal{J}}$, since there are at most n^2 distinct ways of selecting j, k. Now m of those n^2 elements are of the for $c_j^{\mathcal{J}}$ for some $j = 1, \ldots, m$, and by the induction hypothesis we find that $c_j^{\mathcal{J}} \notin D_2^{\mathcal{J}}$. Since $c_j^{\mathcal{J}} \notin D_1^{\mathcal{J}}$ is immediate, we thus find that $c_j^{\mathcal{J}} \notin D^{\mathcal{J}}$ for all $j = 1, \ldots, m$. Summing up, we conclude that $\mathcal{J}$ can have most $n^2 - m$ distinct R-successors for c which are in D. Since $n^2 - m < n^2 - (n^2 - n) = n$, we find that $\mathcal{J} \not\models \{c\} \sqsubseteq \geqslant n\, R.D$, as required.

For the rest of the proof, assume that $F \notin \mathbf{D}_{\geq\omega-k}$ for all $F \in \{E, C_1, \ldots, C_p\}$ and $k \geq 0$. In particular, we can use the constructions of Definition 7.6.2 and 7.6.7. Now if $\{\{c\} \sqsubseteq C\} \cup [\![E \sqsubseteq \leqslant 0\, R^{-}.\neg\neg\{c\}]\!]_{\leq}$ is unsatisfiable, then $C_1 \sqcup \ldots \sqcup C_p \in \mathbf{D}_{\leq n-1}$. Since we assumed that $C_1 \sqcup \ldots \sqcup C_p \in \mathbf{D}_a^{+}$, this again implies $D \in \mathbf{D}^{\geqslant n}$. Hence, $\{\{c\} \sqsubseteq C\} \cup [\![E \sqsubseteq \leqslant 0\, R^{-}.\neg\neg\{c\}]\!]_{\leq}$ must be satisfiable (note that this includes the case $E = \bot$). It is easy to see that $\{\{c\} \sqsubseteq C\} \cup [\![E \sqsubseteq \leqslant 0\, R^{-}.\neg\neg\{c\}]\!]_{\leq}$ semantically emulates $\{\{c\} \sqsubseteq \geqslant n\, R.C_1 \sqcup \ldots \sqcup C_p\}$, and that the claim can thus be established by induction. So for the remaining considerations we can assume that E is not present at all, i.e. that $C = \geqslant n\, R.C_1 \sqcup \ldots \sqcup C_p$.

Using the assumptions on C_i, we can apply Definition 7.6.7 and define $T := \bigcup_{1\leq i\leq p} ([\![C_i \rightsquigarrow A_i]\!]_{B\leq} \cup \{R(x, y) \wedge A_i(y) \to B_i(x)\})$ for fresh concept names $A_1, \ldots,$ A_p, $B_1, \ldots, B_p$. It is easy to verify that $\{\{c\} \sqsubseteq C\} \cup T$ is consistent. Now consider the theory $T' := T \cup \{B_i(x) \to \bot \mid T \cup \{\{c\} \sqsubseteq C\} \cup \{B_i \sqsubseteq \bot\}$ is consistent$\}$, where it should be noted how the B_i are used to avoid inconsistencies that could arise immediately when requiring $A_i \sqsubseteq \bot$. Consider the case (A) that $T' \cup \{\{c\} \sqsubseteq C\}$

is inconsistent. Then there are two disjoint subsets $I_1, I_2 \subseteq \{1, \ldots, p\}$ for which $T_k(c, C) \coloneqq T \cup \{B_i \sqsubseteq \bot \mid i \in I_k\}$ is such that $T_k(c, C) \cup \{\{c\} \sqsubseteq C\}$ is consistent for $k = 1, 2$, while $T_1(c, C) \cup T_2(c, C) \cup \{\{c\} \sqsubseteq C\}$ is inconsistent. Every product interpretation of models of $T_k(c, C)$ ($k = 1, 2$) entails T (by Proposition 7.5.2) and $B_i \sqsubseteq \bot$ (by Definition 7.5.1), and thus cannot be a model of $\{\{c\} \sqsubseteq C\}$, as required.

Now consider the case (B) where $T' \cup \{\{c\} \sqsubseteq C\}$ is consistent. Then there is B_h such that $T \cup \{\{c\} \sqsubseteq C\} \cup \{B_h \sqsubseteq \bot\}$ is inconsistent. This implies that $\{\{c\} \sqsubseteq C\} \cup \{\geqslant 1\, R.C_h \sqsubseteq \bot\}$ is inconsistent. Since $C_h \notin \mathbf{C}_{\geq} \subseteq \mathbf{D}_B$, we conclude that either $\bigsqcup_{1 \leq i \leq p, i \neq h} C_i \in \mathbf{D}_{\leq n-1}$ or this concept is empty, i.e. $p = h = 1$.

First consider the case (B.1) where $C_h \in \mathbf{D}_{\leq 1}$. Then $C_1 \sqcup \ldots \sqcup C_p \notin \mathbf{D}_{\leq n-1}$ implies $p = n$ and $C_i \in \mathbf{D}_{\leq 1}$ for all $i \neq h$, $1 \leq i \leq p$. Since C is not of the $\mathbf{D}_H$-form $\geqslant n\, R.\mathbf{D}_{n!}$, there is k such that $C_k \notin \mathbf{D}_a$. Now $C_k \in \mathbf{D}_{\leq 1}$ implies that $C_k = \{a\} \sqcap C'_k$ for some individual a and concept $C'_k \notin \mathbf{D}_a$. As each model of C requires one R-successor of c in each concept of the form C_i, we find that $\{\{c\} \sqsubseteq C\}$ semantically emulates $\{\{a\} \sqsubseteq C_k\}$. The claim follows by induction.

As a second case (B.2), assume that $C_h \notin \mathbf{D}_{\leq 1}$. Then $C_h \notin \mathbf{D}_{\leq k}$ for all $k \geq 0$ since C_h is not a disjunction. Since this implies that $T \cup \{\{c\} \sqsubseteq C\} \cup \{B_i \sqsubseteq \bot \mid i \neq h\}$ is consistent, this theory must be equal to $T' \cup \{\{c\} \sqsubseteq C\}$.

Consider the case (B.2.1) where $C_h \notin \mathbf{D}_a$. For fresh individuals $c_1, \ldots, c_n$ define $T'' \coloneqq T' \cup \{\forall R(c, x) \to \bigvee_{1 \leq i \leq n} c_i \approx x\}$. Note that $T'' \cup \{\{c\} \sqsubseteq C\}$ is satisfiable by interpretations $\mathcal{I}$ that have $c_i^{\mathcal{I}} \in C_h^{\mathcal{I}}$ as the n distinct R-successors of c. Define $T_i(c, C) \coloneqq \bigcup_{1 \leq j \leq n} T_i(c_j, C_h) \cup T''$ ($i = 1, 2$).

To show that this satisfies the claim, consider models $\mathcal{I}_i$ of $\{\{c\} \sqsubseteq C\} \cup T_i(c, C)$ ($i = 1, 2$). Since the induction hypothesis only applies to named individuals, we introduce n^2 fresh constants $\langle c_j, c_k\rangle$ for $j, k \in \{1, \ldots, n\}$. $\mathcal{I}_1$ is extended to $\mathcal{I}'_1$ over this extended signature by setting $\langle c_j, c_k\rangle^{\mathcal{I}'_1} \coloneqq c_j^{\mathcal{I}_1}$, so that $\mathcal{I}'_1 \models \langle c_j, c_k\rangle \approx c_j$. The extended interpretation $\mathcal{I}'_2$ is defined analogously for the second components. Due to the constructions in this proof, for any constants e, f, we find that $T_i(e, C_h)$ is the same as $T_i(f, C_h)$ with e uniformly replaced by f ($i = 1, 2$). Thus, we find that $\mathcal{I}'_i \models T_i(\langle c_j, c_k\rangle, C_h)$ for $i = 1, 2$ and all $j, k \in \{1, \ldots, n\}$. Moreover, $\mathcal{I}'_i \models \{\{\langle c_j, c_k\rangle\} \sqsubseteq C_h\}$ so the induction hypothesis can be applied to obtain $\mathcal{I}'_1 \times_{(\mathbf{I}' \times \mathbf{I}')} \mathcal{I}'_2 \not\models \{\langle c_j, c_k\rangle\} \sqsubseteq C_h$ where $\mathbf{I}'$ denotes the extended set of constants.

It is not hard to see that the interpretations $\mathcal{J}' = \mathcal{I}'_1 \times_{(\mathbf{I}' \times \mathbf{I}')} \mathcal{I}'_2$ and $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I} \times \mathbf{I})} \mathcal{I}_2$ are equal (possibly up to renaming of domain elements). In particular, $\mathcal{J}'$ entails $\langle c_j, c_k\rangle \approx \langle\langle c_j, c_{j'}\rangle, \langle c_{k'}, c_k\rangle\rangle$. Hence we find that $\mathcal{J} \not\models \{\langle c_j, c_k\rangle\} \sqsubseteq C_h$. Moreover, since $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy T'', we find that $\langle c^{\mathcal{J}}, \delta\rangle \in R^{\mathcal{J}}$ implies $\delta = \langle c_j, c_k\rangle^{\mathcal{J}}$ for some $j, k \in \{1, \ldots, n\}$. Thus we obtain $\mathcal{J} \not\models \{\{c\} \sqsubseteq C\}$ as required.

As the final case (B.2.2), assume that $C_h \in \mathbf{D}_a$. Since $D \notin \mathbf{D}^{\geqslant n}$, we find $D \neq C_h$, i.e. $p > 1$. We concluded $\bigsqcup_{1 \leq i \leq p, i \neq h} C_i \in \mathbf{D}_{\leq n-1}$ above for all sub-cases of (B). Hence D is of the form $\mathbf{D}_a \sqcup \mathbf{D}^{+}_{m!} \sqcup \mathbf{D}_{l!}$ – where we assume that m is the least natural number for which D has this form – and m and l do not satisfy the relevant conditions in the definition of $\mathbf{D}^{\geqslant n}$. Accordingly, we denote D as $C_h \sqcup M_1 \sqcup \ldots \sqcup M_m \sqcup L_1 \sqcup \ldots \sqcup L_l$. Since $M_1, \ldots, M_m, L_1, \ldots, L_l \in \mathbf{D}_{\leq 1}$, they are each of the form $\{d\} \sqcap C$ for some individual name d: let $e_1, \ldots, e_m, f_1, \ldots, f_l$ denote these individual names. Set $r := n - (m + l)$, and consider fresh individual names $c_1, \ldots, c_r$. Define a set $X := \{c_1, \ldots, c_r, e_1, \ldots, e_m, f_1, \ldots, f_l\}$ of all constants considered as R-successors of c. Using the induction hypothesis, define

$$\begin{aligned} T_i(c, C) \; := \; & [\![e_1, \ldots, e_m, f_1, \ldots, f_l \notin C_h]\!]_\times \cup \\ & [\![c_1, \ldots, c_r \in C_h, e_1, \ldots, e_m, f_1, \ldots, f_l \notin C_h]\!]_\times \cup \\ & \textstyle\bigcup_{1 \leq j \leq m} T_i(e_j, M_j) \cup \{\forall x. R(c, x) \to \bigvee_{d \in X} d \approx x\} \end{aligned}$$

for $i = 1, 2$. Note that the construction of Lemma 7.6.3 is possible: if C_h would be in $\mathbf{D}^+_B$, then $C \in \mathbf{D}_a$ would imply $C \in \mathbf{D}_B$, which cannot be.

To show that this satisfies the claim, consider models $\mathcal{I}_i$ of $\{\{c\} \sqsubseteq C\} \cup T_i(c, C)$ ($i = 1, 2$), and let $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I} \times \mathbf{I})} \mathcal{I}_2$ be the corresponding product interpretation. By the constructions of $T_i(c, C)$, we obtain that $\langle c^\mathcal{J}, \delta \rangle \in R^\mathcal{J}$ implies $\delta = \langle a, b \rangle^\mathcal{J}$ for some $a, b \in X$. We distinguish various cases:

- If $a, b \in \{e_1, \ldots, e_m, f_1, \ldots, f_l\}$ and $a \neq b$, then $\langle a, b \rangle^\mathcal{J} \notin E^\mathcal{J}$ for all $E = M_1, \ldots, M_m, L_1, \ldots, L_l$ can be concluded from $\langle a, b \rangle^\mathcal{J} \neq d^\mathcal{J}$ for all $d = e_1, \ldots, e_m, f_1, \ldots, f_l$. Moreover, $\langle a, b \rangle^\mathcal{J} \notin C_h$ by Lemma 7.6.3.
- If $a = b = e_j$ for some $j = 1, \ldots, m$, then $\langle a, b \rangle^\mathcal{J} \notin C_h$ again by Lemma 7.6.3. As above, $\langle a, b \rangle^\mathcal{J} \notin L_i^\mathcal{J}$ for all $i = 1, \ldots, l$. A similar argument shows $\langle a, b \rangle^\mathcal{J} \notin M_i^\mathcal{J}$ for all $i = 1, \ldots, m$ with $i \neq j$, whereas $\langle a, b \rangle^\mathcal{J} \notin M_j^\mathcal{J}$ follows by the induction hypothesis.
- If $a \in \{e_1, \ldots, e_m, f_1, \ldots, f_l\}$ and $b \in \{c_1, \ldots, c_r\}$, then $\langle a, b \rangle^\mathcal{J} \notin C_h$ follows from Lemma 7.6.4. The conclusion $\langle a, b \rangle^\mathcal{J} \notin E^\mathcal{J}$ for all $E = M_1, \ldots, M_m, L_1, \ldots, L_l$ follows as before.

In each of these cases, we thus find that $\langle a, b \rangle^\mathcal{J} \notin D^\mathcal{J}$. Therefore, the only elements $\langle a, b \rangle^\mathcal{J}$ that might be in $D^\mathcal{J}$ are such that either $a = b \in \{f_i, \ldots, f_l\}$ or $a, b \in \{c_1, \ldots, c_r\}$. This yields a maximum of $l + r^2$ R-successors for $c^\mathcal{J}$. Since $D \notin \mathbf{D}^{\geqslant n}$, we find that $r(r - 1) < m$ (the case $r \leq 0$ cannot occur for any case under (B)). Equivalently, $r^2 - r < m$ which in turn is equivalent to $r^2 - n + m + l < m$. But then $r^2 + l < n$, and we find $\mathcal{J} \not\models \{c\} \sqsubseteq C$, as required. □

The previous lemma already suffices to exclude a significant amount of axioms from DLP:

Corollary 7.6.9 *Let C be a name-separated concept expression in DLP normal form, let A be a fresh concept name, and let c be a fresh constant symbol.*

(1) If $C \notin \mathbf{D}_H \cup \{\top, \bot\}$, then $A \sqsubseteq C$ cannot be $\mathbf{FOL}_{\approx}$-emulated by any datalog program.

(2) If $C \notin \mathbf{D}_a \cup \{\top, \bot\}$, then $\{c\} \sqsubseteq C$ cannot be $\mathbf{FOL}_{\approx}$-emulated by any datalog program.

(3) If $C \notin \mathbf{D}_H \cup \{\top, \bot\}$, and $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$, and $C \notin \mathbf{C}_{\neq\top}$, then C cannot be $\mathbf{FOL}_{\approx}$-emulated by any datalog program.

Proof. If $C \notin \mathbf{D}_a^+$, then the result follows from Proposition 7.6.6 in all cases. Thus assume that $C \in \mathbf{D}_a^+$ for the remainder of the proof.

For claim (1), consider fresh individual symbols a and b, and construct T_1 and T_2 as in Lemma 7.6.8 (1). Define $T_i' := T_i \cup \{A(a), A(b)\}$ for $i = 1, 2$. Then T_1 and T_2 satisfy the preconditions of Lemma 7.5.3 for the knowledge base $\mathrm{KB} = \{\{a\} \sqsubseteq A, \{b\} \sqsubseteq A, A \sqsubseteq C\}$. In particular, $T_i \cup \{A \sqsubseteq C\}$ is satisfiable since C is in DLP normal form and $C \neq \bot$. This suffices to establish the claim.

For claim (2) and (3), we can use the theories T_1 and T_2 of Lemma 7.6.8 (2) and (1), respectively. To ensure that the preconditions of Lemma 7.5.3 hold for claim (3), we need to ensure that $\{C\} \cup T_i$ is satisfiable for $i = 1, 2$. To this end, $C \notin \mathbf{C}_{\neq\top} \cup \{\bot\}$ ensures that $\{C\}$ is satisfiable. $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$ ensures that C is satisfiable by interpretations of arbitrary domain sizes, and it is not hard to see that $\{C\} \cup T_i$ is consistent when considering the construction in Lemma 7.6.8. □

The previous result already covers a significant amount of concept expressions that are not in $\{\top, \bot\} \cup \mathbf{D}_H \cup \mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top}$. It remains to show that concepts in $\mathbf{D}_{\leq n} \setminus (\mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top})$ for some $n \geq 1$ cannot belong to DLP.

Lemma 7.6.10 *Let C be a name-separated concept expression in DLP normal form such that $C \notin \{\top, \bot\}$, and $C \in \mathbf{D}_{\leq n} \setminus (\mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top} \cup \mathbf{D}_H)$ for some $n \geq 1$. Then C cannot be $\mathbf{FOL}_{\approx}$-emulated by any datalog program.*

Proof. Observe that, for any $m \geq 1$, we find $\mathbf{C}_H^p \subset \mathbf{D}_H^p \subset \mathbf{C}_\bot^{=m} \subset \mathbf{C}_\bot^{=m+1}$. We define the *degree* $d(D)$ of a concept expression D as follows. If $D \in \mathbf{C}_\bot^{=m}$ for some $m \geq 1$, then let $d(D)$ be the largest such m. Otherwise, if $D \in \mathbf{D}_H^p$, then define $d(D) := 1$. Otherwise set $d(D) := 0$. Now since $C \in \mathbf{D}^{=n}$ it is of the form $C = (\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_n\} \sqcap C_n)$, and we can assume that $d(C_i) \leq d(C_{i+1})$ for all $i = 1, \ldots, n-1$. Using this notation, it is not hard to see that $C \notin \mathbf{D}^{=n}$ is equivalent to saying that $d(C_i) < i$ for some $i = 1, \ldots, n$.

First consider the case that $i > 1$. We find that C is semantically equivalent to $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_i\} \sqcap C_i)$. To see this, assume that $n \geq i$. Every model of C has at most n elements in its domain. Since $d(C_n) \geq n$ by construction, $C_n \in \mathbf{C}_\bot^{=n}$. By Lemma 7.3.6, we thus obtain $C_n \sqsubseteq C$ as a consequence of C, showing that C is equivalent to $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_{n-1}\} \sqcap C_{n-1})$. The claim thus follows by induction.

Now $C_j \notin \mathbf{C}_\bot^{=i}$ holds for all $j \leq i$. Using Lemma 7.3.6, we thus find that $\{c_j\} \sqsubseteq C_j$ is satisfiable by models of at most i elements in their domain. By name separation of C, we find that C is satisfiable, and clearly C is only satisfied by models with exactly $i > 1$ domain elements. Finite domain sizes can be enforced by $\mathbf{FOL}_\approx$ theories, and hence must be preserved by $\mathbf{FOL}_\approx$-emulation. But domain sizes greater than 1 are not preserved by the product construction of Definition 7.5.1, so the fact that C cannot be $\mathbf{FOL}_\approx$-emulated in datalog is a consequence of Proposition 7.5.2.

Consider the case $i = 1$. Using the same argument as above, we find that C is semantically equivalent to $\{c_1\} \sqcap C_1$. By construction, $C_1 \notin \mathbf{D}_H^p$. The claim is now shown by a miniature version of the proof steps that were used to establish Corollary 7.6.9, where relevant constructions and arguments largely collapse due to the requirement that the domain of interpretation is unary. We first provide two auxiliary constructions for the "propositional" variants of Definition 7.6.1 and 7.6.7. Given a name-separated concept $D \notin \mathbf{D}_\top^p$ and a constant d, recursively construct a datalog program $[\![d \notin D]\!]^p$ as follows:

- If $D \in \mathbf{C}_\bot^{=1}$ then $[\![d \notin D]\!]^p := \emptyset$.
- If D is of the form $\mathbf{A}$, $\neg\mathbf{A}$, $\neg\{\mathbf{I}\}$, $\exists\mathbf{R}.\mathsf{Self}$, or $\neg\exists\mathbf{R}.\mathsf{Self}$, then $[\![d \notin D]\!]^p := \mathsf{datalog}(\{d\} \sqsubseteq \neg D)$.
- If $D = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_\top^p$, then $[\![d \notin D]\!]^p := [\![d \notin D_1]\!]^p$.
- If $D = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}_\top^p$, then $[\![d \notin D]\!]^p := [\![d \notin D_1]\!]^p \cup [\![d \notin D_2]\!]^p$.
- If $D = \leqslant 0\, R.\neg D'$ with $D' \notin \mathbf{D}_\top^p$, then $[\![d \notin D]\!]^p := \{R(d,d)\} \cup [\![d \notin D']\!]^p$.
- If $D = \geqslant n\, R.D'$ with $n > 0$, then $[\![d \notin D]\!]^p := \{\neg R(d,d)\}$.

If $D \notin \mathbf{D}_B^p$ then, for a concept name A, we recursively construct a datalog program $[\![\{d\} \sqcap D \sqsubseteq A]\!]_B^p$ as follows:

- If D has the form $\mathbf{A}$ or $\exists\mathbf{R}.\mathsf{Self}$, then $[\![\{d\} \sqcap D \sqsubseteq A]\!]_B^p := \mathsf{datalog}(\{d\} \sqcap D \sqsubseteq A)$.
- If $D = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_B^p$, then $[\![\{d\} \sqcap D \sqsubseteq A]\!]_B^p := [\![\{d\} \sqcap D_1 \sqsubseteq A]\!]_B^p$.
- If $D = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}_B^p$ and $D_2 \notin \mathbf{D}_\top^p$, then $[\![\{d\} \sqcap D \sqsubseteq A]\!]_B^p := [\![\{d\} \sqcap D_1 \sqsubseteq A]\!]_B^p \cup [\![d \notin D_2]\!]^p$.
- If $D = \leqslant 0\, R.\neg D'$ with $D' \notin \mathbf{D}_B^p$, then $[\![\{d\} \sqcap D \sqsubseteq A]\!]_B^p := [\![\{d\} \sqcap D' \sqsubseteq A]\!]_B^p \cup \{R(d,d)\}$.
- If $D = \geqslant 1\, R.D'$ with $D' \notin \mathbf{C}_\bot^{=1}$, then $[\![\{d\} \sqcap D \sqsubseteq A]\!]_B^p := \{R(d,x) \to A(x)\}$.

To establish the claim, we recursively construct theories $T_1 := T_1(c_1, C_1)$ and $T_2 := T_2(c_1, C_1)$ that satisfy the preconditions of Lemma 7.5.3. Note that C cannot be an atomic class, nominal, Self restriction, or the negation thereof.

Consider the case $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}^p_B$. It is easy to see that $T_i(c_1, C) \coloneqq T_i(c_1, D_1) \cup \{\neg A_i(x)\} \cup \bigcup_{j=1,2} [\![\{d\} \sqcap D_j \sqsubseteq A_j]\!]^p_B$ $(i = 1, 2)$ satisfy the claim for fresh concept names A_1, A_2. Furthermore, if $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}^p_H$ and $D_2 \in \mathbf{D}^p_B$ then the claim is satisfied by $T_i(c_1, C) \coloneqq T_i(c_1, D_1) \cup [\![d \notin D_2]\!]^p$ $(i = 1, 2)$. Similarly, for the case $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}^p_H$, the theories $T_i(c_1, D_1)$ $(i = 1, 2)$ satisfy the claim.

Now consider the case $C = \geqslant n\, R.D$. Then $n = 1$ and $D \notin \mathbf{D}^p_H$. Since C is semantically equivalent to D on singleton domains, the claim follows again by induction. A similar reasoning is possible for the case $C = \leqslant n\, R.\neg D$ with $n = 0$ and $D \notin \mathbf{D}^p_H$. □

We are now, finally, in a position to state the main theorem of this section.

Theorem 7.6.11 *If C is a concept expression in DLP normal form such that $C \notin \mathcal{DLP}$, then C cannot be contained in any DLP description logic in the sense of Definition 7.3.1.*

Proof. By Definition 7.3.5, $C \notin \{\top, \bot\} \cup \mathbf{C}_H \cup \mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top}$ for all $n \geq 1$. If $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$ and $C \notin \mathbf{D}^+_a$, then the result follows by Proposition 7.6.6. If $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$ and $C \in \mathbf{D}^+_a$, then the result follows by Corollary 7.6.9. If $C \in \mathbf{D}_{\leq n}$ for some $n \geq 0$, then the result follows by Lemma 7.6.10. □

7.7 Summary

DLP provides an interesting example for the general problem of characterising syntactic fragments of a logic that are motivated by semantic properties. We derived and motivated a number of design principles for achieving such a characterisation for DLP, most notably the principles of *modularity* (closure under unions of knowledge bases) and *structurality* (closure under non-uniform renaming of signature symbols), and showed that the presented DLP description logic is the largest one possible. Formalisms like our maximal $\mathcal{DLP}$ are unnecessarily large for practical applications, but understanding overall options and underlying design principles is indispensable for making an informed choice of DL for a concrete task.

Our work also clarifies the differences between DLP and the description logics $\mathcal{SROEL}(\sqcap_s, \times)$ (and thus $\mathcal{EL}$) and Horn-$\mathcal{SHIQ}$ which can both be expressed in terms of datalog as well. First of all, neither $\mathcal{SROEL}(\sqcap_s, \times)$ nor Horn-$\mathcal{SHIQ}$ can be $\mathbf{FOL}_\approx$-emulated in datalog (DLP 2). The datalog obtained in these cases still preserves satisfiability even when arbitrary ABox facts (without complex concepts) are added. In other words, $\mathcal{SROEL}(\sqcap_s, \times)$ and Horn-$\mathcal{SHIQ}$ satisfy a weaker version of DLP 2 based on $\mathbf{FOL}^{\text{ground}}_\approx$-emulation of Definition 2.2.2, where

$\mathbf{FOL}_{\approx}^{\text{ground}}$ is the variable-free fragment of $\mathbf{FOL}_{\approx}$. Under those weakened principles, a larger space of possible DL fragments is allowed, but it is not clear whether (finitely many) maximal languages exist in this case. There is clearly no largest such language, since both $\mathcal{SROEL}(\sqcap_s, \times)$ and $\mathcal{DLP}$ abide by the weakened principles whereas their (intractable) union does not.

Even when weakening the principles of DLP like this, Horn-$\mathcal{SHIQ}$ is still excluded since it cannot be modular (DLP 5) by Proposition 7.1.1. It is open which results can be established for Horn-$\mathcal{SHIQ}$-like DLs based on the remaining weakened principles.

This chapter also heavily exploits the notion of semantic *emulation* as introduced in Section 2.2. In essence, emulation requires that a theory can take the place of another theory in all logical contexts, based on a given interface signature. Examples given in this chapter illustrate that emulation can indeed be very different from semantic equivalence. Yet, our criteria can be argued to define minimal requirements for preserving a theory's semantics even in combination with additional information, so emulation appears to be a natural tool for enabling information exchange in distributed knowledge systems. We therefore belief that notions of emulation (and, closely related, conservative extension) are natural tools for studying the semantic interplay of heterogeneous logical formalisms in general.

Finally, the approach of this chapter – seeking a structural logical fragment that is provably maximal under certain conditions – immediately leads to a number of further research questions. For example, what is the maximal fragment of SWRL that can be expressed in $\mathcal{SROIQ}$? This fragment would contain forms of DL Rules and DL-safe rules as introduced in Chapter 8 and 9. But also the maximal $\mathbf{FOL}_{\approx}$ fragment that can be expressed in some well-known subsets such as the Guarded Fragment [AvBN98] or the two-variable fragment might be of general interest. We argue that ultimate answers to such questions can indeed be obtained by a careful articulation of basic design principles. At the same time, our study indicates that the required definitions and arguments can become surprisingly complex when dealing with a syntactically rich formalism like description logic. The main reason for this is that constructs that are usually considered "syntactic sugar" have nontrivial semantic effects when considering logical fragments that are structurally closed.

7.8 Related Work

DLP has originally been introduced in [GHVD03, Vol04] although already these sources provide a number of distinct characterisations and variants. As explained in Section 6.2, the Horn DL $\mathcal{RL}$ that subsumes the abstract part of OWL RL [MCH+09] can also be considered as an extension of these works. It has been discussed above how datalog reductions for other DLs, especially for Horn-$\mathcal{SHIQ}$ and for (extensions of) $\mathcal{EL}$, relate to DLP. Further pointers to works on datalog reductions are given in Section 8.7.

We are not aware of any model-theoretic characterisation of datalog that is closely related to the constructions introduced in Section 7.5. The least model property of datalog is well-known [Llo88], and various generalised model constructions have been studied to characterise logics [CK90]. In particular, [CK90] provides a characterisation theorem for Horn sentences, stating that a first-order formula is semantically equivalent to some Horn sentence iff its validity is preserved under *reduced products*. Moreover, it is well-known that models of universal theories are closed under sub-model constructions. While both of these constructions can be related to the products in Definition 7.5.1, we point out that they are not distinguishing function-free universal Horn logic from universal Horn logic with function symbols. The reason why our constructions are adequate for establishing the given results must therefore be sought in the pre-conditions of Lemma 7.5.3 which can no longer be established if the signature includes function symbols (assuming a suitable extension of Definition 7.5.1 to such signatures).

More generally, constructions of models can be characterised by means of categorical algebra based on suitable notions of *morphisms* between models and logical theories. *Institution theory* has been proposed as a meta-logical framework for studying logics in an abstract fashion [GB92]. Institution theory uses binary relations ("$\models$") as an abstraction for various model theories, and thus is related to *formal concept analysis* (FCA) where binary relations are studied as *formal contexts* [GW97]. Indeed, the logical perspective on formal contexts is well-known [KG09], and relationships to DLs have also been explored [Rud06, Ser07]. Other approaches that can be related to institution theory in a more abstract setting are *information flow theory* and *channel theory* [Gog06, Ken09]. We are not aware of extended discussions of model constructions in any of these frameworks.

Another related branch of formal logic is the characterisation of logics based on model-theoretic properties in general, in the spirit of the original *Lindström theorem* for first-order logic (see [CK90] for an introduction). Again, we are not aware of any such characterisation for universal function-free Horn-logic, i.e. for datalog. Characterisation theorems of another type abound in modal logics and related fields, starting with the seminal characterisation result by van Benthem (see [BvBW06]). In these cases, various notions of *bisimulation* are employed to relate models, and the preservation under bisimulation then characterises formulae of certain logics. These results can yield insights for characterising DLs, and could thus be useful when investigating the problem of representing other logics in terms of DL – the converse of what was studied within this chapter.

Chapter 8

Description Logic Fragments of SWRL: DL Rules

We have already noted in previous chapters that various description logic axioms can also be presented as (datalog) rules, and, equivalently, certain datalog rules can be cast into description logic axioms with the same semantics. It is clear that there must still be rules and axioms that cannot be rewritten in this way, or at least that it is not possible to do this rewriting automatically. Otherwise, one could use a rewriting algorithm followed by a standard reasoning algorithm for datalog or description logics, respectively, to obtain a decision procedure for the combined reasoning tasks. Such a procedure cannot exist according to the undecidability result for SWRL (Fact 4.2.2).

In this chapter, we address the question which SWRL rules can be emulated by description logic knowledge bases. The class of decidable SWRL fragments that is obtained from this consideration is called *Description Logic Rules*, and different description logics lead to different classes of DL Rules depending on the expressive power that is available for emulating rules.

We begin this chapter by providing a number of motivating examples in Section 8.1, thereby introducing the essential methods that are used subsequently for emulating SWRL rules in DL. Thereafter, in Section 8.2, we define $\mathcal{SROIQ}$ rules as a large class of DL Rules that are also expressible in OWL 2. Section 8.3 further extends this framework for DLs that support logics with role constructors, where especially role conjunctions and concept products are useful for encompassing SWRL rules that could not be emulated in $\mathcal{SROIQ}$. Based on these rather large DL Rule languages, Section 8.4 provides a general definition for obtaining a DL Rule language for many further description logics, and shows that the worst-case complexity of reasoning is typically the same as for the underlying DL. The close relationship of DL Rules and datalog is exploited in Section 8.5 to obtain a direct translation procedure from $\mathcal{SROEL}(\sqcap_s, \times)$ rules to $\mathcal{SROEL}(\sqcap_s, \times)$, possibly extended with admissible range restrictions (concept products). The results of this section also establish the correctness of the reasoning procedure that was presented for $\mathcal{SROEL}(\sqcap_s, \times)$ in Section 5.4. We conclude by summarising our results in Section 8.6 and provide pointers to related work in Section 8.7.

Various results reported in this chapter have been published in [KRH08a, KRH08b, RKH08a].

8.1 Initial Observations

We first consider some examples to improve our intuition. The following rule appeared in Section 1.3:

$$\texttt{Person}(x) \wedge \texttt{authorOf}(x, y) \wedge \texttt{Book}(y) \to \texttt{Bookauthor}(x).$$

We noted that it can equivalently be expressed by the description logic axiom $\texttt{Person} \sqcap \exists\texttt{authorOf}.\texttt{Book} \sqsubseteq \texttt{Bookauthor}$. The important difference between both representations is that the latter does not use any variables. We have already seen that concept expressions play the rôle of unary predicates in SWRL. It is not necessary to state the argument of these unary predicates since it is always the same variable on both sides of a class inclusion axiom. The above example could thus equivalently be written as a SWRL rule:

$$(\texttt{Person} \sqcap \exists\texttt{authorOf}.\texttt{Book})(x) \to \texttt{Bookauthor}(x).$$

This explains the whereabouts of variable x. The variable y in turn appears only in two positions in the rule body. Since it is not referred to in any other part of the rule, it suffices to state that there exists some object with the required relationship to x, so the rule atoms $\texttt{authorOf}(x, y) \wedge \texttt{Book}(y)$ are transformed into $\exists\texttt{authorOf}.\texttt{Book}(x)$. Rewriting atoms as description logic concepts in this fashion is called *rolling-up*, since a “branch” of the rule body is rolled-up into a statement about its first variable. A graphical representation of rules that is based on this intuition is introduced in Definition 8.2.1 below.

For further examples, consider the SWRL rule base as specified in Fig. 8.1. The rules given there have already been discussed in Chapter 4, but we introduce a number of additional facts that will be useful for subsequent discussions.[1] Indeed, we will refer to Fig. 8.1 as a running example throughout this and the next chapter. When reasoning as in Section 4.2.1, it is not hard to see that the rule base entails that `anja`, `bijan`, `ian`, and `markus` are – for the sake of the example – `Unhappy`, and the degree to which these conclusions are preserved will support our intuition when comparing various SWRL fragments that are studied below.

[1] Any similarities with real vegetarians, whether happy or not, are entirely coincidental.

(1)	$\mathtt{Vegetarian}(x) \land \mathtt{FishProduct}(y) \to \mathtt{dislikes}(x, y)$
(2)	$\mathtt{orderedDish}(x, y) \land \mathtt{dislikes}(x, y) \to \mathtt{Unhappy}(x)$
(3)	$\mathtt{orderedDish}(x, y) \to \mathtt{Dish}(y)$
(4)	$\mathtt{dislikes}(x, z) \land \mathtt{Dish}(y) \land \mathtt{contains}(y, z) \to \mathtt{dislikes}(x, y)$
(5)	$\mathtt{Happy}(x) \land \mathtt{Unhappy}(x) \to \bot$

$\mathtt{Vegetarian}(\mathtt{anja})$	$\mathtt{orderedDish}(\mathtt{anja}, \mathtt{thaiRedCurry})$
	$\exists\mathtt{contains}.\mathtt{FishProduct}(\mathtt{thaiRedCurry})$
$\mathtt{Vegetarian}(\mathtt{bijan})$	$\mathtt{orderedDish}(\mathtt{bijan}, \mathtt{fishFingers})$
	$\mathtt{FishProduct}(\mathtt{fishFingers})$
$\mathtt{Vegetarian}(\mathtt{ian})$	$\exists\mathtt{orderedDish}.\exists\mathtt{contains}.\{\mathtt{fishSauce}\}(\mathtt{ian})$
	$\mathtt{FishProduct}(\mathtt{fishSauce})$
$\mathtt{Vegetarian}(\mathtt{markus})$	$\exists\mathtt{orderedDish}.\exists\mathtt{contains}.\mathtt{FishProduct}(\mathtt{markus})$

Figure 8.1: Example SWRL rule base

We can now try to generalise from the first example given above. We have seen that x in the above case is simply an implicit (and necessary) part of the concept inclusion axiom. So for any rule that we wish to rewrite as such an axiom, we need to identify some variable x which plays this special role, and find a way to eliminate all other variables from the rule using a rolling-up method as above. This is not always possible, as rule (2) from Fig. 8.1 illustrates. The conclusion of this rule suggests that the variable y should be eliminated to obtain a class inclusion axiom. But the premise of the rule cannot be rewritten as above. A class expression like $\exists\mathtt{orderedDish}.\top \sqcap \exists\mathtt{dislikes}.\top$ describes elements with relationships $\mathtt{orderedDish}$ and $\mathtt{dislikes}$, but not necessarily to the same element y. Using inverse roles, one could also write $\exists\mathtt{orderedDish}.\exists\mathtt{dislikes}^{-}.\top$ to describe some x who ordered something that is disliked by someone – but not necessarily by x. Indeed, this relationship can only be expressed in DLs that support conjunctions of roles as discussed in Chapter 5.

Yet, there are various further types of datalog (or SWRL) rules that can be expressed in DL axioms. An example is rule (4) of Fig. 8.1. As its conclusion is a binary atom, it can certainly not be expressed as a concept inclusion axiom. $\mathcal{SROIQ}$ role inclusion axioms, on the other hand, can include only role expressions, while rule (4) also contains a unary (concept) atom $\mathtt{Dish}(y)$. This problem can be addressed by adding an auxiliary axiom to the knowledge base. A fresh role name $R_{\mathtt{Dish}}$ is introduced together with the concept inclusion axiom $\mathtt{Dish} \equiv \exists R_{\mathtt{Dish}}.\mathsf{Self}$. Intuitively speaking, this defines the class of dishes to be equivalent to the class of those things which have the relationship $R_{\mathtt{Dish}}$ to themselves. With this additional axiom, one can rewrite rule (4) as follows:

$$\texttt{dislikes}(x,z) \wedge R_{\texttt{Dish}}(y,y) \wedge \texttt{contains}(y,z) \rightarrow \texttt{dislikes}(x,y).$$

This step is the core of the transformation to $\mathcal{SROIQ}$. Using inverse roles, we can now write the rule premise as a chain:

$$\texttt{dislikes}(x,z) \wedge \texttt{contains}^{-}(z,y) \wedge R_{\texttt{Dish}}(y,y) \rightarrow \texttt{dislikes}(x,y).$$

This rule can now easily be expressed as a $\mathcal{SROIQ}$ role inclusion axiom. Together with the auxiliary axiom we have used above, rule (4) is thus represented by the following description logic knowledge base:

$$\texttt{Dish} \equiv \exists R_{\texttt{Dish}}.\mathsf{Self}$$
$$\texttt{dislikes} \circ \texttt{contains}^{-} \circ R_{\texttt{Dish}} \sqsubseteq \texttt{dislikes}$$

Note that the second axiom no longer contains the requirement that $R_{\texttt{Dish}}$ refers to the same variable in first and second position. Indeed, the resulting knowledge base is not semantically equivalent to the original rule but it can be shown to semantically emulate it.

While these examples provide us with a significant set of tools for translating rules into axioms, there is still a case that we have not addressed yet. Consider rule (1) of Fig. 8.1. Again, the conclusion of the rule is a binary atom, so the use of a role inclusion axiom seems to be required. Yet, even if we use the above method for replacing the unary predicates $\texttt{Vegetarian}(x)$ and $\texttt{FishProduct}(y)$ with new auxiliary roles, we only obtain the following rule:

$$R_{\texttt{Vegetarian}}(x,x) \wedge R_{\texttt{FishProduct}}(y,y) \rightarrow \texttt{dislikes}(x,y).$$

But this cannot be rewritten as a role composition axiom, since there is a "gap" between x and y. This problem can be overcome by inserting the universal role U to connect x and y:

$$R_{\texttt{Vegetarian}}(x,x) \wedge U(x,y) \wedge R_{\texttt{FishProduct}}(y,y) \rightarrow \texttt{dislikes}(x,y).$$

Since the relation denoted by U is defined to comprise all pairs of individuals, adding the atom $U(x,y)$ does not impose any restrictions on the applicability of the rule. Yet it helps us to bring the rule into the right syntactic shape for being expressed in $\mathcal{SROIQ}$. Together with the required auxiliary axioms, we thus obtain:

$$\texttt{Vegetarian} \equiv \exists R_{\texttt{Vegetarian}}.\mathsf{Self}$$
$$\texttt{FishProduct} \equiv \exists R_{\texttt{FishProduct}}.\mathsf{Self}$$
$$R_{\texttt{Vegetarian}} \circ U \circ R_{\texttt{FishProduct}} \sqsubseteq \texttt{dislikes}$$

These examples already sketch the most important techniques for transforming SWRL rules into $\mathcal{SROIQ}$ knowledge bases. A final aspect that was not covered yet is the use of constant symbols (individual names) in rules. Since $\mathcal{SROIQ}$ features nominals, constant symbols can be used to transform role atoms into concept atoms. For example, the rule body $R(x, a) \wedge S(c, d)$ can be expressed as $\exists R.\{a\}(x) \wedge \exists S.\{d\}(c)$. Constants in concept atoms in turn can be removed by introducing fresh variables. In the previous example, we obtain $\exists R.\{a\}(x) \wedge \exists S.\{d\}(y) \wedge \{c\}(y)$.

8.2 Defining $\mathcal{SROIQ}$ Rules

Based on the above observations, we can provide a formal definition of DL Rules for $\mathcal{SROIQ}$. Clearly, the details of the definition depend on the capabilities of the available description logic. Our above examples have employed role composition ($\circ$), inverse roles ($\cdot^-$), and the universal role U, as well as local reflexivity (Self) and nominal concepts. Moreover, if a $\mathcal{SROIQ}$ knowledge base is to be obtained, additional structural restrictions like regularity and simplicity of roles need to be taken into account. Therefore, we first provide a general definition of $\mathcal{SROIQ}^{\text{free}}$ rules which is then strengthened to obtain a definition of $\mathcal{SROIQ}$ rules that accounts for these requirements.

A main criterion for deciding whether or not a rule can be formulated in description logic is the structure of variable dependencies within the rule body. This structure is described by the *dependency graph* of the rule body – the directed graph with the body's variables and constants as its nodes, and with the role atoms $R(s, t)$ representing edges from s to t. With this intuition in mind, we can define some graph-theoretic notions for the body of a rule. Here and in the following, it is often convenient to consider the body as a set of atoms and to use the according notation.

Definition 8.2.1 Consider a SWRL rule $B \to H$.

- A *path* in B from a term s to a term s' is a set $\{S_1(s, t_2), \ldots, S_n(t_n, s')\} \subseteq B$ with $n \geq 1$. In this case, n is the *length* of the path.
- Two terms s and s' are *connected* in B if either $s = s'$, or there is a sequence of terms $s = t_1, t_2, \ldots, t_n = s'$ such that, for all $i \in \{1, \ldots, n-1\}$, there is either a path from t_i to t_{i+1} or a path from t_{i+1} to t_i.
- A term t is a *root* in B if t occurs in B, and there is no path in B to t.
- B is *tree-shaped* if B contains exactly one root t, and there is exactly one path from t to any term s in B. In this case, t is *the* root of B.
- If H is of the form $C(t)$ or $R(t, s)$, then t is *the root* of H. ◇

(1) Head Normalisation

- If H is empty, then set $H := \bot(x)$, where $x \in \mathbf{V}$ is arbitrary.
- For each variable x in H: if x does not occur in B, then set $B := B \wedge \top(x)$.

(2) Role Atom Normalisation For all atoms $R(s,t)$ in $B \to H$:

- if $t \in \mathbf{I}$, then replace $R(s,t)$ by $\exists R.\{t\}(s)$, else
- if $s \in \mathbf{I}$, then consider a variable y not occurring in $B \to H$, replace $R(s,t)$ by $R(y,t)$, and set $B := B \cup \{\{s\}(y)\}$, else
- if $s = t$, then replace $R(s,t)$ by $\exists R.\mathsf{Self}(t)$.

(3) Concept Atom Normalisation For all atoms $C(s)$ in $B \to H$:

- if $s \in \mathbf{I}$, then consider a variable y not occurring in $B \to H$, replace $C(s)$ by $C(y)$, and set $B := B \cup \{\{s\}(y)\}$.

(4) Connecting the Body For every root x of B:

- if x is not connected in B to the root z of H, then $B := B \cup \{U(z,x)\}$.

(5) Orienting the Body For every role atom $R(x,y) \in B$:

- if the root of H is not connected to x in $B \setminus \{R(x,y)\}$, then $B := B \cup \{\mathrm{Inv}(R)(y,x)\} \setminus \{R(x,y)\}$.

Figure 8.2: Normalising a SWRL rule $B \to H$

In essence, we are interested in SWRL rules with a tree-shaped body. However, $\mathcal{SROIQ}$ allows for a number of exceptions to that structure as illustrated by the examples in Section 8.1. To deal with these cases without introducing more complex graph-theoretic properties, we define a normalisation procedure for SWRL rules. The normalisation procedure for a SWRL rule $B \to H$ is specified in Fig. 8.2. Step (1) ensures that head and body are non-empty, and that all variables in the head occur in the body as well. Steps (2) and (3) together ensure that the rule does not contain any constant symbols as parameters of atoms. Note that the second case of (2) could have been addressed like the first case, using $\mathrm{Inv}(R)$ instead of R. This would slightly increase the obtained class of $\mathcal{SROIQ}$ rules as defined below, but it would make the normalisation algorithm less useful for defining DL Rules in description logics without inverse roles. Step (4) ensures that all terms of the body are connected in the sense of Definition 8.2.1, while step (5) attempts to orient the role atoms in the body to "point away" from the root variable of the head.

It is readily seen that the normal form of a SWRL rule is semantically equivalent to the original rule, since all of the individual transformation steps ensure semantic equivalence. Moreover, it is obvious that the procedure terminates, since it only iterates over a limited number of elements in each step (all iterations are assumed to refer to the elements found in the rule $B \to H$ as encountered *before* starting the iteration, i.e. subsequent changes to $B \to H$ will not affect the iterations). We can sum up those observations as follows:

Lemma 8.2.2 *For any SWRL rule $B \to H$, the normalisation algorithm of Fig. 8.2 produces a semantically equivalent SWRL rule $B' \to H'$. The time of this computation and the size of $B' \to H'$ are linearly bounded in the size of $B \to H$.*

Note that step (4) of the normalisation is non-deterministic, as illustrated by the example rule $A(x) \wedge R(y_1, z) \wedge S(y_2, z) \to C(x)$. The rule body contains three roots x, y_1, and y_2, of which the latter two are not connected to x when entering step (4). Depending on the order of iteration, we obtain either $A(x) \wedge R(y_1, z) \wedge S(y_2, z) \wedge U(x, y_1) \to C(x)$ or $A(x) \wedge R(y_1, z) \wedge S(y_2, z) \wedge U(x, y_2) \to B(x)$. The subsequent step (5) then results in either $A(x) \wedge U(x, y_1) \wedge R(y_1, z) \wedge S^-(z, y_2) \to C(x)$ or $A(x) \wedge U(x, y_2) \wedge S(y_2, z) \wedge R^-(z, y_1) \to C(x)$. We assume that this non-determinism is prevented by using some arbitrary but fixed iteration order for processing variables in step (4). The remaining steps are deterministic, if the choice of fresh variable names y is assumed to be deterministic. With these assumptions, it makes sense to speak of the (unique) *normal form* of some SWRL rule. We can now define $\mathcal{SROIQ}^{\text{free}}$ rules:

Definition 8.2.3 A SWRL rule $B \to H$ in normal form is a $\mathcal{SROIQ}^{\text{free}}$ *rule* if the following conditions are satisfied

– the rule contains only atoms of the form $C(s)$ and $R(s,t)$ where C is a concept expression and R is a role expression of $\mathcal{SROIQ}^{\text{free}}$,
– B is tree-shaped and has the same root as H,
– if $H = R(x,y)$ with $R \notin \mathbf{R}_\text{n}$ and $x, y \in \mathbf{V}$, then $B = S(x,y)$ with $S \in \mathbf{R}_\text{s}$ simple.

A SWRL rule is a $\mathcal{SROIQ}^{\text{free}}$ *rule* if its normal form is a $\mathcal{SROIQ}^{\text{free}}$ rule. ◇

The first condition is an obvious requirement for transforming SWRL rules into $\mathcal{SROIQ}^{\text{free}}$ knowledge bases, and especially it precludes the use of predicate symbols that are not part of the DL signature as induced by the given SWRL signature. The second condition ensures that the relations between variables in B can be expressed in $\mathcal{SROIQ}^{\text{free}}$. The third condition, finally, defines general restrictions on the use of simple roles that will become relevant in various fragments of $\mathcal{SROIQ}^{\text{free}}$ rules. It is easy to check that the rules (1), (3), (4), and (5) of Fig. 8.1 are indeed covered by this definition, where `dislikes` is the only role name that must be non-simple. Unfortunately, in the absence of rule (2), no conclusions are obtained regarding instances of `Unhappy`. $\mathcal{SROIQ}$ rules are obtained by adding regularity restrictions to this definition:

Definition 8.2.4 Given a $\mathcal{SROIQ}^{\text{free}}$ rule of the normal form $B \to R(x,y)$ with $x, y \in \mathbf{V}$, let $\rho(B,x,y) \subseteq B$ be the unique path $\rho(B,x,y) := \{S_1(x,x_2),\ldots,S_n(x_n,y)\}$ from x to y.

A $\mathcal{SROIQ}^{\text{free}}$ rule base RB is *regular* if there is a strict (irreflexive) total order $\prec$ on $\mathbf{R}$ such that

– for $R \notin \{S, \text{Inv}(S)\}$, we find $S \prec R$ iff $\text{Inv}(S) \prec R$, and
– for every rule $B \to R(s,t) \in \text{RB}$ and normal form $B' \to R(x,y)$ with $x, y \in \mathbf{V}$, the set $\rho(B',x,y)$ is of one of the forms:

$$\{R(x,z), R(z,y)\}, \qquad \{\text{Inv}(R)(x,y)\},$$
$$\{R_1(x,z_2), R_2(z_2,z_3), \ldots, R_n(z_n,y)\},$$
$$\{R(x,z_1), R_1(z_1,z_2), \ldots, R_n(z_n,y)\}, \qquad \{R_1(x,z_1), \ldots, R_n(z_{n-1},z_n), R(z_n,y)\}$$

such that $R, R_1, \ldots, R_n \in \mathbf{R}$, and $R_i \prec R$ for $i = 1, \ldots, n$.

A $\mathcal{SROIQ}$ *rule base* is a regular $\mathcal{SROIQ}^{\text{free}}$ rule base that contains only $\mathcal{SROIQ}$ concept and role expressions. A $\mathcal{SROIQ}$ *rule* is a $\mathcal{SROIQ}^{\text{free}}$ rule that occurs in some $\mathcal{SROIQ}$ rule base. ◇

The previous definition should be compared with Definition 3.1.4 which introduced analogous restrictions for defining $\mathcal{SROIQ}$. The main difference in the definition of regularity for rules is that we now need to determine the set $\rho(B,x,y)$

of role atoms that are actually relevant for this restriction. As in the case of RBox axioms for $\mathcal{SROIQ}$, certain $\mathcal{SROIQ}^{\text{free}}$ rules cannot be part of any $\mathcal{SROIQ}$ rule base, as illustrated by the rule $R(x, y) \land S(y, z) \land R(z, v) \to R(x, v)$. Moreover, it should be marked that the restrictions that are imposed on $\mathcal{SROIQ}$ concept expressions in Definition 3.1.4 depend on the declarations of simple role names $\mathbf{N}_s$ in the given signature, which also constrain the general structure of $\mathcal{SROIQ}^{\text{free}}$ rules (Definition 8.2.3). In the case of Self, this entails that $\mathcal{SROIQ}$ rules must not entail atoms of the form $R(x, x)$ for non-simple R.

Definition 8.2.4 and 8.2.3 make no attempt to maximise the defined class of DL Rules. In principle, it would be feasible to do this in a systematic way, to determine a maximal DL Rule language (for a given DL) similar to the maximal DLP language that was studied in Chapter 7. But said chapter also illustrates that such a canonical treatment may require rather complex and technically involved arguments, especially due to the rich syntax of description logics. In the case of DL Rules, a maximal structural fragment would need to combine the above insights on DL Rules with the approach of *DL-safe variables* as introduced in Chapter 9, since the latter can also be formulated in a structurally stable way as long as structurality is not extended to variable names.[2] The endeavour of fully characterising a maximal DL Rule language is clearly beyond the scope of this work.

It is not hard to find concrete examples of SWRL rules that are not covered by the above algorithm/definition even though they can be expressed in $\mathcal{SROIQ}$. For example, the rule $\geqslant 2\, R.\top(x) \land \{a\}(x) \land \{b\}(y) \to R(x, y)$ requires R to be non-simple so that $\geqslant 2\, R.\top$ is not a $\mathcal{SROIQ}$ concept. Yet, the $\mathcal{SROIQ}$ knowledge base $\{\{a\} \sqcap \geqslant 2\, R.\top \sqsubseteq \exists R.\{b\}\}$ emulates the rule, where R is now a simple role name as required. Further examples include rules like $R(x, y) \land S(x, y) \to U(x, y)$ that are semantically trivial even though they do not satisfy the restrictions on $\mathcal{SROIQ}^{\text{free}}$ rules. Another significant class of SWRL rules that can be expressed in $\mathcal{SROIQ}$ are DL-safe rules as introduced in Chapter 9, although they may require exponentially large knowledge bases.

The essential property of $\mathcal{SROIQ}^{\text{free}}$ rules is that they can be emulated by $\mathcal{SROIQ}^{\text{free}}$ knowledge bases. An algorithm for obtaining a suitable knowledge base KB from a $\mathcal{SROIQ}^{\text{free}}$ rule $B \to H$ is specified in Fig. 8.3. The basic techniques applied here – rolling up side branches and replacing concept atoms by role atoms in RIAs – have already been explained in the earlier examples. Note that, for all inputs $B \to H$, the preconditions for creating either a concept inclusion axiom or a role inclusion axiom must be satisfied. If $H' = D(x)$, then B'

[2]A DL Rules language that is closed under renaming of single occurrences of variables is clearly overly limited, since it could not capture essential parts of the above definition of $\mathcal{SROIQ}^{\text{free}}$ rules; it would thus be significantly less expressive than $\mathcal{DLP}$.

Input: A $\mathcal{SROIQ}^{\text{free}}$ rule $B \to H$
Output: A $\mathcal{SROIQ}^{\text{free}}$ knowledge base KB

Initialise $B' \to H'$ to be the normal form of $B \to H$

Roll Up Side Branches

- While B' contains an atom $R(x, y)$ where y occurs in no other role atom of $B' \to H'$: Define the set $B_y := \{C(t) \in B' \mid t = y\}$, and a concept $E := \bigsqcap_{C(y) \in B_y} C$ where the empty conjunction is assumed to be $\top$. Then set $B' := B' \cup \{\exists R.E(x)\} \setminus B_y$.

Create Concept Inclusion Axiom If H' is of the form $D(x)$:

- In this case, B' is of the form $C_1(x) \wedge \ldots \wedge C_n(x)$. Set KB $:= \{C_1 \sqcap \ldots \sqcap C_n \sqsubseteq D\}$.

Create Role Inclusion Axiom If H' is of the form $S(x, y)$:

- Initialise KB $:= \emptyset$.
- For each concept atom $C(z)$ in B': Set $B' := B' \cup \{R_C(z, z)\} \setminus \{C(z)\}$ where R_C is a fresh simple role name, and set KB $:=$ KB $\cup \{C \equiv \exists R_C.\mathsf{Self}\}$.
- Now B' is of the form $R_1(x, x_2) \wedge \ldots \wedge R_n(x_n, y)$. Set KB $:=$ KB $\cup \{R_1 \circ \ldots \circ R_n \sqsubseteq S\}$.

Figure 8.3: Transforming $\mathcal{SROIQ}^{\text{free}}$ rules into $\mathcal{SROIQ}^{\text{free}}$ knowledge bases

contains only concept atoms. This must be the case after rolling-up, since H' contains no binary atoms, and since B' is tree-shaped. If $H' = S(x, y)$, then B' must eventually have the form $R_1(x, x_2) \wedge \ldots \wedge R_n(x_n, y)$. To see this, first note that all concept atoms $C(z)$ in B' have been replaced by role atoms $R_C(z, z)$. Second, every role atom $R(v, w) \in B'$ with $w \neq y$ and for which there no atom of the form $S(w, z) \in B'$ must have been eliminated when rolling up side branches. Thus the remaining atoms must all be role atoms that form a chain.

While the correctness of the transformation will be shown below, the algorithm does not always produce the result that might be considered most obvious. On the one hand, the algorithm never generates ABox axioms, which would sometimes lead to a simpler presentation. As an example, the simple SWRL fact $R(a, b)$ leads to the normalised rule $\{a\}(x) \to \exists R.\{b\}(x)$ for which the algorithm creates the knowledge base $\{\{a\} \sqsubseteq \exists R.\{b\}\}$. This captures the intended semantics but it is rather not the preferred way of expressing the original statement. On the other hand, the algorithm does not implement any additional simplifications that can be admissible in some cases. For example, the rule $\{a\}(x) \wedge R(x, y) \wedge \{b\}(y) \to S(x, y)$ can be expressed as a GCI $\{a\} \sqcap \exists R.\{b\} \sqsubseteq \exists S.\{b\}$, whereas the transformation algorithm produces a knowledge base $\{\{a\} \equiv \exists R_{\{a\}}.\mathsf{Self}, \{b\} \equiv \exists R_{\{b\}}.\mathsf{Self}, R_{\{a\}} \circ R \circ R_{\{b\}} \sqsubseteq S\}$.

The output KB of the transformation algorithm for some input rule $B \to H$ will be denoted by KB($B \to H$). This notation is extended to $\mathcal{SROIQ}^{\text{free}}$ rule bases RB by defining KB(RB) $\coloneqq \bigcup_{B\to H\in \text{RB}}$ KB($B \to H$). We can now state the main result of this section.

Theorem 8.2.5 *Every* $\mathcal{SROIQ}^{\text{free}}$ *rule base* RB *is semantically emulated by the* $\mathcal{SROIQ}^{\text{free}}$ *knowledge base* KB(RB)*, the size of which is linearly bounded by the size of* RB.

If RB *is a* $\mathcal{SROIQ}$ *rule base, then* KB(RB) *is a* $\mathcal{SROIQ}$ *knowledge base. Thus the problem of deciding consistency of a* $\mathcal{SROIQ}$ *rule base is* N2ExpTime-*complete.*

Proof. To show that KB(RB) is a $\mathcal{SROIQ}^{\text{free}}$ knowledge base, we must verify that the use of simple role names agrees with Definition 3.1.2. Thus consider the creation step for role inclusion axioms. If the head of the rule is of the form $S(x, y)$ in this step, it must have been of this form in the normalised input rule already. Since S is simple, Definition 8.2.3 implies that $B = R(x, y)$ with $R \in \mathbf{R}_{\text{s}}$. Applying the transformation steps to a rule of this form, it is easy to see that we obtain a rule $R_1(x, y) \to S(x, y)$ in Step 6b, so the generated RIA is indeed allowed in $\mathcal{SROIQ}^{\text{free}}$.

We have already observed in Lemma 8.2.2 that the size of the normal form of a $\mathcal{SROIQ}^{\text{free}}$ rule is linearly bounded, and it is easy to see that this property also holds for the size of KB(RB).

It suffices to show semantic emulation for rule bases that consist of a single rule $B \to H$. This result is established by showing that the following property is preserved throughout the transformation algorithm: KB $\cup \{B' \to H'\}$ semantically emulates $B \to H$, where $B' \to H'$ denotes the modified input rule at the current stage of the computation. By Lemma 8.2.2, the claim holds after the algorithm's initialisation, and it is easily verified that its validity is preserved by each individual transformation step.

Now assume that RB is a $\mathcal{SROIQ}$ rule base. It is obvious that KB(RB) contains only $\mathcal{SROIQ}$ role and concept expressions. It remains to show that KB(RB) satisfies the regularity restrictions of $\mathcal{SROIQ}$. Thus let $\prec$ denote the strict total order on $\mathbf{R}$ that exists for RB according to Definition 8.2.4. We claim that it satisfies all properties of the strict total order that is required to show regularity of KB(RB) based on Definition 3.1.4. For this it suffices to show that the path $\rho(B, x, y)$ is indeed exactly the set of those role atoms that are considered when creating the final role inclusion axiom in Fig. 8.3 for an input rule of the form $B \to R(x, y)$. This is easy to see by noting that rolling-up eliminates exactly those role atoms that are not included in $\rho(B, x, y)$.

Now the claimed complexity result follows from the well-known worst-case complexity result for $\mathcal{SROIQ}$ [Kaz08]. □

In particular, this shows that we have indeed identified a decidable SWRL fragment. Since the above worst-case complexity is very high, it does not allow us to draw conclusions about the practical implementability of $\mathcal{SROIQ}$ rules. In this respect, it is more important that we can provide a linear-time transformation from $\mathcal{SROIQ}$ rules to $\mathcal{SROIQ}$. Given that the computed knowledge base is not significantly larger than the original rule base, it is feasible to use this transformation to implement inferencing for $\mathcal{SROIQ}$ rules. In contrast, the considerations in Section 9.3 illustrate that an N2ExpTime worst-case complexity can sometimes even be established if a transformation incurs an exponential blow-up in the size of the knowledge base.

Based on the existing practical experiences with $\mathcal{SROIQ}$ inference engines, it can thus be argued that inferencing for $\mathcal{SROIQ}$ rules can also be accomplished with realistic computing resources in relevant cases. However, the actual performance in "average" cases strongly depends on the structure of typical knowledge bases that are obtained from $\mathcal{SROIQ}$ rule bases. It is likely that these knowledge bases are rather different from today's $\mathcal{SROIQ}$ knowledge bases, since DL Rules provide a different modelling metaphor that emphasises expressive features of $\mathcal{SROIQ}$ that are hard to access when constructing $\mathcal{SROIQ}$ axioms directly.

8.3 Adding Role Constructors

Bodies of SWRL rules are arbitrary conjunctions of atoms, whereas $\mathcal{SROIQ}$ supports only conjunctions of concepts but not of roles. Based on the insights that have been obtained in Chapter 5, it is therefore natural to extend $\mathcal{SROIQ}$ rules to $\mathcal{SROIQ}(B_s, \times)$ rules by allowing simple role expressions as well. This also allows for the use of concept product expressions to formulate rules of the form $A(x) \wedge B(y) \to R(x, y)$ directly as $(A \times B)(x, y) \to R(x, y)$. This may look like an unnecessary complication at first since the respective rule could also be expressed in $\mathcal{SROIQ}$, but a closer inspection shows that the use of concept products allows R to be simple whereas $\mathcal{SROIQ}$ can only emulate this rule if R is non-simple.

$\mathcal{SROIQ}(B_s, \times)$ (or $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$) rules in their general form might be less relevant than $\mathcal{SROIQ}$ rules in practice, since there is no sufficient tool support for reasoning in $\mathcal{SROIQ}(B_s, \times)$, but they provide the largest DL Rule language considered within this work. Moreover, $\mathcal{SROIQ}(B_s, \times)$ rules serve as a convenient conceptual framework for $\mathcal{SROEL}(\sqcap_s, \times)$ rules that are more likely to play a role in practical applications, and for which an inferencing algorithm is specified in Section 8.5.

Note that our definition of SWRL assumes $\mathcal{SROIQ}^{\text{free}}$ as the underlying DL,

(1) Head Normalisation
(2) Role Atom Normalisation
(3) Concept Atom Normalisation

(3.a) Concept Product Normalisation If $H = R(x, y)$ with $R \in \mathbf{R}_\mathrm{s}$ then:

- define a set $B_a := \{C(z) \in B \mid z$ occurs in no role atom of $B \to H\}$, and concepts $E := \bigsqcap_{C(x)\in B} C \sqcap \bigsqcap_{C(z)\in B_a} \exists U.C(x)$ and $F := \bigsqcap_{C(y)\in B} C$,
- define $B := B \cup \{(E \times F)(x, y)\} \setminus (\{C(t) \in B \mid t = x \text{ or } t = y\} \cup B_a)$.

(3.b) Role Conjunction Normalisation For all role atoms $R(x, y) \in B$:

- if $R \in \mathbf{R}_\mathrm{s}$ and there is some $S(x, y) \in B$ with $S \in \mathbf{R}_\mathrm{s}$, then $B := B \cup \{(R \sqcap S)(x, y)\} \setminus \{S(x, y), R(x, y)\}$, else
- if $R \in \mathbf{R}_\mathrm{s}$ and there is some $S(y, x) \in B$ with $S \in \mathbf{R}_\mathrm{s}$, then $B := B \cup \{(\mathrm{Inv}(R) \sqcap S)(y, x)\} \setminus \{S(y, x), R(x, y)\}$.

(4) Connecting the Body
(5) Orienting the Body

Figure 8.4: Normalising a SWRL rule $B \to H$ in $\mathcal{SROIQ}(B_\mathrm{s}, \times)^{\text{free}}$

so that role constructors cannot occur in such rules. However, it is easy to see that this restriction is not essential, since role expressions are hardly considered when processing DL Rules. Large parts of the transformation algorithm for $\mathcal{SROIQ}$ rules simply preserve role expressions, so that complex role expressions behave like atomic roles (role names). The only exception is step (5) in Fig. 8.2, where a role expression is replaced by its inverse. When dealing with $\mathcal{SROIQ}(B_\mathrm{s}, \times)$ roles, this operation must of course be defined as in Section 5.1. In the following, we will therefore tacitly assume that SWRL rules may contain complex role expressions as well, and in particular we admit such expressions in intermediate results that are created when transforming rules. Whether or not they are supported in the input rule base is not essential to our presentation, and in particular we obtain a larger decidable fragment of SWRL without role constructors as well.

Due to the simplicity requirement of $\mathcal{SROIQ}(B_\mathrm{s}, \times)$ role expressions, it is still impossible to model conjunctions of chains of roles, and, in essence, we therefore still require the dependency graph of a $\mathcal{SROIQ}(B_\mathrm{s}, \times)^{\text{free}}$ rule to be free of undirected cycles, and large parts of the definition for $\mathcal{SROIQ}^{\text{free}}$ rules can be re-used. The most efficient way of defining $\mathcal{SROIQ}(B_\mathrm{s}, \times)^{\text{free}}$ rules thus is to extend the normal form transformation of $\mathcal{SROIQ}^{\text{free}}$ rules to take advantage of the additional expressive features. To this end, consider the extended normalisation procedure as specified in Fig. 8.4, where steps (1)–(3), (4), and (5) are the same as in Fig. 8.2.

The new steps (3.a) and (3.b) use role constructors to address cases that could not be handled in $\mathcal{SROIQ}^{\mathrm{free}}$. In (3.a), concept products are used to avoid violations of simplicity constraints for roles. The first step defines auxiliary concepts that are used to combine various concept atoms into a single expression E and F. Note that the construction of E corresponds to the construction of GCIs from $\mathcal{SROIQ}^{\mathrm{free}}$ rules that do not contain role atoms, where the universal role is used to connect independent atoms to the root variable x of the head. Step (3.b) iteratively combines simple role expressions using role conjunction, where some expressions might need to be inverted in the process.

It is not hard to see that the extended $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ normal form of a rule is again semantically equivalent to the original rule. As in the case of $\mathcal{SROIQ}^{\mathrm{free}}$ normal forms of rules, we assume an arbitrary but fixed global ordering for iterating over role atoms, so that the normalisation of $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ rules is indeed deterministic.

The definition of $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ and $\mathcal{SROIQ}(B_{\mathrm{s}},\times)$ rules and rule bases now is completely analogous to the definition of the corresponding notions for $\mathcal{SROIQ}^{\mathrm{free}}$ and $\mathcal{SROIQ}$.

Definition 8.3.1 $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ *rules* are defined as in Definition 8.2.3 but using $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ instead of $\mathcal{SROIQ}^{\mathrm{free}}$ in all places. A $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ *rule base* is a set of $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ rules. *Regular* $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ rule bases and $\mathcal{SROIQ}(B_{\mathrm{s}},\times)$ *rules* and *rule bases* are defined as in Definition 8.2.4 but using $\mathcal{SROIQ}(B_{\mathrm{s}},\times)$ and $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ instead of $\mathcal{SROIQ}$ and $\mathcal{SROIQ}^{\mathrm{free}}$, respectively, in all places. $\diamond$

Due to the structural similarity of $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ rules and $\mathcal{SROIQ}^{\mathrm{free}}$ rules, it is easy to see that the transformation of Fig. 8.3 can also be used to transform $\mathcal{SROIQ}(B_{\mathrm{s}},\times)^{\mathrm{free}}$ rules. Theorem 8.2.5 and 5.2.2 thus can be combined into the following result:

Theorem 8.3.2 *The problem of deciding satisfiability of* $\mathcal{SROIQ}(B_{\mathrm{s}},\times)$ *rule bases is* N2ExpTime*-complete.*

Returning to our running example from Fig. 8.1, we see that rule (2) can now be supported since it is transformed to a role conjunction in step (3.b) of the normalisation of Fig. 8.4. However, the resulting role expression is only allowed in $\mathcal{SROIQ}(B_{\mathrm{s}},\times)$ if the role names `orderedDish` and `dislikes` are simple. This, however, conflicts with rule (4) of Fig. 8.1 that requires `dislikes` to be non-simple. Therefore, depending on the choice of simple and non-simple roles, we find that either rules (1), (3), (4), and (5), or rules (1), (2), (3), and (5) can occur together within a single $\mathcal{SROIQ}(B_{\mathrm{s}},\times)$ rule base. The former rule base is also supported by $\mathcal{SROIQ}$ rules, but does not allow for any conclusions regarding unhappy individuals. The latter rule base still completely lacks rule (4), but allows us to conclude `Unhappy(bijan)`.

8.4 Further Classes of DL Rules

The above discussions were focussed on highly expressive DL Rule languages, suitable for identifying rather large decidable fragments of SWRL. In this section, we show how to define DL Rules for smaller description logics, and derive some immediate complexity results. The discussion in this section is generally based on $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ rules as the most general decidable fragment of SWRL introduced above, and the normal form construction of rules thus will refer to the extended transformation in Fig. 8.4.

But already the normalisation of Fig. 8.2 has the drawback of introducing nominals into rules in steps (2) and (3). The elimination of constant symbols from rules allowed for a more unified treatment of DL Rules, and the rewriting of $R(x, a)$ to $\exists R.\{a\}(x)$ in rule heads directly enlarges the class of DL Rules by avoiding unnecessary simplicity restrictions that would apply when translating such rules into RIAs. However, the use of nominals for encoding constants excludes a number of description logics where this feature is not available, which is especially undesirable since the use of constants is common in rule-based modelling. Fortunately, nominals are not necessary when giving up semantic equivalence in favour of emulation, as long as the nominal occurs in a negative position, i.e., in essence, if it occurs non-negated in a rule body or negated in a rule head.

Lemma 8.4.1 *Consider a SWRL rule base* RB *in the normal form as obtained by the transformation in Fig. 8.4. A rule base* RB′ *is obtained from* RB *by executing the following steps for each individual symbol $a \in \mathbf{I}$:*

- *introduce a fresh concept name N_a and add a new fact $N_a(a)$,*
- *in all concept atoms $C(x)$ in a rule body of* RB *that contain a subconcept $\{a\}$ in a position p with $\mathsf{pol}(C, p) > 0$, replace this occurrence $\{a\} = C|_p$ by N_a,*
- *in all concept atoms $C(x)$ in a rule head of* RB *that contain a subconcept $\{a\}$ in a position p with $\mathsf{pol}(C, p) < 0$, replace this occurrence $\{a\} = C|_p$ by N_a,*

where positions and polarities are defined as in Fig. 6.2. Then RB′ *semantically emulates* RB.

Proof. It is easy to see that RB′ $\models$ RB due to the restriction on the polarity of the replaced nominals. Conversely, every model $\mathcal{I}$ of RB can be extended to a model $\mathcal{I}'$ of RB′ by setting $N_a^{\mathcal{I}'} := \{a\}^{\mathcal{I}}$ for all individual names $a \in \mathbf{I}$. □

This additional transformation significantly increases the class of DL Rules obtained for description logics without nominals, which will also be crucial to obtain a generalisation of DL-safe rules in Chapter 9.

Definition 8.4.2 Consider a DL $\mathcal{L}$ that is a fragment of $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$, and that supports concept conjunctions, existential restrictions, local reflexivity (Self), and (general) role inclusion axioms. Given a set of SWRL rules RB, let RB′ denote the corresponding set of rules in normal form and with nominals eliminated as in Lemma 8.4.1. Then RB is an $\mathcal{L}$ *rule base* if:

(1) RB is a $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ rule base,

(2) all concept and role expressions in RB′ are allowed in $\mathcal{L}$,

(3) if $\mathcal{L}$ contains only regular knowledge bases, then RB is regular in the sense of Definition 8.2.4.

$\mathcal{L}$ *rules* are SWRL rules that occur in some $\mathcal{L}$ rule base. ◇

This rather compact definition deserves some explanation. We restrict to DLs that feature at least the basic operators that were used to emulate DL Rules since only very restricted rule languages can be obtained without them. Condition (1) ensures that we can apply the construction of Section 8.2 and 8.3 to obtain a $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ knowledge base that semantically emulates the given $\mathcal{L}$ rule base RB. Let KB(RB) denote the according knowledge base that is obtained by applying the transformations steps of Fig. 8.3 to the pre-transformed rule base RB′ as in Definition 8.4.2. We thus incorporate the additional transformation of Lemma 8.4.1, so that conditions (2) and (3) suffice to establish the following result.

Proposition 8.4.3 *Consider an $\mathcal{L}$ rule base* RB *for some description logic $\mathcal{L}$ as in Definition 8.4.2. Then the $\mathcal{SROIQ}(B_s, \times)^{free}$ knowledge base* KB(RB) *is an $\mathcal{L}$ knowledge base that semantically emulates* RB.

The complexity of checking satisfiability of $\mathcal{L}$ rule bases is the same as the complexity of checking satisfiability of $\mathcal{L}$ knowledge bases.

Proof. The claimed semantic emulation is an immediate consequence of the according results for $\mathcal{SROIQ}^{\text{free}}$ rules and $\mathcal{SROIQ}(B_s, \times)^{\text{free}}$ rules, together with the fact that semantic emulation is preserved by Lemma 8.4.1. It is easy to see that KB(RB) contains only role and concept expressions that are allowed in $\mathcal{L}$. Indeed, item (2) of Definition 8.4.2 ensures that the pre-transformed rule base RB′ contains only such expressions, and the translation algorithm of Fig. 8.3 introduces only constructs that were supposed to be available in $\mathcal{L}$.

If $\mathcal{L}$ is a fragment of $\mathcal{SROIQ}(B_s, \times)$, i.e. if it imposes regularity restrictions on RBoxes, then these conditions are also satisfied by KB(RB) due to item (3) of Definition 8.4.2.

For the claimed complexity result, note that checking satisfiability of $\mathcal{L}$ knowledge bases must be P-hard, since $\mathcal{L}$ supports conjunctions of concepts. Satisfiability checking of propositional Horn logic is a well-known P-complete problem for which there is an obvious LogSpace reduction to the satisfiability problem of $\mathcal{L}$ knowledge bases. Now for inclusion, it suffices to note that the construction of KB(RB) is also possible in LogSpace. For hardness, we observe that the standard transformation of $\mathcal{L}$ knowledge bases to semantically equivalent $\mathcal{L}$ rule bases (see Section 4.2.1) is again possible in LogSpace. □

This result confirms that Definition 8.4.2 provides a suitable generic definition of DL Rule languages. A more careful inspection of this definition is useful to understand its implications. While conditions (1) and (3) should be obvious, the effects of (2) are slightly more complex, since it refers to the result of rule normalisation. Basic characterisations of DL Rules as in Definition 8.2.3 require the rule body to be tree-shaped, and the normalisation in Fig. 8.2 attempts to create this form by using $\mathcal{SROIQ}(B_s, \times)$ constructs. When using a weaker DL, some of these constructs might not be available, so that the according normalisation rule is not allowed. In other words, it is generally allowed to use DL rules that already have tree-shaped bodies, while deviations from that form are only admissible if the DL is sufficiently expressive.

As an example, consider the description logic $\mathcal{SROEL}(\sqcap_s, \times)$ as defined in Section 5.4. Theorem 5.4.7 showed that standard reasoning tasks for this logic are P-complete when restricting to admissible knowledge bases. Since admissibility is only concerned with the use of concept products on the right-hand side of concept inclusions, it does not restrict the use of concept products in rule bodies as encountered in Fig. 8.4. Other uses of concept products cannot occur since our definition of SWRL based on $\mathcal{SROIQ}$ does not include them. Therefore, we immediately obtain following corollary of Proposition 8.4.3 and Theorem 5.4.7.

Corollary 8.4.4 *The problem of deciding satisfiability of a* $\mathcal{SROEL}(\sqcap_s, \times)$ *rule base is* P*-complete w.r.t the size of the rule bases.*

$\mathcal{SROEL}(\sqcap_s, \times)$ does not feature inverse roles – it is known that this would increase its reasoning complexity to ExpTime [BBL08] – such that step (5) of the normalisation in Fig. 8.2 is not applicable. In effect, bodies of $\mathcal{SROEL}(\sqcap_s, \times)$ rules need to be a conjunction of tree-shaped bodies that do not share variables, and the root of one of these components must be the root of the rule's head. On the other hand, the use of concept products and role conjunctions in the additional transformations of Fig. 8.4 effectively relaxes the restrictions imposed on simple roles. Namely, a simple role name R can occur in a rule head $R(x, y)$ as long as the rule's body contains only role atoms of the form $S(x, y)$ where S is simple, or of the form $T(z, a)$ where $a \in \mathbf{I}$.

Definition 8.4.2 could still be generalised further. In particular, it currently is tailored toward DLs that generally allow or disallow concept expressions in GCIs. A notable class of logics for which this is not the case are Horn description logics as considered in Chapter 6, where different restrictions apply to concepts depending on whether they occur as premises or as conclusions. It is easy to see that Definition 8.4.2 could be generalised to cover this type of DLs by being more specific about the type of concept expression that is allowed in rule heads and bodies.

In addition, existential role restrictions are in fact only required for rolling up concept expressions in rule bodies, such that even $\mathcal{DLP}$ provides sufficient expressivity for defining a class of DL Rules. The resulting formalism of $\mathcal{DLP}$ rules has the interesting property that its rule bases can be semantically emulated by datalog programs. This provides us with a way of using rule-based inference engines for evaluating a certain kind of DL Rules. The following section illustrates that this can be a viable approach for other DLs as well.

8.5 Implementing DL Rules in Datalog

In order to obtain decidability and complexity results in the previous sections, we took the approach of reducing DL Rules to knowledge bases of the underlying description logics, thus enabling inference engines for description logics to be used when reasoning with DL Rules. Conversely, the proximity of DL Rules to first-order rule languages suggests to ask for similar translations that allow inference problems to be expressed in a rule language. The above discussion of $\mathcal{DLP}$ rules indicated that this is possible in some cases. Establishing this result for $\mathcal{DLP}$ was straightforward due to the strong semantic relationships that exist between $\mathcal{DLP}$ and datalog. But from Chapter 7 we also know that $\mathcal{DLP}$ already is the maximal – in the sense of said chapter – fragment of $\mathcal{SROIQ}$ with such close connections to datalog.

Given that we cannot expect other DLs to have such close connections to datalog, we must be content with weaker semantic relationships. Fortunately, even equisatisfiability suffices to translate standard reasoning problems, but this generalisation also opens a significantly larger field for possible solutions. Indeed, a number of translations to (disjunctive) datalog have been proposed to address reasoning tasks for description logics, see Section 8.7. It is not immediately clear how to adapt these approaches to DL Rules since general role inclusion axioms are typically not covered by the approaches.

In addition, one motivation for expressing inferencing problems for DL Rules in datalog would be that datalog can accommodate rule-like axioms in a more natural way, without requiring complicated rewritings. Horn DLs in general can be expected to allow for a more direct translation to datalog, but the example of Horn-$\mathcal{SHIQ}$ illustrates that the required algorithms can still be rather complex. In this section, we show that there is a significantly simpler translation for $\mathcal{SROEL}(\sqcap_s, \times)$ rules, for which reasoning is possible in polynomial time. In contrast to the approach that has been sketched for $\mathcal{DLP}$ rules above, our translation directly converts $\mathcal{SROEL}(\sqcap_s, \times)$ into an equisatisfiable datalog program that mirrors the basic structure of the input rules. As explained above, it cannot be expected that the resulting datalog semantically emulates the original rule base, but it turns out that important entailments are still preserved.

We do not consider an extended definition of SWRL here that would allow the use of $\mathcal{SROEL}(\sqcap_s, \times)$ role constructors in $\mathcal{SROEL}(\sqcap_s, \times)$ rules. Hence, additional role constructors do not occur in the input rules. However, we want to obtain the datalog translation for $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases that was given in Section 5.4 as a special case, so we need to ensure that (rule versions) of all $\mathcal{SROEL}(\sqcap_s, \times)$ axioms in normal form are covered (see Definition 5.4.2). For most normal forms, a semantically equivalent $\mathcal{SROEL}(\sqcap_s, \times)$ rule is obvious, but axioms of the form $R \sqsubseteq C \times D$ cannot be represented. Such axioms are clearly equivalent to two axioms $R \sqsubseteq C \times \top$ and $R \sqsubseteq \top \times D$, where the former can be represented as a $\mathcal{SROEL}(\sqcap_s, \times)$ rule $R(x, y) \to C(x)$. To cover the latter axiom as well, we allow additional *range restriction* rules of the form $R(x, y) \to D(y)$.

As noted in Section 5.4, concept products on the right-hand side of RIAs must be restricted in order to retain tractability. The above decomposition of such axioms shows that the problem is due to range restrictions only, since axioms $R \sqsubseteq C \times \top$ can always be represented as GCIs $\exists R.\top \sqsubseteq C$. Similar to the structural restriction that were defined for concept products in Definition 5.4.4, we thus can define admissibility for range restrictions.

Definition 8.5.1 Consider a $\mathcal{SROEL}(\sqcap_s, \times)$ rule base RB and a set of range restriction rules RR. For every role name R, define $\mathsf{ran}(R) := \{D \mid R(x, y) \to D(y) \in \text{RR}\}$. The range restrictions RR are *admissible* for RB if, for every rule $B \to R(t, y)$ with $t \in \mathbf{V} \cup \mathbf{I}$ and $y \in \mathbf{V}$, and for every $D \in \mathsf{ran}(R)$, one of the following holds:

- $D(y) \in B$, or
- there is some atom $S(s, y) \in B$ with $s \in \mathbf{V} \cup \mathbf{I}$ and $D \in \mathsf{ran}(S)$.

An *extended* $\mathcal{SROEL}(\sqcap_s, \times)$ rule base is the union of a $\mathcal{SROEL}(\sqcap_s, \times)$ rule base RB with a set of range restrictions RR that are admissible for RB. $\diamond$

It should be noted that this definition is compatible with the normalisation rules for DL Rules in the sense that a set of range restrictions is admissible for the normalisation of a rule base if and only if it is admissible for the rule base. Returning to the example from Fig. 8.1, we find that rule (3) is a range restriction. Rules (1), (2), and (5) can occur within a $\mathcal{SROEL}(\sqcap_s, \times)$ rule base, while rule (4) is disallowed since its normal form includes inverse roles. Given these rules, it is easy to see that rule (3) is an admissible range restriction, so the rules (1), (2), (3), and (5) together with all facts of Fig. 8.1 form an extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule base. As in the case of $\mathcal{SROIQ}(B_s, \times)$ rules, we can only conclude `Unhappy(bijan)`, but recognising the given rules as an extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule base allows us to apply polynomial-time algorithms for inferencing.

Definition 8.5.1 is slightly more restrictive than Definition 5.4.4 since it considers only the explicitly asserted ranges of each role, while the earlier definition used the hierarchy of simple roles to derive "obvious" implied restrictions. It is clear that such an extension would be possible in the above case as well, but since rules can generally have more different forms than RIAs, the formulation would not be as natural as for the case of $\mathcal{SROEL}(\sqcap_s, \times)$. Even without this, every extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule base is transformed to an admissible $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base when using the algorithm of Fig. 8.4 together with the translation for range restriction rules by means of concept products as discussed above. Conversely, every admissible $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base is semantically emulated by an extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule base that is obtained by the obvious translation of axioms, together with additional range restriction rules to explicitly state the implicit range restrictions as considered in Definition 5.4.4.

To simplify our presentation, we first transform $\mathcal{SROEL}(\sqcap_s, \times)$ rules into a simpler form.[3] In contrast to the approach taken to represent $\mathcal{SROIQ}$ rules as knowledge bases, we now perform an inverted rolling-up to decompose concept expressions to individual rule atoms.

Proposition 8.5.2 *Every* $\mathcal{SROEL}(\sqcap_s, \times)$ *rule base* RB *is semantically emulated by a* $\mathcal{SROEL}(\sqcap_s, \times)$ *rule base* RB′ *such that the following holds for every rule* $B \to H \in \mathrm{RB}'$*:*

– *all variables in* H *occur in* B*,*

– *if* $C(t) \in B$ *then* $C = A$*,* $C = \top$*, or* $C = \{a\}$ *where* $A \in \mathbf{A}$ *and* $a \in \mathbf{I}$*,*

– *if* $C(t) \in H$ *then* $C = A$*,* $C = \exists R.A$*,* $C = \bot$*, or* $C = \{a\}$ *where* $A \in \mathbf{A}$ *and* $a \in \mathbf{I}$.

Moreover, RB′ *can be computed in linear time w.r.t. the size of* RB.

[3]We avoid the term "normal form" here since it was already introduced with another meaning for DL Rules above.

Proof. The transformation algorithm iteratively transforms RB. In each iteration, a rule $B \to H$ that is not in the required form is selected. If $H = \top(t)$, then delete $B \to H$ from RB. If $H = (C \sqcap D)(t)$, then replace it with new rules $B \to X(t)$, $X(t) \to C(t)$, and $X(t) \to D(t)$, where X is a fresh concept name. If $H = \exists R.C(t)$ with $C \notin \mathbf{A}$, then replace it with rules $B \to \exists R.X(t)$ and $X(x) \to C(x)$ where X is again a fresh concept name. If $H = \exists R.\mathsf{Self}(t)$ then replace it with $R(t,t)$.

If B contains an atom $\bot(t)$, then delete $B \to H$ from RB. If B contains an atom $\exists R.C(t)$, then replace it with $R(t,y) \wedge C(y)$ where $y \in \mathbf{V}$ does not occur in $B \to H$ yet. If B contains an atom $(C \sqcap D)(t)$, then replace it with $C(t), D(t)$. If B contains an atom $\exists R.\mathsf{Self}(t)$ then replace it with $R(t,t)$.

Finally, if H contains a variable x that does not occur in B, then add $\top(x)$ to B. It is easy to see that this construction leads to the required result after a linear number of steps. □

The construction in the previous proof can be assumed to be deterministic if the order of the transformation steps is fixed. Note that range restriction rules already satisfy the requirements of Proposition 8.5.2. $\mathcal{SROEL}(\sqcap_s, \times)$ rules can be transformed to datalog as follows:

Definition 8.5.3 Given an extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule base RB, the datalog program $\mathsf{P}(\text{RB})$ is defined as follows. The following new symbols are introduced:

- concept names Self_R for each simple role name $R \in \mathbf{N}_s$,
- individual names $d_{R,A}$ for each $R \in \mathbf{N}$ and $A \in \mathbf{A}$.

In the following, we will always use $\mathbf{I}$, $\mathbf{A}$, $\mathbf{N}$, $\mathbf{N}_n$, $\mathbf{N}_s$ to refer to the original signature of RB, not including the additional symbols added above. Let RB′ denote the simplified $\mathcal{SROEL}(\sqcap_s, \times)$ rule base obtained from RB as in Proposition 8.5.2. The program $\mathsf{P}(\text{RB})$ is obtained from RB′ as follows:

(a) For all rules $B \to H \in \text{RB}'$, the program $\mathsf{P}(\text{RB})$ contains the rule $B' \to H'$ that is obtained from $B \to H$ by replacing all occurrences of $R(x,x)$ by $\mathsf{Self}_R(x)$, all occurrences of $\{a\}(t)$ by $a \approx t$, and all occurrences of $\exists R.A(t)$ with $A \in \mathbf{A}$ by the conjunction $R(t, d_{R,A}) \wedge A(d_{R,A})$.[4]

(b) For all rules $B \to S(y,z) \in \text{RB}'$ with $y, z \in \mathbf{V}$ and $S \in \mathbf{N}_s$ simple, $\mathsf{P}(\text{RB})$ contains the rule $B' \to \mathsf{Self}_S(y)$ where B' is obtained from B by replacing z by y, and – afterwards – replacing all occurrences of $R(x,x)$ by $\mathsf{Self}_R(x)$, and all occurrences of $\{a\}(t)$ by $a \approx t$.

[4]Note that this substitution can only occur in rule heads. As usual, conjunctions in rule heads serve as a shortcut notation for two rules with the same body and either of the conjuncts as their head.

(1) $\mathtt{Vegetarian}(x) \wedge \mathtt{FishProduct}(y) \to \mathtt{dislikes}(x, y)$
$\mathtt{Vegetarian}(x) \wedge \mathtt{FishProduct}(x) \to \mathsf{Self}_{\mathtt{dislikes}}(x)$
(2) $\mathtt{orderedDish}(x, y) \wedge \mathtt{dislikes}(x, y) \to \mathtt{Unhappy}(x)$
(3) $\mathtt{orderedDish}(x, y) \to \mathtt{Dish}(y)$
(5) $\mathtt{Happy}(x) \wedge \mathtt{Unhappy}(x) \to \bot$

$\mathtt{Vegetarian}(\mathtt{anja})$	$\mathtt{orderedDish}(\mathtt{anja}, \mathtt{thaiRedCurry})$
	$\mathtt{contains}(\mathtt{thaiRedCurry}, d_{\mathtt{contains},\mathtt{FishProduct}})$
	$\mathtt{FishProduct}(d_{\mathtt{contains},\mathtt{FishProduct}})$
$\mathtt{Vegetarian}(\mathtt{bijan})$	$\mathtt{orderedDish}(\mathtt{bijan}, \mathtt{fishFingers})$
	$\mathtt{FishProduct}(\mathtt{fishFingers})$
$\mathtt{Vegetarian}(\mathtt{ian})$	$\mathtt{orderedDish}(\mathtt{ian}, d_{\mathtt{orderedDish},X1})$
	$X1(d_{\mathtt{orderedDish},X1})$
	$X1(x) \to \mathtt{contains}(x, d_{\mathtt{contains},\{\mathtt{fishSauce}\}})$
	$\mathtt{fishSauce} \approx d_{\mathtt{contains},\{\mathtt{fishSauce}\}}$
	$\mathtt{FishProduct}(\mathtt{fishSauce})$
$\mathtt{Vegetarian}(\mathtt{markus})$	$\mathtt{orderedDish}(\mathtt{markus}, d_{\mathtt{orderedDish},X2})$
	$X2(d_{\mathtt{orderedDish},X2})$
	$X2(x) \to \mathtt{contains}(x, d_{\mathtt{contains},\mathtt{FishProduct}})$

$X1$ and $X2$ are fresh concept names from the simplification of Proposition 8.5.2.

For each role $R \in \{\mathtt{contains}, \mathtt{dislikes}, \mathtt{orderedDish}\}$ and each individual $a \in \{\mathtt{anja}, \mathtt{bijan}, \mathtt{fishFingers}, \mathtt{fishSauce}, \mathtt{ian}, \mathtt{markus}, \mathtt{thaiRedCurry}\}$ a rule $R(a, a) \to \mathsf{Self}_R(a)$

Figure 8.5: Datalog program for the extended $\mathcal{SROEL}(\sqcap_s, \times)$ rules of Fig. 8.1

(c) For each $a \in \mathbf{I}$ and $R \in \mathbf{N}_s$ simple, $\mathsf{P}(\mathrm{RB})$ contains the rule $R(a, a) \to \mathsf{Self}_R(a)$.

In all cases, x denotes an arbitrary variable $x \in \mathbf{V}$, and t denotes an arbitrary term $t \in \mathbf{V} \cup \mathbf{I}$. ◇

It is easy to see that $\mathsf{P}(\mathrm{RB})$ is indeed a datalog program. Note that atoms of the form $\mathsf{Self}_R(x)$ are created only in cases where R must be simple: in (a) this is the case since only such occurrences of $R(x, x)$ are allowed in a $\mathcal{SROEL}(\sqcap_s, \times)$ rule, and in (b) it follows since S is simple so that R must also be simple for all atoms $R(y, z) \in B$.

As an example, Fig. 8.5 shows a datalog translation of the rules of Fig. 8.1 that were identified above to be allowed in an extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule base. An interesting point to observe is that the auxiliary individual $d_{\mathtt{contains},\mathtt{FishProduct}}$ is used both for the fish product in Anja's curry and for the fish product in Markus' unnamed dish. Clearly, the rule base does not entail that both fish products are the same, but the restrictions of $\mathcal{SROEL}(\sqcap_s, \times)$ rules ensure that it is impossible to query for that information. In other words, identifying both individuals does not lead to undesired conclusions.

The correctness proof for this construction constitutes an essential part of the technical contributions of this section, and we first provide some intuition on how the proof proceeds. To show that RB and $\mathsf{P}(\mathrm{RB})$ are equisatisfiable, we construct models of $\mathsf{P}(\mathrm{RB})$ from models of RB, and vice versa. It is well-known that, in the case of $\mathcal{EL}^{++}$, models can be generated by introducing only a single element for each atomic concept [BBL05]. For $\mathcal{SROEL}(\sqcap_s, \times)$ rules, however, the added features of role conjunction and local reflexivity change the situation: considering only one characteristic element per atomic concept leads to undesired entailments in both cases. Our model constructions therefore deviate from the classical $\mathcal{EL}^{++}$ construction that worked for the simple $\mathcal{EL}$ rules in [KRH08a] with only minor modifications.

For instance, the rule base $\{a\}(x) \to \exists R.C(x), \{a\}(x) \to \exists S.C(x)$ does not entail any conjunction of the form $R(a, x) \wedge S(a, x)$. Yet, every interpretation in which the extension of C is a singleton set would necessarily entail this conjunction. This motivates the above use of $d_{R,C}$ in $\mathsf{P}(\mathrm{RB})$, which, intuitively, represent elements of C that have been "generated" by a rule head of the form $\exists R.C(x)$. Thus we admit $|\mathbf{N}|$ distinct characteristic individuals for each concept, and this suffices for the proper model construction in the presence of role conjunctions.

The second problematic feature are expressions of the form $R(x, x)$, which again preclude the consideration of only one characteristic individual per concept. The use of concept atoms $\mathsf{Self}_R(x)$ enables the translation of models for RB to models of $\mathsf{P}(\mathrm{RB})$ (the soundness of the satisfiability checking algorithm). The latter may indeed entail additional statements of type $R(x, x)$ without impairing the validity of the datalog rules that use $\mathsf{Self}_R(x)$.

In the other direction, models of RB are built from models of $\mathsf{P}(\mathrm{RB})$ by creating infinitely many "parallel copies" of a basic model structure. These copies form an infinite sequence of levels in the model, and simple roles relate only to successors in higher levels. Exceptions to this construction principle, such as the concept product rules discussed earlier, make the exact formalisation technically involved. The below proof for this case hinges upon the simplicity of roles in concepts Self_S, and it is not clear if a relaxation of this requirement would be possible.

Lemma 8.5.4 *If is an extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule base* RB *in the simplified form of Proposition 8.5.2, then* RB *is satisfiable if* $\mathsf{P}(\mathrm{RB})$ *is satisfiable.*

Proof. If $\mathsf{P}(\mathrm{RB})$ is satisfiable, then it has a least Herbrand model $\mathcal{J}$ since it is a datalog program [AHV94]. This notion is typically defined for datalog *without* equality only, so we take the perspective that the equality predicate $\approx$ is part of the signature of $\mathsf{P}(\mathrm{RB})$ and has been axiomatised as in Section 4.1.3. With this convention, the domain of $\mathcal{J}$ is exactly the Herbrand universe $\Delta^{\mathcal{J}} = \mathbf{I} \cup \{d_{R,A} \mid R \in \mathbf{N}, A \in \mathbf{A}\}$ (if equality was part of the logic, the domain would consist of the $\approx$ equivalence classes of the Herbrand universe; our approach avoids this notational burden).

To define an interpretation $\mathcal{I}$ of RB, we also consider $\mathcal{SROEL}(\sqcap_s, \times)$ rules as a fragment of first-order logic without equality. In other words, we consider $\approx$ as a signature symbol that is interpreted as a congruence relation, i.e. as an equivalence relation with the additional property that the elements of any of its equivalence classes cannot be distinguished by first-order formulae over the given signature. It is clear that the traditional perspective can be obtained by factorising $\mathcal{I}$ with $\approx$, but the expanded view simplifies our presentation.

Now define $\Delta^{\mathcal{I}} := \mathbf{I} \cup \{d_{R,A,n} \mid R \in \mathbf{N}, A \in \mathbf{A}, n \geq 0\}$ where we assume this to be a disjoint union. For each $\delta \in \Delta^{\mathcal{I}}$, the level $\nu(\delta)$ is defined as $\nu(a) := 0$ if $a \in \mathbf{I}$, and $\nu(d_{R,A,n}) := n$. The projection $\iota : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ is defined by $\iota(a) := a$ for $a \in \mathbf{I}$, and $\iota(d_{R,A,n}) := d_{R,A}$. For each $a \in \mathbf{I}$, set $a^{\mathcal{I}} := a$. For any $A \in \mathbf{A}$, set $A^{\mathcal{I}} := \{\delta \in \Delta^{\mathcal{I}} \mid \iota(\delta) \in A^{\mathcal{J}}\}$. Finally, for each role name $R \in \mathbf{N}$, set $\langle \delta, \delta' \rangle \in R^{\mathcal{I}}$ iff $\langle \iota(\delta), \iota(\delta') \rangle \in R^{\mathcal{J}}$ and one of the following conditions holds:

- $\iota(\delta) \neq \iota(\delta')$, or
- $\iota(\delta) = \iota(\delta')$ and $\iota(\delta) \in \mathsf{Self}_R^{\mathcal{J}}$, or
- $\nu(\delta) < \nu(\delta')$.

Finally, $\approx$ is interpreted by setting $\approx^{\mathcal{I}} := \{\langle \delta, \delta' \rangle \mid$ there is $a \in \mathbf{I}$ such that $\langle \iota(\delta), a \rangle \in \approx^{\mathcal{J}}$ and $\langle \iota(\delta'), a \rangle \in \approx^{\mathcal{J}}\} \cup \{\langle \delta, \delta \rangle \mid \delta \in \Delta^{\mathcal{I}}\}$, where it is easy to check that this is indeed a congruence relation for $\mathcal{I}$.

We claim that $\mathcal{I}$ is a model of RB. Given a variable assignment $\mathcal{Z}$ for $\mathcal{I}$, let $\mathcal{Z}'$ denote the variable assignment for $\mathcal{J}$ defined as $\mathcal{Z}'(x) := \iota(\mathcal{Z}(x))$. Then, for any atom α of the form $C(t)$ or $R(t, s)$ over the signature of RB, we find that $\mathcal{I}, \mathcal{Z} \models \alpha$ implies $\mathcal{J}, \mathcal{Z}' \models \alpha$. Moreover, $\mathcal{I}, \mathcal{Z} \models a \approx t$ implies $\mathcal{J}, \mathcal{Z}' \models a \approx t$ and thus $\mathcal{J}, \mathcal{Z}' \models \{a\}(t)$. Finally, $\mathcal{I}, \mathcal{Z} \models R(t, t)$ implies $\mathcal{J}, \mathcal{Z}' \models \mathsf{Self}_R(t)$.

Now consider any rule $B \to H$ such that $\mathcal{I}, \mathcal{Z} \models B$. By the previous observations, $\mathcal{J}, \mathcal{Z}' \models B'$, where $B' \to H'$ is the rule obtained from $B \to H$ in (a) of Definition 8.5.3. Since $\mathcal{J} \models \mathsf{P}(\mathrm{RB})$, we obtain $\mathcal{J}, \mathcal{Z}' \models H'$. We need to show that $\mathcal{I}, \mathcal{Z} \models H$. This follows directly from the definition of $\mathcal{I}$ for atoms of the form $C(t) \in H$ and $\{a\}(t) \in H$.

For atoms $\exists R.B(t) \in H$, we find that $\langle t^{\mathcal{I},\mathcal{Z}}, d_{R,B,n} \rangle \in R^{\mathcal{I}}$ for any $n > \nu(t^{\mathcal{I},\mathcal{Z}})$, since $\mathcal{J}, \mathcal{Z}' \models R(t, d_{R,B}) \wedge B(d_{R,B})$. This also shows that $d_{R,B,n} \in B^{\mathcal{I}}$, so we can conclude $\mathcal{I}, \mathcal{Z} \models \exists R.B(t)$ as required.

For atoms $R(t, s) \in H$, we need to verify that one of the conditions in the definition of $R^{\mathcal{I}}$ is satisfied. Thus assume that $\iota(t^{\mathcal{I},\mathcal{Z}}) = \iota(s^{\mathcal{I},\mathcal{Z}})$, and $\nu(t^{\mathcal{I},\mathcal{Z}}) \geq \nu(s^{\mathcal{I},\mathcal{Z}})$. If $t \in \mathbf{I}$ or $s \in \mathbf{I}$, then $\iota(t^{\mathcal{I},\mathcal{Z}}) = \iota(s^{\mathcal{I},\mathcal{Z}})$ implies $s = t$. We thus obtain $\mathcal{I}, \mathcal{Z} \models R(t, s)$ since $\iota(t^{\mathcal{I},\mathcal{Z}}) \in \mathsf{Self}_R^{\mathcal{J}}$, which in turn is a consequence of the fact that $\mathcal{J}$ satisfies the rules (c) of Definition 8.5.3. If $t, s \notin \mathbf{I}$, then $\mathsf{P}(\mathrm{RB})$ contains a rule $B'' \to \mathsf{Self}_R(y)$ by item (b) of Definition 8.5.3, and we can draw a similar conclusion by observing that $\mathcal{J}, \mathcal{Z}' \models B''$.

This shows that $\mathcal{I}$ satisfies all rules of RB, including range restrictions. □

The other direction of the proof is slightly more complex, since Herbrand models cannot be assumed to be available for $\mathcal{SROEL}(\sqcap_s, \times)$ rule bases. Instead of directly relating domain elements to elements of the original model, we now assign characteristic concepts $\kappa(\delta)$ to each domain element δ.

Lemma 8.5.5 *If* RB *is an extended* $\mathcal{SROEL}(\sqcap_s, \times)$ *rule base in the simplified form of Proposition 8.5.2, then* $\mathsf{P}(\mathrm{RB})$ *is satisfiable if* RB *is satisfiable.*

Proof. Assume that RB has some model $\mathcal{I}$. We define an interpretation $\mathcal{J}$ of $\mathsf{P}(\mathrm{RB})$ with domain $\Delta^{\mathcal{J}} := \{a^{\mathcal{I}} \mid a \in \mathbf{I}\} \cup \{d_{R,C} \mid R \in \mathbf{N}, C \in \mathbf{A}, (C \sqcap \exists R^-.\top)^{\mathcal{I}} \not\subseteq \{a\}^{\mathcal{I}}$ for all $a \in \mathbf{I}\}$, where we assume that this is a disjoint union. Note that we use inverse roles for describing semantic conditions here, although inverses cannot be used in $\mathcal{SROEL}(\sqcap_s, \times)$. For each individual name d in $\mathsf{P}(\mathrm{RB})$, set $d^{\mathcal{J}}$ as follows:

- If $d \in \mathbf{I}$, then $d^{\mathcal{J}} := d^{\mathcal{I}}$.
- If $d = d_{R,C} \in \Delta^{\mathcal{J}}$, then $d^{\mathcal{J}} := d$.
- If $d = d_{R,C} \notin \Delta^{\mathcal{J}}$ and $(C \sqcap \exists R^-.\top)^{\mathcal{I}} \subseteq \{a\}^{\mathcal{I}}$ for some $a \in \mathbf{I}$, then $d^{\mathcal{J}} := a^{\mathcal{I}}$ for some (arbitrary) such a.

Moreover, we assign a concept expression $\kappa(\delta)$ to any element $\delta \in \Delta^{\mathcal{J}}$ as follows:

- if $\delta = a^{\mathcal{I}}$ with $a \in \mathbf{I}$ then $\kappa(\delta) := \{a\}$ for some (arbitrary) such a,
- if $\delta = d_{R,C}$ then $\kappa(\delta) := C \sqcap \exists R^-.\top$.

Now $\mathcal{J}$ interprets roles and concepts as follows (where we assume that C and R are symbols occurring in RB):

(A) $\delta \in C^{\mathcal{J}}$ iff $\kappa(\delta)^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

(B) $\delta \in \mathsf{Self}_R^{\mathcal{J}}$ iff $\langle \epsilon, \epsilon \rangle \in R^{\mathcal{I}}$ for all $\epsilon \in \kappa(\delta)^{\mathcal{I}}$

(C) $\langle \delta, a^{\mathcal{I}} \rangle \in R^{\mathcal{J}}$ for $a \in \mathbf{I}$ iff $\kappa(\delta)^{\mathcal{I}} \subseteq \exists R.\{a\}^{\mathcal{I}}$

(D) $\langle \delta, d_{S,C} \rangle \in R^{\mathcal{J}}$ for $R \in \mathbf{N}_\mathrm{n}$ iff $\kappa(\delta)^{\mathcal{I}} \subseteq \exists R.\kappa(d_{S,C})^{\mathcal{I}}$, and $\kappa(d_{S,C})^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $D \in \mathsf{ran}(R)$

(E) $\langle \delta, d_{S,C}\rangle \in R^{\mathcal{J}}$ for $R \in \mathbf{N}_\mathrm{s}$ and $\kappa(\delta)^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$ iff $\langle \epsilon, \epsilon'\rangle \in R^{\mathcal{I}}$ for all $\epsilon \in \kappa(\delta)^{\mathcal{I}}$ and $\epsilon' \in \kappa(d_{S,C})^{\mathcal{I}}$ with $\langle \epsilon, \epsilon'\rangle \in S^{\mathcal{I}}$, and $\kappa(d_{S,C})^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $D \in \mathsf{ran}(R)$

(F) $\langle \delta, d_{S,C}\rangle \in R^{\mathcal{J}}$ for $R \in \mathbf{N}_\mathrm{s}$ and $\kappa(\delta)^{\mathcal{I}} \nsubseteq \exists S.C^{\mathcal{I}}$ iff $\langle \epsilon, \epsilon'\rangle \in R^{\mathcal{I}}$ for all $\epsilon \in \kappa(\delta)^{\mathcal{I}}$ and $\epsilon' \in \kappa(d_{S,C})^{\mathcal{I}}$, and $\kappa(d_{S,C})^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $D \in \mathsf{ran}(R)$

We claim that $\mathcal{J}$ is a model for P(RB). For the rules of type (c) in Definition 8.5.3 this is easy to see. Now consider some rule $B' \to H'$ generated from a rule $B \to H \in$ RB by item (a). Assume there is a variable assignment $\mathcal{Z}'$ for $\mathcal{J}$ such that $\mathcal{J}, \mathcal{Z}' \models B'$. We show how to iteratively construct a variable assignment $\mathcal{Z}$ for $\mathcal{I}$ such that $\mathcal{I}, \mathcal{Z} \models B$, where the construction starts at the root element of B:

While $\mathcal{Z}$ has not been defined for all variables occurring in B, do the following:

- Select a variable x occurring in B such that there is no atom $R(y, x) \in B$ with $y \in \mathbf{V}$ such that $y \neq x$ and $\mathcal{Z}(y)$ not defined yet. Note that such an x always exists, since $B \to H$ is a DL Rule, and thus has no proper cycles.
- Select a value $\mathcal{Z}(x) \in \kappa(\mathcal{Z}'(x))^{\mathcal{I}}$ as follows:
  1. If $\mathcal{Z}'(x) = a^{\mathcal{I}}$ with $a \in \mathbf{I}$ then set $\mathcal{Z}(x) := a^{\mathcal{I}}$.
  2. Otherwise, if there is some $R(t, x) \in B'$ with $R \in \mathbf{N}_\mathrm{n}$, then let $\mathcal{Z}(x)$ be some element $\epsilon \in \kappa(\mathcal{Z}'(x))^{\mathcal{I}}$ such that $\langle t^{\mathcal{I},\mathcal{Z}}, \epsilon\rangle \in R^{\mathcal{I}}$.
     For the remaining cases, assume that (1) and (2) do not hold, and hence $\mathcal{Z}'(x) = d_{S,C}$, and all role atoms in B' that contain x in the second position refer to simple roles.
  3. If there is some $R(t, x) \in B'$ such that $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$, then let $\mathcal{Z}(x)$ be some element $\epsilon \in \kappa(\mathcal{Z}'(x))^{\mathcal{I}}$ such that $\langle t^{\mathcal{I},\mathcal{Z}}, \epsilon\rangle \in S^{\mathcal{I}}$.
  4. Otherwise let $\mathcal{Z}(x)$ be some element $\epsilon \in \kappa(\mathcal{Z}'(x))^{\mathcal{I}}$.

Finally, for all variables x not occurring in B, let $\mathcal{Z}(x)$ be arbitrary.

We need to verify that $\mathcal{Z}$ is indeed well-defined. For that we must show that the choice of $\mathcal{Z}(x)$ in (1)–(4) above is always possible. To this end, note that $\kappa(\mathcal{Z}'(x))^{\mathcal{I}}$ is non-empty by definition of κ. We check all cases separately:

(1) The given choice clearly is possible, and $\mathcal{Z}(x) \in \kappa(\mathcal{Z}'(x))^{\mathcal{I}}$.

(2) Since $R(t, x) \in B'$ with R non-simple, this atom is the only role atom with x in its second component by definition of $\mathcal{SROEL}(\sqcap_\mathrm{s}, \times)$ rules, hence the choice of $R(t, x)$ is canonical. From $\mathcal{J}, \mathcal{Z}' \models B'$ and (D) in the definition of $\mathcal{J}$ we conclude that $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} \subseteq \exists R.\kappa(\mathcal{Z}'(x))^{\mathcal{I}}$. By definition of $\mathcal{Z}$ (for case $t \in \mathbf{V}$) and $\mathcal{J}$ (for case $t \in \mathbf{I}$), we find that $t^{\mathcal{I},\mathcal{Z}} \in \kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}}$, and thus there must be a possible choice for $\mathcal{Z}(x)$.

(3) In this case, the choice of $\mathcal{Z}(x)$ depends on the term t in the first position of the selected atom $R(t, x)$. However, by the definition of DL rules, all atoms of the form $R'(t', x)$ must have the same term in their first component, and thus the choice of t is again canonical. By assumption, we find $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$, and we can apply a similar argument as in case (2) to conclude that the required choice of $\mathcal{Z}(x)$ is possible.

(4) Trivial.

We further claim that $\mathcal{I}, \mathcal{Z} \models B$, which is shown by considering all atoms that may occur in B:

- $C(t)$ with $C \in \mathbf{A}$. By Definition 8.5.3, B' also contains $C(t)$ and hence $\mathcal{J}, \mathcal{Z}' \models C(t)$. If $t \in \mathbf{V}$ then, by construction of $\mathcal{Z}$, we find that $\mathcal{Z}(t) \in \kappa(\mathcal{Z}'(t))^{\mathcal{I}}$. Hence, by item (A) in the definition of $\mathcal{J}$, $\mathcal{Z}(t) \in C^{\mathcal{I}}$. Otherwise, if $t \in \mathbf{I}$ then we find that $\kappa(t)^{\mathcal{I}} = \{t\}^{\mathcal{I}} = \{t^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$ as required, where the subset inclusion follows again from (A).
- $\{a\}(t)$. In this case, $t^{\mathcal{J},\mathcal{Z}'} = a^{\mathcal{I}}$ and $\kappa(t^{\mathcal{J},\mathcal{Z}'}) = \{b\}$ for some b with $b^{\mathcal{I}} = a^{\mathcal{I}}$. Thus $t^{\mathcal{I},\mathcal{Z}} \in \kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} = \{a^{\mathcal{I}}\}$ as required.
- $R(t, u)$. First assume that $u \in \mathbf{V}$. If $t = u$, then $\mathsf{Self}_R(u) \in B'$ and we can use (B) to conclude $\mathcal{I}, \mathcal{Z} \models R(t, u)$. Otherwise, if $u \in \mathbf{V}$ and $t \neq u$, we can distinguish the cases as in the definition of $\mathcal{Z}$:

  (1) $\mathcal{I}, \mathcal{Z} \models R(t, a^{\mathcal{I}})$ is a direct consequence of (C).

  (2) The choice in case (2) of the definition of $\mathcal{Z}$ directly implies $\mathcal{I}, \mathcal{Z} \models R(t, u)$, where it is important to note that only one such (non-simple) role atom with second argument u can occur.

  (3) Again we have argued above that all role atoms with u in their second position must then be simple and refer to the same t in their first position. $\mathcal{Z}(u)$ was chosen such that $\langle t^{\mathcal{I},\mathcal{Z}}, \mathcal{Z}(u)\rangle \in S^{\mathcal{I}}$. Therefore, $\mathcal{J}, \mathcal{Z}' \models R(t, u)$ and (E) imply that $\langle t^{\mathcal{I},\mathcal{Z}}, \mathcal{Z}(u)\rangle \in R^{\mathcal{I}}$ as required.

  (4) Case (F) in the definition of $\mathcal{J}$ applies, and hence we again conclude that $\langle t^{\mathcal{I},\mathcal{Z}}, \mathcal{Z}(u)\rangle \in R^{\mathcal{I}}$.

  Finally, if $u \in \mathbf{I}$, then we can also apply the same reasoning as in case (1) above.

We thus find that $\mathcal{I}, \mathcal{Z} \models B$, and, since $\mathcal{I}$ is assumed to be a model of RB, we conclude that $\mathcal{I}, \mathcal{Z} \models H$. Moreover, for any variable x in B for which there is no atom $R(t, x) \in B$, and for any $\epsilon \in \kappa(\mathcal{Z}'(x))^{\mathcal{I}}$, we can construct such a variable assignment $\mathcal{Z}$ which additionally satisfies $\mathcal{Z}(x) = \epsilon$. This is easily seen since the value of $\mathcal{Z}$ is chosen by item (4) in the definition of $\mathcal{Z}$ in this case.

We can now show that $\mathcal{J}, \mathcal{Z}' \models H'$. First consider the case that $B \to H$ is a range restriction $R(x, y) \to C(y)$. If $\mathcal{Z}(y') = a^{\mathcal{I}}$ with $a \in \mathbf{I}$ then $\mathcal{Z}(y) = a^{\mathcal{I}}$ by the definition of $\mathcal{Z}$, and we find $a^{\mathcal{I}} \in C^{\mathcal{I}}$ since $\mathcal{I}$ satisfies $B \to H$. But then $\kappa(\mathcal{Z}(y))^{\mathcal{I}} = \{a\}^{\mathcal{I}} \sqsubseteq C^{\mathcal{I}}$, and hence $\mathcal{Z}'(y) \in C^{\mathcal{J}}$ by (A) as required. If $\mathcal{Z}'(y) \notin \mathbf{I}$, then $\mathcal{J}, \mathcal{Z}' \models R(x, y)$ must be due to (D), (E), or (F) in the definition of $\mathcal{J}$. In each case, $\kappa(\mathcal{Z}'(y))^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ is a necessary precondition, since $C \in \mathsf{ran}(R)$, and hence we obtain $\mathcal{J}, \mathcal{Z}' \models C(y)$ from (A).

Now assume that $B \to H$ is a $\mathcal{SROEL}(\sqcap_s, \times)$ rule that is no range restriction. We distinguish cases by considering the different types of atoms that may occur in H. According to Proposition 8.5.2, we have to consider three basic kinds of atoms: $A(t)$, $\exists R.B(t)$, and $R(t, u)$, where $A \in \mathbf{A} \cup \{\top\} \cup \{\{a\} \mid a \in \mathbf{I}\}$ and $B \in \mathbf{A}$. If $t \in \mathbf{V}$ then, by the definition of $\mathcal{SROEL}(\sqcap_s, \times)$ rules, we find that there is no atom $R(u, t) \in B$ with $u \neq t$. Thus, for any $\epsilon \in \kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}}$, there is an assignment $\mathcal{Z}$ such that $t^{\mathcal{I},\mathcal{Z}} = \epsilon$ and $\mathcal{I}, \mathcal{Z} \models H$. This also is trivially true if $t \notin \mathbf{V}$, since $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}}$ contains only a single element $t^{\mathcal{I}}$ in this case. Using this insight (†), we can consider the various possible kinds of atoms in H:

- If $A(t) \in H$ with $A \in \mathbf{A} \cup \{\bot\}$ then also $A(t) \in H'$. Then (†) shows that $\epsilon \in A^{\mathcal{I}}$ for all $\epsilon \in \kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}}$, and we can conclude that $t^{\mathcal{J},\mathcal{Z}'} \in A^{\mathcal{J}}$ by case (A) in the definition of $\mathcal{J}$. For $A = \bot$ this is a contradiction, showing that this case cannot occur.
- If $\{a\}(t) \in H$ then $a \approx t \in H'$. By (†), we find that $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} = \{a\}^{\mathcal{I}}$ and thus $t^{\mathcal{J},\mathcal{Z}'} \in \{a\}^{\mathcal{J}}$ by (A). But this implies $t^{\mathcal{J},\mathcal{Z}'} = a^{\mathcal{J}}$ as required.
- If $\exists R.B(t) \in H$ with $B \in \mathbf{A}$ then $R(t, d_{R,B}) \wedge B(d_{R,B}) \in H'$. By considering the possible values of κ, it is easy to see that $\kappa(d_{R,B}^{\mathcal{J}})^{\mathcal{I}} \subseteq B^{\mathcal{I}}$, which establishes the second part of the above conjunction by (A).

  To show that $R(t, d_{R,B})$ is also entailed, we again apply (†) as in the previous item to conclude that $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} \subseteq \exists R.B^{\mathcal{I}}$. Thus we just need to observe that the conditions for $\mathcal{J}, \mathcal{Z}' \models R(t, d_{R,B})$ that are given in (E) and (F), respectively, are satisfied.
- If $R(t, u) \in H$ with $t \neq u$ then $R(t, u) \in H'$. Using (†) and the fact that $u^{\mathcal{I},\mathcal{Z}} \in \kappa(u^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}}$, we find that $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} \subseteq \exists R.\kappa(u^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}}$. This establishes the required conditions for (D) and thus settles all cases where either $u \in \mathbf{I}$, or $u \in \mathbf{V}$ with $\mathcal{Z}'(u) = a^{\mathcal{I}}$ and $a \in \mathbf{I}$. For the remaining cases, assume that $u \in \mathbf{V}$ with $\mathcal{Z}'(u) = d_{S,C}$.

  Observe that, for all $D \in \mathsf{ran}(R)$, we find $d_{S,C}^{\mathcal{J}} \in D^{\mathcal{J}}$ since the conditions of Definition 8.5.1 hold, and since all range restrictions of RB are satisfied by $\mathcal{J}$ as shown above. By (A), this ensures that the conditions on $\mathsf{ran}(R)$ as stated in (D)–(F) hold. In particular, this settles the case $R \in \mathbf{N}_n$ by (D).

It remains to check the case where $R \in \mathbf{N}_s$. By the restrictions on simple roles in $\mathcal{SROEL}(\sqcap_s, \times)$ rules, we conclude that u occurs in the second position of role atoms in B' only if the atom is of the form $R'(t, u)$ with R' simple. If there is such an atom $R'(t, u) \in B'$ and if $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} \subseteq \exists S.C^{\mathcal{I}}$, then the value for $\mathcal{Z}(u)$ was chosen by case (3) of the definition of $\mathcal{Z}$. We can thus derive a similar statement as (†), and conclude that $\mathcal{Z}(u)$ might take any value $\epsilon' \in \kappa(\mathcal{Z}'(u))^{\mathcal{I}}$ for which $\langle t^{\mathcal{I},\mathcal{Z}}, \epsilon' \rangle \in S^{\mathcal{I}}$. Since we derive $\langle t^{\mathcal{I},\mathcal{Z}}, \epsilon' \rangle \in R^{\mathcal{I}}$ in all these cases, we can invoke (E) to conclude $\mathcal{J}, \mathcal{Z}' \models R(t, u)$.

If there is no role atom $R'(t, u)$ in B', or if $\kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}} \not\subseteq \exists S.C^{\mathcal{I}}$ for all such atoms, then $\mathcal{Z}(u)$ is chosen in case (4) of the definition of $\mathcal{Z}$. A similar argument as before shows that the conditions of case (F) are satisfied in this case, and we obtain $\mathcal{J}, \mathcal{Z}' \models R(t, u)$ as required.

– If $R(t, t) \in H$ then $\mathsf{Self}_R(t) \in H'$. Applying (†) again, we find that $\langle \epsilon, \epsilon \rangle \in R^{\mathcal{I}}$ for all $\epsilon \in \kappa(t^{\mathcal{J},\mathcal{Z}'})^{\mathcal{I}}$. Using (B), we can again derive $\mathcal{J}, \mathcal{Z}' \models \mathsf{Self}_R(t)$.

This shows that $\mathcal{J}, \mathcal{Z}' \models H'$ and concludes the proof for rules of type (a).

Finally, for rules generated in item (b) of Definition 8.5.3, note that one could similarly obtain these rules by item (a) by adding, for each rule $B \to H \in \mathrm{RB}$ with $R(x, y) \in H$ and $R \in \mathbf{N}_s$ simple, a rule $B' \to R(x, x)$, where B' is obtained from B by replacing all occurrences of y with x. Since adding such rules clearly does not affect the semantics of RB, case (b) is covered by case (a).

We conclude that $\mathcal{J}$ is indeed a model for all rules in $\mathsf{P}(\mathrm{RB})$ as required. □

Summing up the result of Proposition 8.5.2, Lemma 8.5.4, and Lemma 8.5.5, we obtain the following theorem:

Theorem 8.5.6 *Given an extended* $\mathcal{SROEL}(\sqcap_s, \times)$ *rule base* RB *in normal form,* RB *is unsatisfiable iff* $\mathsf{P}(\mathrm{RB})$ *is unsatisfiable.*

Definition 8.5.3 thus suggests an approach for implementing $\mathcal{SROEL}(\sqcap_s, \times)$ rules in datalog without the need of first transforming rule bases to $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases. This translation does not directly establish the tractability of reasoning problems that was stated in Corollary 8.4.4. The latter result was based on a tractability result for admissible $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases (Theorem 5.4.7) that was obtained by further decomposing axioms in the knowledge base so as to limit the number of variables that occur in each datalog rule after the translation. It is not hard to see that a similar result could be achieved by decomposing $\mathcal{SROEL}(\sqcap_s, \times)$ rules, and indeed we provide a more general result for an extension of $\mathcal{SROEL}(\sqcap_s, \times)$ rules in Section 9.4.

Such normalisations, however, are mainly relevant for obtaining worst-case complexity results, and it should not be taken for granted that they would actually improve the computational behaviour of inferencing engines. On the one hand, available datalog implementations are typically optimised for datalog rules with an arbitrary number of variables per rule, and the decomposition of such rules into many rules with a bounded number of variables would not necessarily lead to performance gains. On the other hand, a dedicated inference engine for extended $\mathcal{SROEL}(\sqcap_s, \times)$ rules may employ optimisations that exploit the tree structure of rules directly, without requiring an explicit decomposition that introduces new signature symbols.

8.6 Summary

In this chapter, we have introduced DL Rules as a novel class of decidable SWRL fragments. The main characteristic of DL rule bases is that they can be emulated by knowledge bases of an underlying description logic based on a transformation that can be performed in linear time. The expressiveness of DL Rule languages varies depending on the description logic on which they are based, and accordingly the worst-case complexity of satisfiability checking in DL rule languages agrees with the worst-case complexity of reasoning in this DL. We have specifically considered the highly expressive languages of $\mathcal{SROIQ}$ rules and $\mathcal{SROIQ}(B_s, \times)$ rules for which reasoning is N2ExpTime-complete, but also the class of extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule bases where polynomial-time reasoning is possible. In all of these cases, the most important defining feature of DL Rules is the tree-like dependency structure of their rule bodies.

When considering the impact of a DL on the resulting DL Rule language, we can distinguish expressive features that are only relevant for extending the available concept expressions from those that play a crucial rôle for emulating rules. The first kind of feature includes operators like concept union and cardinality restrictions. If features of this kind are not available, then the resulting DL Rule language simply does not comprise SWRL rules that include such concept expressions. The second kind of feature, in contrast, is required to capture the semantics of the basic logical constructs that rules provide, even if no complex role or concept expressions occur in its atoms. The most important of these features are concept conjunction, existential role restriction (on the left-hand side of GCIs), local reflexivity (Self), and general role inclusion axioms, all of which are necessary for emulating a reasonable amount of SWRL rules in DL.

Further features of the second kind are useful for encompassing a broader class of SWRL rules that are not exactly tree-shaped but that can be transformed into such a shape by applying obvious rewritings. These features are the universal role, inverse roles, role conjunctions, and concept products. Nominal classes, in contrast, have also been used to normalise the structure of rules but were shown to be dispensable in all cases in which they are introduced for this normalisation.

Reasoning in DL Rules is generally possible by transforming rule bases into DL knowledge bases. But this generic approach may lead to collections of axioms which disguise the original rule structure that could otherwise be useful to guide the search in inference engines. An alternative approach is to develop rule-based inferencing algorithms that can preserve the structure of rules while still supporting DL constructs beyond DLP (see Chapter 7). To this end, we have presented an algorithm for translating reasoning problems for extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule bases to datalog. The correctness proof of this method also establishes the correctness of the datalog transformation provided for $\mathcal{SROEL}(\sqcap_s, \times)$ in Section 5.4 which is obtained as a special case.

8.7 Related Work

DL Rules depend on expressive features that have been (re)introduced for DLs only with the proposal of $\mathcal{SRIQ}$ [HKS05] and $\mathcal{SROIQ}$ [HKS06]. Complex role inclusion axioms had originally been included even in KL-ONE – an early predecessor of today's description logics – where they were called *role-value maps* [BS85], but it had soon been recognised that these features lead to undecidability of basic inference problems [SS89]. Only much later have complex role inclusion axioms been introduced again into description logic research, at first only to confirm that undecidability occurs even with very restricted cases [Wes01]. Regularity conditions for retaining decidability were first proposed in [HS04], and more recent work suggested generalisations of these conditions [Kaz09b] that could also be relevant for enlarging the class of DL rule bases. For the case of $\mathcal{EL}$, it is well-known that no regularity conditions are required when introducing role inclusion axioms [BBL05].

It has long been known that DL concept expressions correspond to tree-shaped conjunctive formulae of first-order logic, and that GCIs thus correspond to certain SWRL rules. An extensive treatment of possible rolling-up approaches in the context of DL conjunctive query answering can be found in [Tes01]. Applying simple rolling-up methods to rules with unary head atoms has also led to the first proposals for decidable fragments of SWRL [PSG$^+$05]. The possibility of expressing larger classes SWRL rules by combining local reflexivity with general role inclusion axioms has first been introduced independently in [GSH08] and [KRH08a]. [GSH08] focusses on DL Rules for $\mathcal{SROIQ}$ and discusses slightly different rewriting method that takes "obvious" inferences into account for simplifying rule bodies (not all cases covered in [Tes01] are included, e.g. one could simplify role conjunctions of roles with a common functional superrole). This allows the approach to subsume more rule bases, but it also introduces another non-local criterion for determining whether a rule is supported or not. The related work [GH08] introduced a prototypical user interface to support the modelling of such rules. [KRH08a] includes tractability results for DL rule languages based on $\mathcal{EL}^{++}$ and DLP, all of which are subsumed by the more general results in this chapter. More recently, it has been proposed to introduce qualified role inclusion axioms as additional logical operators that are directly processed in inference algorithms [TSS09]. It is not hard to see that the approach of this chapter could exploit such constructs to emulate DL Rules in a more direct way.

The reduction of inference tasks of description logics to suitable inference tasks in datalog has been considered in a number of independent works. Examples include resolution-based approaches for $\mathcal{EL}$ [Kaz06] and $\mathcal{SHIQ}$ [HMS05, Mot06], as well as approaches for $\mathcal{SHIQ}$ based on ordered binary decision diagrams [RKH08d, RKH08c]. In many of these cases, disjunctive datalog – the extension of datalog with disjunction in rule heads – is required [Mot06, RKH08d, RKH08c]. Notable exceptions occur when considering Horn description logics such as Horn-$\mathcal{SHIQ}$ [HMS05] and $\mathcal{EL}$ [Kaz06], as discussed in Chapter 6. However, not all approaches lead to non-disjunctive datalog when applied to Horn DLs, as illustrated by the reduction in [RKH08d, RKH08c] that requires disjunctions to encode binary decision diagrams.

Chapter 9

Extending DL Rules with DL-Safe Variables

In this chapter, we extend the class of *DL-safe* SWRL rules which are based on the idea of limiting the interaction between datalog and description logics to a "safe" amount that does not endanger decidability.[1] DL-safe datalog rules have originally been introduced in [MSS05], where it was also shown that they do not increase the worst-case complexity of the DL $\mathcal{SHIQ}$.

We generalise this approach to the extended class of *DL+safe rules* that combine DL-safe rules with DL Rules as discussed in Chapter 8. Although DL+safe rules can still be expressed in terms of the underlying description logic, the according rewriting might incur an exponential growth of the size of the knowledge base. This contrasts the linear transformation that was obtained for DL Rules, and thus allows us to argue that DL+safe rules provide a real extension of expressiveness.

When considering $\mathcal{SROIQ}$ as the underlying DL, it turns out that this extension does not lead to an increased worst-case complexity of reasoning tasks. Given the very high worst-case complexity of $\mathcal{SROIQ}$, this does not allow us to conclude that the implementation of $\mathcal{SROIQ}$ rules with DL-safe variables is practically feasible. Indeed, our proof method leads to an exponential blow-up of the size of the input theory that would be prohibitive in practice. These observations motivate our definition of ELP as the most expressive tractable SWRL fragment that is considered within this work.

The structure of this chapter is as follows. Section 9.1 starts by providing a general introduction to DL-safe rules that provides the basic intuitions and motivations for the subsequent considerations. Section 9.2 introduces DL+safe rules as

[1]The name "DL-safe" actually originates from a related notion of "safety" that has been considered for datalog in the field of deductive databases.

an extension of DL-safe rules that exploit the insights of Chapter 8 to encompass additional SWRL rule bases. In Section 9.3, it is shown that satisfiability checking in $\mathcal{SROIQ}$+safe rules is N2ExpTime-complete, and thus not harder than reasoning in $\mathcal{SROIQ}$. Section 9.4 introduces ELP as a more light-weight rule language that can be processed by extending the datalog transformation from Section 8.5. We conclude by summarising our results in Section 9.5 and provide pointers to related work in Section 9.6.

The results of Section 9.4 can also be found in [KRH08b] though this does not encompass the full generality of DL+safe rules yet.

9.1 Introducing DL-Safe Rules

The restrictions that DL-safe rules impose on SWRL to preserve decidability can be viewed from two perspectives. On the one hand, one can give syntactic "safety" conditions that ensure the desired behaviour. This corresponds to the original definition of DL-safe rules. On the other hand, one can modify the semantics of SWRL rules so as to ensure that every rule is implicitly restricted to allow only "safe" interactions with description logic knowledge bases. This approach has become very common in practice, since it is indeed always possible to evaluate arbitrary SWRL rules in a DL-safe way, without requiring the user to adhere to specific syntactic restrictions. We begin with the original definition and explain the second perspective afterwards.

Definition 9.1.1 Consider a signature $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$ of SWRL as in Definition 4.2.1, with designated subsets of DL concept names $\mathbf{A} \subseteq \mathbf{P}$, simple role names $\mathbf{N}_\mathrm{s} \subseteq \mathbf{P}$, and non-simple role names $\mathbf{N}_\mathrm{n} \subseteq \mathbf{P}$. A *DL atom* is a SWRL atom of the form $P(t_1, \ldots, t_n)$ where P is a DL concept or role, i.e. $P \in \mathbf{C}$ or $P \in \mathbf{R}$ where $\mathbf{C}$ and $\mathbf{R}$ are defined based on the $\mathcal{SROIQ}$ signature $\langle \mathbf{I}, \mathbf{A}, \mathbf{N} \rangle$. All other SWRL atoms are *non-DL atoms*.

A SWRL rule of the form $B \to H$ is *DL-safe* if all variables in $B \to H$ occur in a non-DL atom in B. A set of SWRL rules is *DL-safe* if all of its rules are DL-safe. ◇

Note that the distinction of DL atoms and non-DL atoms only makes sense if we disallow rules that entail information about non-DL atoms from DL atoms – this is obviously given when restricting attention to DL-safe rules. The previous definition is also the first case where it is relevant to distinguish the designated sets $\mathbf{A}$ and $\mathbf{N}$ from arbitrary unary or binary predicates in $\mathbf{P}$. In particular, the underlying SWRL signature is relevant for determining if a set of rules is DL-safe or not.

For an example, consider again the SWRL rules from Fig. 8.1 on page 173. The predicates `orderedDish`, `contains`, and `FishProduct` are used in description logic concepts and thus must be role and concept names, respectively. Therefore, rule (1) is not DL-safe since y is used only in the DL atom $\texttt{FishProduct}(y)$. For similar reasons, rule (3) is not allowed but all other rules are indeed DL-safe.

DL-safety is easily recognised by checking whether there are enough non-DL atoms in each rule premise. Some care must still be taken since DL-safety is not an intrinsic feature that a SWRL rule may have since it depends on underlying SWRL signature. To see this, we can take a different perspective on the rules of Fig. 8.1. As we have seen in Section 8.1, rule (1) and rules (3) to (5) could similarly be considered as $\mathcal{SROIQ}$ rules, while rule (2) does not meet the requirements. Using DL Rules and DL-safe rules together is no problem since the former are merely a syntactic shortcut for description logic axioms. We just have to consider all predicates in DL Rules as role and concept names. Rule (1) and rule (3), which we found not to be DL-safe above, could thus also be considered as DL Rules. But when doing so, the predicates `Dish` and `dislikes` also must be part of the DL signature, and thus rules (2) and (4) are no longer DL-safe.

Summing up, we can treat the rules of Fig. 8.1 in at least two ways: either we use rules (2), (4), and (5) as DL-safe rules, or we use rule (1) and rules (3) to (5) as $\mathcal{SROIQ}$ rules. In each case, we can also use the given facts, but no further rules. Hence, neither approach is quite satisfying, since we have to neglect one or the other rule in each of the cases. But the definition of DL-safety suggests a way to get closer to our original rule set. Namely, whenever a rule is not DL-safe for a particular signature, it can be modified to become DL-safe by adding further non-DL atoms to the rule premise for all variables that did not appear in such atoms yet. We can introduce a fresh unary non-DL predicate O and use atoms of the form $O(x)$ to ensure the DL-safety conditions for a variable x. When viewing rules (1) and rules (3) to (5) as DL Rules, e.g., we can modify rule (2) to become DL-safe as follows:

(2') $\texttt{orderedDish}(x, y) \land \texttt{dislikes}(x, y) \land O(x) \land O(y) \to \texttt{Unhappy}(x)$

This new rule is indeed DL-safe since both x and y occur in non-DL atoms, and hence it can be used together with the other (DL) rules. But, unfortunately, this rule does not allow for any additional conclusions since one can always find an interpretation where O is interpreted as the empty set, so that rule (2') is never applicable. Adding $O(x)$ and $O(y)$ imposes additional conditions for applying the rule. Therefore we would like to ensure that O must encompass as many elements as possible. A first idea might be to add the rule $O(x)$, i.e. the fact that O encompasses *all* elements. But this rule would not be DL-safe, as x does not occur in a non-DL atom in the premise. A little reflection shows that we can only assert that concrete elements belong to O, e.g., by writing $O(\texttt{markus})$. By giving additional facts of this kind, we can extend the applicability of rule (2') to further cases.

Thus, consider a SWRL rule base that consists of the rules (1) and (3) to (5), and all facts of Fig. 8.1, together with the additional rule (2') and facts $O(c)$ for each individual name a that occurs in Fig. 8.1. Based on this rule base, we can obtain all conclusions of the underlying $\mathcal{SROIQ}$ rule base. For example, we find that Markus ordered a dish that he dislikes, as expressed by the description logic assertion

$$(\exists\texttt{orderedDish}.\exists\texttt{dislikes}^{-}.\{\texttt{markus}\})(\texttt{markus})$$

which we could check with a DL reasoner. An explicit way to read this expression is as follows: Markus belongs to the class of things who ordered a dish that is disliked by someone in the class {`markus`}, of which Markus is the only member.

In spite of this conclusion, we cannot infer that Markus is unhappy. The DL-safe rule (2') is applicable only if the variables x and y represent members of the class denoted by O. But we can always find an interpretation where this is not the case for the element that represents the unnamed dish that Markus ordered.

In contrast, we know that Anja ordered a particular Thai curry dish called "Thai Red Curry" and again we may conclude that she dislikes this dish. Since the domain element that corresponds to Anja's dish is represented by the constant symbol `thaiRedCurry`, the DL-safe rule (2') is applicable and we derive `Unhappy(anja)`. The only other instance of `Unhappy` that we can conclude is `bijan`, which follows by applying rules (1) and (2').

This example also provides some intuition of why the DL-safety restriction is enough to ensure decidability of reasoning. Namely, DL-safety effectively restricts the applicability of rules to those domain elements that are identified by constant symbols, i.e. to the elements for which we can instantiate the predicate O (or any other non-DL predicate we may use). Since we only ever have a finite number of constant symbols, rules are applicable in only a finite number of cases. The DL-safe rule (2'), e.g., could also be replaced by rules without variables that enumerate all the basic cases that are covered:

$\texttt{orderedDish}(\texttt{anja},\texttt{thaiRedCurry}) \wedge \texttt{dislikes}(\texttt{anja},\texttt{thaiRedCurry})$
$\quad\rightarrow \texttt{Unhappy}(\texttt{anja})$
$\texttt{orderedDish}(\texttt{markus},\texttt{thaiRedCurry}) \wedge \texttt{dislikes}(\texttt{markus},\texttt{thaiRedCurry})$
$\quad\rightarrow \texttt{Unhappy}(\texttt{markus})$
$\texttt{orderedDish}(\texttt{markus},\texttt{anja}) \wedge \texttt{dislikes}(\texttt{markus},\texttt{anja})$
$\quad\rightarrow \texttt{Unhappy}(\texttt{markus})$
...

While this still yields exponentially many rules, these rules now are easier to deal with for a description logic reasoner. In fact, rules without variables can always be considered as DL Rules, and could thus even be transformed into description logic axioms. This approach, however, is not feasible in practice, since it creates an exponential amount of new axioms that the reasoner must take into account. Reasoners with direct support for DL-safe rules, in contrast, may process such rules rather efficiently, and in an optimised fashion. Examples of systems that currently support DL-safe rules are KAON2 [MS06] and Pellet [SPG+07].

It has been mentioned that there is a second perspective that one may take on DL-safe rules. The above discussions have shown that, intuitively, DL-safe rules are applicable only to elements that are denoted by constant symbols. Instead of imposing a syntactic requirement to ensure this, we may directly build this restriction into the semantics of SWRL. One way to do that is to change the definition of variable assignments, requiring that variables can only be assigned to domain elements of the form $a^{\mathcal{I}}$ for some constant symbol $a \in \mathbf{I}$. Such domain elements are sometimes called *named elements*. Another possible approach is to assume that the premise of every rule (DL-safe or not) is silently extended with conditions $O(x)$ where O is defined by facts $O(a)$ for each constant symbol a. Both approaches are essentially equivalent in that they allow us to write arbitrary SWRL rules and use them like DL-safe rules. This is, in fact, what some description logic reasoners that support DL-safe rules will automatically do when encountering a rule that is not DL-safe.

The above perspective is convenient since it allows users to specify arbitrary rules without considering the details of their semantics. However, this approach introduces some confusion, since the term “DL-safe rule” might now be used for two different things. On the one hand, it might refer to a SWRL rule that respects the syntactic restrictions explained above. On the other hand, it might denote a rule that is syntactically similar to SWRL, but which is evaluated under a modified semantics that restricts its conclusions. The second approach can also be viewed as an incomplete way of reasoning with SWRL: all conclusions that the rules entail under the “DL-safe semantics” are also correct conclusions under the standard SWRL semantics, but some conclusions might not be found. An example of such a lost conclusion is `Unhappy(markus)` which we could derive in SWRL in Section 4.2.1 but not with the DL-safe rules above.

While the relationship between the two approaches is straightforward, it is important to clarify the intended meaning when specifying SWRL rules. This is even more the case when Description Logic Rules are also considered, since SWRL rules that are not DL-safe may still be suitable as DL Rules.

9.2 DL Rules with Safe Variables

The extended introduction to DL-safe rules in the previous section already explained that DL Rules and DL-safe rules can be used in combination. This immediately leads to larger decidable fragments of SWRL, but this loose integration of the two approaches can be further extended. In this section, we introduce the concept of DL-safe variables and we show how it can be applied to obtain larger decidable fragments of SWRL. The resulting class of rule languages is called *DL+safe rules* since it represents a natural integration of DL Rules and DL-safe rules that generalises both approaches.

Definition 9.2.1 Consider a SWRL signature $\langle \mathbf{I}, \mathbf{P}, \mathbf{V} \rangle$ as in Definition 9.1.1, and a SWRL rule $B \to H$ over that signature. A variable x is *DL-safe* for $B \to H$ if it occurs in a non-DL atom in B. ◇

A DL-safe rule therefore is a SWRL rule that contains only DL-safe variables in its head. As before, this notion is only useful if we ensure that non-DL atoms are not entailed from DL atoms in the considered rule bases. If this can be taken for granted, then the satisfiability of rule bases is typically not affected when replacing rules with DL-safe variables by their groundings, defined as follows.

Definition 9.2.2 The DL-safe grounding $\mathsf{ground}(B \to H)$ of a rule $B \to H$ is the set of all rules that can be obtained by uniformly replacing DL-safe variables in $B \to H$ with individual names of the given signature. Given a set of SWRL rules RB, we use $\mathsf{ground}(\mathrm{RB})$ to denote the union of all DL-safe groundings for rules of RB. ◇

Much of the discussion of Section 9.1 applies to DL-safe variables as well. In particular, we are free to choose the alternative perspective that DL-safe variables are subject to a different semantic interpretation that restricts variable assignments for those variables to named elements. This approach could be formalised by including a designated set of DL-safe variables into SWRL signatures. To avoid confusion, we stay true to the formulation of Definition 9.2.1.

In essence, DL-safe variables behave like individual names, and we can extend the definition of DL Rules accordingly.

Definition 9.2.3 Consider a description logic $\mathcal{L}$ as in Definition 8.4.2. Given a set RB of SWRL rules over a signature $\mathscr{S}$, let $\mathscr{S}'$ denote the signature obtained from $\mathscr{S}$ by declaring all unary predicates to be concept names. Then RB is an *$\mathcal{L}$+safe rule base* over $\mathscr{S}$ if

- all rules in RB that contain a non-DL atom as their head are DL-safe, and
- $\mathsf{ground}(\mathrm{RB})$ is an $\mathcal{L}$ rule base over $\mathscr{S}'$ according to Definition 8.4.2. ◇

Since all atoms in DL Rules must be DL atoms, this definition implicitly requires rules to contain unary non-DL atoms only. This restriction could be weakened, but such an extension would not contribute much to the results of this chapter since we typically consider only non-DL atoms of the form $O(x)$.

It should be noted that DL+safe rules are a generalisation of DL Rules and DL-safe rules. Clearly, a DL rule base simply is a DL+safe rule base without non-DL atoms. For the case of DL-safe rules, note that every SWRL rule without variables is a DL Rule according to Definition 9.2.3. We point out that the elimination of nominals in Lemma 8.4.1 is essential for this result. Hence, any DL-safe rule is also a DL+safe rule (given that only unary and binary atoms are used, as discussed above). Since any DL knowledge base can directly be expressed as a DL Rule base, this shows that any combination of a description logic knowledge base with a set of DL-safe rules can be expressed as a DL+safe rule base.

For an extended example, consider again the rules of Fig. 8.1 on page 173. As before, we find that rules (1) and (3) to (5) are $\mathcal{SROIQ}$ rules, and hence they are clearly $\mathcal{SROIQ}$+safe rules as well. This is not the case for rule (2), but we do not need to restrict it quite as strongly as rule (2') in Section 9.1. Namely, it suffices if one of the variables is forced to be DL-safe, so we obtain two possible approximations:

(2.a) $\mathtt{orderedDish}(x, y) \wedge \mathtt{dislikes}(x, y) \wedge O(y) \rightarrow \mathtt{Unhappy}(x)$
(2.b) $\mathtt{orderedDish}(x, y) \wedge \mathtt{dislikes}(x, y) \wedge O(x) \rightarrow \mathtt{Unhappy}(x)$

where O is instantiated for all individual names as before. The grounding of either rule is a $\mathcal{SROIQ}$ rule: assuming that the DL-safe variable is replaced by constants a and b, we can apply the algorithm of Section 8.2 to obtain $\mathcal{SROIQ}$ axioms:

(2.a) $\exists\mathtt{orderedDish}.\{a\} \sqcap \exists\mathtt{dislikes}.\{a\} \sqsubseteq \mathtt{Unhappy}$
(2.b) $\{b\} \sqcap \exists U.(\{b\} \sqcap \exists\mathtt{orderedDish}.\exists\mathtt{dislikes}^{-}.\{b\}) \sqsubseteq \mathtt{Unhappy}$

where the axiom for (2.b) could be simplified by omitting the outermost conjunction and existential on the left-hand side; this optimisation is not part of the transformation algorithm. When using the $\mathcal{SROIQ}$+safe rule base with rule (2.a), only `Unhappy(bijan)` and `Unhappy(anja)` are entailed, whereas with rule (2.b) we additionally obtain the missing conclusions `Unhappy(ian)` and `Unhappy(markus)`.

Continuing with the example, we can also consider $\mathcal{SROIQ}(B_s, \times)$+safe rules as an underlying formalism. As observed in Section 8.3, the simplicity restrictions on role conjunctions allow us to consider either rules (1) and (3) to (5), or rules (1) to (3) and (5) as $\mathcal{SROIQ}(B_s, \times)$ rules. In the first case rule (2) can be treated as before, while in the second case we can consider restricted versions of rule (4) that still allow `dislikes` to be simple. It turns out that it suffices to make any of the variables x, y and z in the body of rule (4) DL-safe, leading to rules (4.x), (4.y), and (4.z). With the extended transformation of Section 8.3, we obtain the following axioms when using the constants a, b, c for grounding:

(4.x) $\{a\} \times (\mathtt{Dish} \sqcap \exists \mathtt{contains}.\exists \mathtt{dislikes}^{-}.\{a\}) \sqsubseteq \mathtt{dislikes}$
(4.y) $(\exists \mathtt{dislikes}.\exists \mathtt{contains}^{-}.\{b\}) \sqcap \exists U.(\mathtt{Dish} \sqcap \{b\}) \sqsubseteq \exists \mathtt{dislikes}.\{b\}$
(4.z) $(\exists \mathtt{dislikes}.\{c\}) \times (\mathtt{Dish} \sqcap \exists \mathtt{contains}.\{c\}) \sqsubseteq \mathtt{dislikes}$

Together with rule (4.x), the rule base entails all named instances of Unhappy, just as in the case of $\mathcal{SROIQ}$+safe rules with rule (2.b) above. With rule (4.y) we can only conclude Unhappy(bijan) and Unhappy(anja), whereas rule (4.z) lets us derive only Unhappy(bijan) and Unhappy(ian).

The next proposition shows that grounding can be used to reduce satisfiability checking of DL+safe rules (and also DL-safe rules) to satisfiability checking for the corresponding class of DL Rules.

Proposition 9.2.4 *Consider a description logic $\mathcal{L}$ as in Definition 9.2.3. Then every $\mathcal{L}$+safe rule base* RB *is equisatisfiable to* ground(RB).

In particular, if the problem of checking satisfiability of $\mathcal{L}$ knowledge bases is decidable then checking satisfiability of $\mathcal{L}$+safe rule bases is also decidable.

Proof. Consider a $\mathcal{L}$+safe rule base RB. We claim that RB and ground(RB) are equisatisfiable. Clearly, every model of RB is also a model of ground(RB). For the converse, consider a model $\mathcal{I}$ of ground(RB). An interpretation $\mathcal{I}'$ is defined to coincide with $\mathcal{I}$ regarding domain, interpretation of individuals, and interpretation of roles and concepts. For every n-ary predicate $P \notin \mathbf{A} \cup \mathbf{N}$, define $P^{\mathcal{I}} := \{\langle \delta_1, \ldots, \delta_n \rangle \in P^{\mathcal{I}} \mid \text{for all } i = 1, \ldots, n : \delta_i = a^{\mathcal{I}} \text{ for some } a \in \mathbf{I}\}$. In other words, $\mathcal{I}'$ restricts the extension of non-DL predicates to named individuals. It is easy to see that $\mathcal{I}'$ is a model of RB, since all rules in RB with non-DL atoms as heads are DL-safe.

By Definition 9.2.3, ground(RB) is an $\mathcal{L}$ rule base where the signature is modified to consider all unary predicates as concept names as in the definition. Satisfiability of ground(RB) can then be decided based on Proposition 8.4.3. □

The satisfiability-preserving reduction in the previous proof yields an exponential blow-up of the number of input rules, and hence is not a useful basis for obtaining tight upper boundaries for the worst-case complexity of satisfiability checking. Yet, this result can be considered as a way of expressing DL+safe Rules in terms of DL Rules, and in particular as a reduction of DL-safe rules to description logic axioms. In this sense, DL-safe rules do not introduce additional expressiveness, although the term “syntactic sugar” is rather not appropriate given the exponential blow-up of the rewriting and the fact that only satisfiability is preserved. However, the well-known result that DL-safe rules do not increase the ExpTime worst-case complexity of reasoning for $\mathcal{SHIQ}$ [MSS05] suggests that

DL+safe rules may not lead to an exponential increase in complexity. This is confirmed for the DLs $\mathcal{SROIQ}$ and $\mathcal{SROEL}(\sqcap_s, \times)$ in the following two sections.

9.3 Reasoning Complexity of $\mathcal{SROIQ}$+safe Rules

Intuitively, every DL+safe rule represents an exponential number of DL Rules that are obtained by replacing DL-safe variables with individual symbols. Based on this intuition that was the basis of the proof of Proposition 9.2.4, we obtain an upper bound for the complexity of reasoning with DL+safe rules that is exponentially larger than the upper bound of the underlying DL. For example, it is immediately clear that satisfiability of $\mathcal{SROIQ}$+safe rule bases can be decided in non-deterministic triple-exponential time. In this section, we show that this result can be refined to obtain an N2ExpTime upper complexity bound, showing that this reasoning problem must be N2ExpTime-complete. Moreover, we use the results of Chapter 5 to obtain results for the slightly larger class of $\mathcal{SROIQ}(B_s, \times)$+safe rules.

The fact that all standard reasoning tasks for $\mathcal{SROIQ}$ knowledge bases can be decided in N2ExpTime was shown in [Kaz08] by providing an exponential reduction from $\mathcal{SROIQ}$ to C^2 – the two-variable fragment of first-order logic with counting quantifiers – for which reasoning is known to be NExpTime-complete, and we have extended this transformation to $\mathcal{SROIQ}(B_s, \times)$ in Section 5.2. The transformation is based on the use of non-deterministic finite automata (NFA) that have been defined in [HS04, HKS06] to capture the interplay of complex role inclusion axioms. We do not repeat the details of this construction here, and merely quote the essential results. Proofs for the following facts can be found in [HKS06] and the accompanying technical report.

Fact 9.3.1 *Consider a $\mathcal{SROIQ}$ knowledge base* KB. *For each (possibly inverse) role $R \in \mathbf{R}$, there is an NFA $\mathcal{A}_R$ over the alphabet $\mathbf{R}$ such that the following holds for every model $\mathcal{I}$ of* KB, *and for every word $S_1 \ldots S_n$ accepted by $\mathcal{A}_R$:*

$$\textit{If } \langle \delta_i, \delta_{i+1} \rangle \in S_i^{\mathcal{I}} \textit{ for each } i \in \{1, \ldots, n\}, \textit{ then } \langle \delta_1, \delta_{n+1} \rangle \in R^{\mathcal{I}}.$$

Moreover, let $<$ denote a strict linear order that witnesses regularity of RB *as required in Definition 3.1.4. For each $R \in \mathbf{N}$, the number of states of $\mathcal{A}_R$ is bounded exponentially in the* depth *of* KB *that is defined as:*

$$\max\{n \mid \textit{there are } S_1 < \ldots < S_n \textit{ such that } T_{i1} \circ \ldots \circ S_i \circ \ldots \circ T_{im_i} \sqsubseteq S_{i+1} \in \mathrm{KB}\}$$

Considering the DL Rule normalisations from Fig. 8.2 and 8.4, and the transformations from Fig. 8.3, it is easy to see that the grounding of DL-safe variables does not increase the depth of a knowledge base. More formally, we obtain the following.

Lemma 9.3.2 *Given a* $\mathcal{SROIQ}(B_s,\times)$*+safe rule base* RB*, let* RB′ *denote the rule base that is obtained by uniformly replacing each DL-safe variable in* RB *by some (arbitrary) individual name. Moreover, let* KB(RB′) *and* KB(ground(RB)) *denote the* $\mathcal{SROIQ}(B_s,\times)$ *knowledge bases that correspond to* RB′ *and* ground(RB) *as defined in Proposition 8.4.3, where unary predicates of* RB *are considered as concept names. Then the depth of* KB(RB′) *is equal to the depth of* KB(ground(RB)).

The transformation of $\mathcal{SROIQ}$ knowledge bases into C^2 theories in [Kaz08] proceeds in three steps: (1) the input axioms are transformed into a simplified normal form as discussed in Section 5.2, (2) complex role inclusion axioms are eliminated, and (3) the resulting $\mathcal{SROIQ}$ axioms are expressed as formulae of C^2. Step (1) can be executed in linear time and leads to a $\mathcal{SROIQ}$ knowledge base that semantically emulates the original knowledge base. Step (2) applies a technique that was originally introduced in [DN05]. Every axiom of the form $A \sqsubseteq \forall R.B$ is replaced by the following set of axioms, where $\mathcal{A}_R$ is the NFA as introduced above, and X_q are fresh concept names for each state q of $\mathcal{A}_R$:

$$\begin{array}{rl} A \sqsubseteq X_q & q \text{ is the initial state of } \mathcal{A}_R \\ X_q \sqsubseteq \forall S.X_{q'} & \mathcal{A}_R \text{ has a transition } q \xrightarrow{S} q' \\ X_q \sqsubseteq B & q \text{ is a final state of } \mathcal{A}_R \end{array}$$

Moreover, all complex RIAs of the form $S_1 \circ \ldots \circ S_n \sqsubseteq R$ with $n \geq 2$ are deleted. The number of new axioms (and fresh concept names) that are introduced for each axiom of the form $A \sqsubseteq \forall R.B$ is bounded by the sum of the number of states and transitions in $\mathcal{A}_R$, and the number of transitions in turn is linear in the number of role names and states. According to Fact 9.3.1, the number of axioms introduced for each axiom $A \sqsubseteq \forall R.B$ is exponentially bounded in the depth of the knowledge base. The overall size of the knowledge base after step (2) therefore is bounded by a function that is linear in the size of the knowledge base and exponential in the depth of the knowledge base.

Step (3), finally, is a simple rewriting that does not increase the size of the knowledge base. Lemma 9.3.2 therefore allows us to draw the following conclusion. As shown in Section 5.2, it is possible to extend this transformation to cover additional types of role expressions, and this extension does not interfere with the RIA elimination in step (2).

Theorem 9.3.3 *The problem of deciding satisfiability of* $\mathcal{SROIQ}(B_s,\times)$*+safe rule bases (and thus also of* $\mathcal{SROIQ}$*+safe rule bases) is* N2ExpTime*-complete w.r.t. the size of the rule bases.*

Proof. Consider a $\mathcal{SROIQ}(B_s,\times)$+safe rule base RB and a $\mathcal{SROIQ}(B_s,\times)$ rule base ground(RB) as in Lemma 9.3.2. Since the size of KB(ground(RB)) is linear in the size of ground(RB) (Theorem 8.2.5), we find that the sizes of ground(RB) and of KB(ground(RB)) both are exponential in the size of RB. By Lemma 9.3.2 and Theorem 8.2.5, the depth of KB(ground(RB)) is linear in the size of RB. The size of the knowledge base that is obtained in step (2) of the above transformation of KB(ground(RB)) to C^2 is bounded by the product of the number of axioms in KB(ground(RB)) and the maximal number of states in NFA $\mathcal{A}_R$. Since both are exponential in the size of RB, the overall bound is still exponential in this size. Hence, the transformation to C^2 in step (3) yields a theory that is exponential in the size of RB, even when taking into account the additional transformation steps that were introduced for $\mathcal{SROIQ}(B_s, \times)$ role expressions in Section 5.2. Since the satisfiability problem for C^2 theories is NEXPTIME-complete [PH05], we find that satisfiability of RB can be decided in N2EXPTIME.

Hardness follows from the N2EXPTIME-hardness of $\mathcal{SROIQ}$ [Kaz08]. □

This shows that the worst-case complexity of reasoning in $\mathcal{SROIQ}$+safe rules is not higher than the worst-case complexity of reasoning in $\mathcal{SROIQ}$. Yet, the exponential increase in the input size, although it is not an increase of the knowledge base's depth, suggests that in-advance grounding is not the most promising approach for implementing reasoners. In particular, the method is guaranteed to require exponential runtime in all cases, whereas successful DL reasoning algorithms typically are able to avoid exponential behaviour for many input problems. It is not hard to see that optimisations could be applied to obtain more promising algorithms, e.g. by noting that grounding leads to a large number of structurally similar axioms that can be treated analogously during reasoning. This can be exploited, for example, when constructing NFAs from the knowledge base. The general strategy underlying such optimisations is *deferred grounding*: instead of initially replacing DL-safe variables with constants, DL-safe variables are kept unchanged and treated like constant symbols in subsequent inferencing steps, until concrete values are really needed. Even when DL-safe variables are eventually instantiated, it is not necessary to compute all possible instantiations at once. These observations suggest that the algorithmic treatment of $\mathcal{SROIQ}$+safe rules could indeed achieve similar levels of efficiency as the treatment of $\mathcal{SROIQ}$ knowledge bases, but further research and development will be required to arrive at practical implementations.

9.4 Tractable DL-Safe Rules: ELP

We have seen that DL+safe rules do not necessarily increase the worst-case complexity of reasoning as compared to the underlying DL. However, DL+safe Rules are inherently intractable since they encompass DL-safe rules which can in turn be used to express arbitrary datalog programs that use unary and binary predicate symbols only. Checking satisfiability of such programs is still NP-complete.[2] In this section, we therefore study how DL-safe variables can be combined with DL Rules to obtain tractable rule languages. This approach leads to the rule language ELP that extends $\mathcal{SROEL}(\sqcap_s, \times)$ rules as defined in Section 8.5 with DL-safe variables while still allowing polytime reasoning.

Intuitively speaking, the high worst-case complexity of datalog is due to the fact that arbitrarily complex relationships can be expressed in rule bodies of unbounded size. We already noted that reasoning becomes tractable when restricting to datalog rules with a bounded number of variables. As an alternative, one can constrain the structure of rule bodies in the spirit of DL Rules, as shown in the next definition.

Definition 9.4.1 Consider a set RB of SWRL rules over some signature $\mathscr{S}$, and let $\mathscr{S}'$ denote the signature obtained from $\mathscr{S}$ by declaring all unary predicates to be concept names.

Then RB is an ELP *rule base* if the following holds:

- all rules in RB that contain a non-DL atom as their head are DL-safe,
- ground(RB) is a $\mathcal{SROEL}(\sqcap_s, \times)$ rule base over $\mathscr{S}'$, and
- RB is a $\mathcal{SROIEL}(\sqcap_s, \times)$ rule base over $\mathscr{S}'$, where $\mathcal{SROIEL}(\sqcap_s, \times)$ is the extension of $\mathcal{SROEL}(\sqcap_s, \times)$ with inverse roles.

A set of range restrictions RR is *admissible* for an ELP rule base RB if RR is admissible for ground(RB) according to Definition 8.5.1. An *extended* ELP *rule base* is the union of an ELP rule base RB and a set of range restrictions RR that are admissible for RB. ◇

The above definition ensures that any ELP rule base is a $\mathcal{SROEL}(\sqcap_s, \times)$+safe rule base, but it also imposes additional restrictions on the structure of DL-safe variables. In essence, the requirement of RB being a $\mathcal{SROIEL}(\sqcap_s, \times)$ rule base implies that the body of any rule in RB does not contain "undirected cycles"[3] other than those that can be expressed by means of local reflexivity and conjunction of simple roles.

Returning to our earlier example from Fig. 8.1, we now find that all rules but rule (4) are in ELP extended with admissible range restrictions. In contrast

[2]Hardness is easy to establish, e.g. by reducing the 3-colouring problem of binary graphs [Pap94] to satisfiability checking. Inclusion can be shown by providing a non-deterministic polynomial-time algorithm for checking ground entailments. This can be accomplished by guessing a suitable proof tree [Llo88], where we note that each node in the tree corresponds to one out of polynomially many available ground atoms, so that a polynomial presentation of the complete tree is possible.

[3]This intuitive terminology alludes to the graphical interpretation from Definition 8.2.1.

to the case of $\mathcal{SROIQ}(B_s, \times)$+safe rules considered in Section 9.2, however, we cannot select an arbitrary variable of rule (4) to be DL-safe. Only if z is DL-safe will the grounded rule be a $\mathcal{SROEL}(\sqcap_s, \times)$ rule. This corresponds to rule (4.z) as considered in Section 9.2 and indeed we found that the $\mathcal{SROIQ}(B_s, \times)$ translation of this rule was a $\mathcal{SROEL}(\sqcap_s, \times)$ axiom. As before, we obtain the conclusions Unhappy(bijan) and Unhappy(ian).

Interestingly, ELP can be considered as a SWRL fragment that subsumes and extends the logical formalisms underlying OWL EL and OWL RL. The former should be obvious, since the DL $\mathcal{SROEL}(\sqcap_s, \times)$ subsumes the abstract – i.e. unrelated to datatypes – logical features of OWL EL. It has been discussed in Section 6.2 that the DL $\mathcal{RL}$ plays a similar rôle for OWL RL. However, the union of both of these logics subsumes Horn-$\mathcal{FLE}$ (see Section 6.4) for which inferencing is already ExpTime-hard. Hence, ELP cannot subsume this union without giving up its main design criterion of tractability. The following theorem shows how ELP can still support inferencing for both languages, and even achieve some amount of interoperability between them.

Theorem 9.4.2 *Given an extended $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge base* KB_1 *and a $\mathcal{RL}$ knowledge base* KB_2 *that are based on a signature $\mathscr{S}$, there is an extended* ELP *rule base* RB *(possibly over an extended signature) such that the following holds for any ground atom α of the form $C(a)$ or $R(a, b)$ over $\mathscr{S}$:*

- *if* $\mathrm{KB}_1 \models \alpha$ *or* $\mathrm{KB}_2 \models \alpha$ *then also* $\mathrm{RB} \models \alpha$*,*
- *if* $\mathrm{RB} \models \alpha$ *then* $\mathrm{KB}_1 \cup \mathrm{KB}_2 \models \alpha$*,*

and RB *can be computed from* KB_1 *and* KB_2 *in logarithmic space w.r.t. the size of the knowledge bases.*

Proof. It has been noted in Proposition 6.2.2 that $\mathcal{RL}$ axioms can be translated into datalog rules by using the first-order transformation specified in Section 3.2. It is well known that all ground entailments of a datalog program can be derived by applying rules only to named individuals, and hence the resulting rules can be extended by auxiliary body atoms $O(x)$ for each variable x they contain. As before, we add facts $O(a)$ for each individual name a of $\mathscr{S}$. Here we assume that O is fresh for $\mathscr{S}$, i.e. that it does not occur in $\mathscr{S}$.

It is easy to see that all rules that result from this transformation of KB_2 are $\mathcal{SROIEL}(\sqcap_s, \times)$ rules, with the only exception of those rules that are obtained from axioms $A \sqsubseteq\ \leqslant 1\, R.B$. Namely, these rules contain equality statements of the form $y_1 \approx y_2$ in their heads, and such atoms have not been allowed in any DL Rule language. As discussed in Section 4.1.3, however, the equality predicate in datalog can be replaced by a suitable axiomatisation. Hence, we introduce a fresh role name $R_{\approx}$, replace atoms $y_1 \approx y_2$ with $R_{\approx}(y_1, y_2)$, and add new rules to axiomatise $R_{\approx}$ as an equality relation as in Section 4.1.3. Moreover, since equalities only occur in DL-safe rules, all the auxiliary rules for axiomatising equality can also be modified to be DL-safe.

It is not hard to see that the rule base that is obtained by applying these translations to the datalog rules that are obtained for KB_2 are indeed in ELP, and that they entail the same ground facts as KB_2. Now RB is obtained as the union of this set of DL-safe rules with KB_1, expressed in terms of SWRL rules as usual. The previous observations immediately establish the first part of the claim. For the other direction, it suffices to note that (the SWRL version of) $\mathrm{KB}_1 \cup \mathrm{KB}_2 \cup \{O(a) \mid a \in \mathbf{I}_{\mathscr{S}}\}$, entails RB. Since α does not contain O, this shows the second part of the claim. □

Note that the resulting ELP rule base entails all individual consequences of KB_1 and KB_2, and some but not all consequences of their (unsafe) union. ELP thus provides a means of combining $\mathcal{SROEL}(\sqcap_s, \times)$ (OWL EL) and $\mathcal{RL}$ (OWL RL) in a way that prevents intractability, while still allowing for a controlled interaction between both languages. We argue that this is a meaningful way of combining both formalisms in practice since only some $\mathcal{RL}$ axioms must be restricted to safe variables. Simple atomic concept and role inclusions, for example, can always be considered as $\mathcal{SROEL}(\sqcap_s, \times)$ axioms, and all concept subsumptions entailed from the $\mathcal{SROEL}(\sqcap_s, \times)$ part of a combined knowledge base do also affect classification of instances in the $\mathcal{RL}$ part. $\mathcal{RL}$ thus gains the terminological expressivity of $\mathcal{SROEL}(\sqcap_s, \times)$ while still having available specific constructs that may only affect the instance level.

Next, we want to show that reasoning with extended ELP rule bases is indeed tractable. Our earlier results on extended $\mathcal{SROEL}(\sqcap_s, \times)$ rule bases already provide a way of deciding satisfiability of ELP rule bases by first grounding DL-safe variables, and then proceeding with the elimination of range restrictions and transformation to datalog. This direct approach, however, would incur an exponential blow-up of the rule base. The proof thus proceeds by decomposing ELP rules into rules containing a limited number of variables. The grounding of DL-safe variables then can only produce a polynomially bounded number of new rules. After translating from $\mathcal{SROEL}(\sqcap_s, \times)$ rule bases to datalog as in Section 8.5, the number of variables per rule is still bounded, which leads to the desired tractability result.

The decomposition of ELP rules into rules with a bounded number of variables exploits the forest-like structure of rule bodies by iteratively reducing branches of trees. Since $\mathcal{SROEL}(\sqcap_s, \times)$ does not support inverse roles, this reduction is more complicated than the normalisation techniques that were used for $\mathcal{SROEL}(\sqcap_s, \times)$ knowledge bases in Section 5.2.

Lemma 9.4.3 *Every extended* ELP *rule base* RB *is semantically emulated by an extended* ELP *rule base* RB′ *that contains at most three variables per rule, and that has the simplified form of Proposition 8.5.2. Moreover,* RB′ *can be computed in time polynomial w.r.t. the size of* RB.

Proof. As a first step, we simplify the form of rules in RB. Nested concept conjunctions and existential role restrictions with compound subconcepts are eliminated as in Proposition 8.5.2. However, we explicitly allow concept expressions of the form $\exists R.\mathsf{Self}(t)$ and $\exists R.\{a\}(t)$, and we will not decompose them in any way. To the contrary, we replace role atoms $R(t,t)$ and $R(t,a)$ with $a \in \mathbf{I}$ by concept expressions $\exists R.\mathsf{Self}(t)$ and $\exists R.\{a\}(t)$, respectively. An essential property is that both of these expressions can later be expressed in SWRL without using DL concept constructors, and without introducing fresh variables. We obtain an extended ELP rule base RB_1 that contains only concept expressions that are of one of the forms $A \in \mathbf{A}$, $\{a\}$ with $a \in \mathbf{I}$, $\exists R.\mathsf{Self}$, $\exists R.\{a\}$, $\top$, $\bot$, and $\exists R.A$ with $A \in \mathbf{A}$ (only in rule heads). Rules with body atoms of the form $\bot(t)$ or head atoms of the form $\top(t)$ are assumed to be deleted. Expressing $R(t,t)$ and $R(t,a)$ in terms of concept expressions is useful since these special cases would otherwise need to be distinguished from other cases where role atoms are considered below. Clearly, RB_1 can be computed in time polynomial w.r.t. the size of RB, and it semantically emulates RB.

Next, we eliminate individual names in argument positions, which can be accomplished by replacing single occurrences of individual names a by fresh variables x, and adding nominal concepts $\{a\}(x)$ to the rule body. This step is similar to steps (2) and (3) in Fig. 8.2, and it is easy to see that the resulting rule base RB_2 is still an extended ELP rule base that semantically emulates RB. Note that it is important for this result that individual occurrences of constants are replaced by different variables. For example, $A(x) \wedge S(a,y) \wedge R(y,a) \to T(x,y)$ is in ELP, and so is $A(x) \wedge S(z,y) \wedge R(y,z') \wedge \{a\}(z) \wedge \{a\}(z') \to T(x,y)$ but not $A(x) \wedge S(z,y) \wedge R(y,z) \wedge \{a\}(z) \to T(x,y)$. In the following, we can therefore assume that all terms in rules are variables (DL-safe or not).

In the next step, we extract role conjunctions from the rules of RB_2 to ensure that all rules with more than three variables contain at most one atom that connects two given variables. As an example, consider the ELP rule $A(x) \wedge O(z) \wedge R(z,y) \wedge S(y,z) \to T(x,y)$ where $O(z)$ is a non-DL atom so that z is DL-safe. Note that we cannot treat the occurrences of z independently as in the case of individual names. Using a fresh role name V, the above rule can be expressed by rules $O(z) \wedge R(z,y) \wedge S(y,z) \to V(z,y)$ and $A(x) \wedge O(z) \wedge V(z,y) \to T(x,y)$. Note that the direction chosen for V is not arbitrary, since the rule $O(z) \wedge R(z,y) \wedge S(y,z) \to V(y,z)$ is not in ELP. Based on the observation that expressions of the form $R(z,y) \wedge S(y,z)$ can only occur in ELP rule bodies if at least one of y and z is DL-safe, it is not hard to

obtain a general transformation rule from this example:
Select a rule $B \to H \in \mathrm{RB}_2$ with more than three variables and do the following:

- if there are $S, R \in \mathbf{N}$ such that $\{R(x,y), S(x,y)\} \subseteq B$, then replace $B \to H$ with rules $B \cup \{V(x,y)\} \setminus \{R(x,y), S(x,y)\} \to H$ and $\{R(x,y), S(x,y)\} \to V(x,y)$ where V is a fresh role name,
- if there are $S, R \in \mathbf{N}$ and a non-DL predicate O with $\{R(x,y), S(y,x), O(x)\} \subseteq B$, then replace $B \to H$ with rules $B \cup \{V(x,y)\} \setminus \{R(x,y), S(y,x)\} \to H$ and $\{R(x,y), S(y,x), O(x)\} \to V(x,y)$ where V is a fresh role name.

Let RB_3 denote the rule base that is obtained from RB_2 by applying the above transformation exhaustively. Clearly, RB_3 again semantically emulates RB and can be computed in polynomially many steps.

We are now ready to transform the extended ELP rules of RB_3 into extended ELP rules with at most 3 variables per rule. To this end, we first introduce some auxiliary notions, where we adopt the graph-based perspective that was first introduced in Definition 8.2.1. Consider some rule $B \to H$:

- A *direct connection* Γ from t to u in B is a singleton set of the form $\Gamma = \{R(t,u)\} \subseteq B$.
- A *connected component* of B is a non-empty subset $S \subseteq B$ such that, for all terms $t \neq u$ occurring in S, we find that t and u are connected in S. A *maximal connected component* (MCC) is a connected component that has no supersets that are connected components.
- A variable x is *final for H* if $H = R(t,x)$ or $H = C(x)$.
- Given a subset S of B, we say that S is *reducible* if it contains variables that are neither a root (as in Definition 8.2.1) of H nor final for H.
- Let S be an MCC of B, and consider a direct connection Γ from a term t to a term u in S. Let $S_{\Gamma,t}$ be the set of all atoms in $S \setminus \Gamma$ that contain some term t' connected to t in $S \setminus \Gamma$. Similarly, let $S_{\Gamma,u}$ be the set of all atoms in S that contain some term u' connected to u in $S \setminus \Gamma$.

Intuitively, the sets $S_{\Gamma,t}$ and $S_{\Gamma,u}$ consist of all atoms to the "left" or to the "right" of the connection Γ that can be reached from t and u, respectively, without using the atom of Γ.

Since DL Rules cannot contain proper dependency cycles, and due to the transformation of RB_2 to RB_3 above, every connected component S of a rule in RB_3 has some root element in S.

We can now proceed to reduce the forest structure of rule bodies.

In each iteration step of the reduction, select some rule $B \to H$ in RB_3 that contains more than three variables and some reducible MCC S of B, and apply one of the following transformations. We use x to denote the root variable of H.

(1) If S contains no variable that is final for H, then let t be a root variable in S. The rule $B \to H$ is replaced by three new rules $(B \setminus S) \cup \{X(x)\} \to H$, $\{\top(x), Y(t)\} \to X(x)$, and $S \to Y(t)$, where X, Y are fresh concept names.

For all other cases, assume that the variable y in S is final for H.

(2) There is a direct connection $\Gamma = \{R(t, u)\} \subseteq S$ such that $u \neq y$ and $S_{\Gamma,u}$ does not contain x or y. Then rule $B \to H$ is replaced by two new rules $B \cup \{X(t)\} \setminus (S_{\Gamma,u} \cup \Gamma) \to H$, and $\Gamma \cup S_{\Gamma,u} \to X(t)$, where X is a fresh concept name.

(3) There is a direct connection $\Gamma = \{R(t, y)\}$ from some variable $t \neq x$ to y. Let s be a root variable of $S_{\Gamma,t}$ if the latter is non-empty, and set $s := t$ otherwise. The rule $B \to H$ is replaced by three new rules $B \cup \{V(s, y)\} \setminus (S_{\Gamma,t} \cup \Gamma) \to H$, $\{W(s, t)\} \cup \Gamma \to V(s, y)$, and $S_{\Gamma,t} \to W(s, t)$, where V, W are fresh non-simple role names. Moreover, if $H = S(x, y)$ then a range restriction $V(z, z') \to D(z')$ is added for every range restriction $S(z, z') \to D(z') \in \mathrm{RB}_3$.

(4) There is a direct connection Γ from y to some variable u such that $S_{\Gamma,u}$ is reducible. We distinguish two cases:

   (a) There is a direct connection from some term $t \notin \{x, y\}$ to u. Then rule $B \to H$ is replaced by two new rules $B \cup \{V(x, u)\} \setminus S_{\Gamma,u} \to H$ and $S_{\Gamma,u} \to V(x, u)$, where V is a fresh non-simple role name.

   (b) The above is not the case, and u is involved in a direct connection $\Gamma' = \{R'(u, u')\}$ besides $\Gamma = \{R(y, u)\}$, such that $S_{\Gamma',u'}$ contains x. The rule $B \to H$ is replaced by two new rules $B \cup \{V(y, u')\} \setminus \{R(y, u), R'(u, u')\} \to H$ and $\{R(y, u), R'(u, u')\} \to V(y, u')$, where V is a fresh non-simple role name.

This iteration is repeated until no further transformation is applicable, and the resulting set of rules is denoted by RB_4. In all considerations below, we will use the notation of the above cases when considering some transformation step, and refer to the generated rules in each step by the order of their appearance in the transformation steps (e.g. by saying "first rule of (2)" or "rule 3 of (3)").

Claim 1 RB_4 is an extended ELP rule base.

For most cases, it is readily seen that the created rules are ELP rules (unless they are range restrictions), which follows from the fact that subsets of rule bodies of ELP rules satisfy the essential requirements of Definition 9.4.1, and in particular still expose the tree shape required by Definition 8.2.3. An additional check is required to verify that, for some new rule head $X(x)$ or $V(x, t)$ with x unsafe, x is indeed a root in the body. This is readily verified for all cases. Moreover, it is easy to see that the translation preserves conditions on simplicity of roles, since all newly introduced roles are non-simple, and since they do never occur in a body position where simplicity is required.

Further care must be taken when introducing auxiliary roles, since auxiliary role atoms create new paths in rule bodies that might violate the required tree shape. New role atoms are introduced in (3), but only to either replace an existing direct connection to the variable y (first rule), or as part of a "chain" of role atoms (rule 2). Similar observations can be made in case (4)(b). For case (4)(a), note that the precondition implies that u already is the target of direct connections from two distinct terms y and t. Thus, u must be a DL-safe variable, and the reduction is permissible, even though it clearly leads to multiple direct connections leading to u in rule 1.

Finally, we need to verify that the range restrictions of RB_4 are admissible for the (grounded) ELP rules of RB_4. It is easy to see that the transformations do not change the dependency between roles, but may introduce new role names during the decomposition. However, admissibility is only concerned with role atoms that lead to the final variable of a rule. The only case where newly introduced role atoms connect to the final variable is (3), and additional range restrictions are explicitly introduced there to ensure admissibility.

Claim 2 After the above translation, all rules in RB_4 have at most three variables in the body.

For a contradiction, suppose that there is some rule $B \to H$ with at least four variables in B. By assumption, none of the cases of the translation is applicable to that rule. However, there must be some reducible MCC S in B. Otherwise, B would contain no variables besides the root and final variable of H, contradicting our assumption. Thus let S be a reducible component in B. Since rule (1) is not applicable, all reducible MCCs of B (and in particular S) contain the final variable y.

Since S is reducible, some atom of S contains a variable that is neither final nor root for H. Since case (3) is not applicable, we conclude that there is no direct connection T from some variable $t \neq x$ to y. But since S is a connected component, all terms of S are connected to y, and hence there must be a direct connection Γ from y to some variable u. Since (2) does not apply, Γ must be such that $S_{\Gamma,u}$ contains the root variable x given that it cannot contain y without violating the tree shape of the rule. Since only one such Γ can exist (again due to the tree shape asserted for extended DL rules), and since $B \to H$ contains more than three variables by assumption, $S_{\Gamma,u}$ must be reducible, and thus the precondition of case (4) holds.

It remains to show that one of the two sub-cases of (4) must apply. Assuming that (a) does not hold, we conclude that there is no direct connection from any term $t \neq y$ to u. We know that u is directly connected with some term other than y, since $S_{\Gamma,u}$ is reducible. Therefore there is some connection Γ' from u to some term u'. Since (2) is not applicable, $S_{\Gamma',u'}$ contains x, and (b) is indeed applicable.

Claim 3 The transformation terminates after a finite number of steps that is polynomially bounded in the size of RB_3.

For any set S of atoms, let $\nu(S)$ be the number of variables in S. Given a rule $B \rightarrow H \in \mathrm{RB}_3$, a number $\gamma(B \rightarrow H)$, called the *reduction number* of $B \rightarrow H$, is then defined by setting $\gamma(B \rightarrow H) := \max(0, \nu(B \cup H) - 3)$. Moreover, $\gamma(\mathrm{RB}_3)$ is defined as the sum of $\gamma(B \rightarrow H)$ for all $B \rightarrow H \in \mathrm{RB}_3$. Clearly, $\gamma(\mathrm{RB}_3)$ is polynomially bounded by the size of RB_3.

We claim that the above transformation terminates after at most $\gamma(\mathrm{RB}_3)$ steps. Clearly, no transformation can be applied if $\gamma(\mathrm{RB}_3) = 0$. It remains to show that, whenever RB_3' is obtained from RB_3 by any of the transformation steps, we find that $\gamma(\mathrm{RB}_3) > \gamma(\mathrm{RB}_3')$. This is achieved by considering all transformations individually. The technical difficulty in this part arises from the individual $\max(\cdot)$ computations involved in γ: even if a rule gets smaller, this might not equally reduce its reduction number, since there are no negative reduction numbers. In other words, each rule may contain up to three variables that do not count. We will sometimes assume that those three have been selected for some rule and speak of “non-counting variables” and “counting variables.”

For case (1), note that S contains some variable that does not occur in H, and that $B \rightarrow H$ has at least 4 variables. We may thus assume that S contains a counting variable. Therefore rule 1 has at least one counting variable less than $B \rightarrow H$. If $\nu(S) \leq 3$, then rules 2 and 3 have a reduction number of 0 and the claim follows. If $\nu(S) > 3$ then we may assume that S contains at most two non-counting variables of B, since $B \rightarrow H$ also contains some variable y final for H that is not contained in S. Hence rule 1 has at least $\nu(S) - 2$ counting variables less. Rule 3 in turn has only $\nu(S) - 3$ counting variables, and rule 2 still has no counting variables, so that the claim follows again.

For case (2), we use n to denote $\nu(S_{\Gamma,u} \cup \{\top(u)\})$, the number of variables in $S_{T,u} \cup \Gamma$ that are distinct from t. Since $S_{\Gamma,u}$ is reducible, $n \geq 1$. Again, since there are 4 or more variables in $B \rightarrow H$, we can assume that $S_{\Gamma,u}$ contains at least one variable that is counting in $B \rightarrow H$. The reduction number of rule 1 therefore is strictly smaller than $\gamma(B \rightarrow H)$, and this suffices whenever $n \leq 3$ (since the reduction number of rule 2 is 0 in that case). Now assume that $n > 3$. Since t can be assumed to be non-counting, $S_{\Gamma,u} \cup \{\top(u)\}$ contains at most 2 non-counting variables of B, and hence rule 1 has at least $n - 2$ counting variables less. Rule 2, in turn, has only $n - 3$ non-counting variables, which again proves the overall reduction.

Case (3) can be shown by a similar argumentation. Rule 2 does not add to the overall reduction number, and the sum of rules 1 and 3 is found to decrease by a case distinction as above. Case (4)(a) is also similar where we note that $t \notin \{x, y\}$ is strictly required to obtain a reduction. For case (4)(b), the result follows since u is assumed to be a variable, so that again the reduction number of the transformed rule 1 decreases (while the other rule has at most three variables).

Claim 4 RB_4 semantically emulates RB_3.

This can be shown by a simple induction, given that all possible transformation steps preserve semantic emulation. This is generally rather easy to see, but we show one case formally for illustration. Thus consider transformation step (1), where $B \to H$ is the considered rule, and $B_1 \to H$, $B_2 \to X(x)$, and $B_3 \to Y(t)$ denote the generated rules.

For the one direction, consider some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \{B_1 \to H, B_2 \to X(x), B_3 \to Y(t)\}$. We claim that $\mathcal{I} \models B \to H$. Thus assume that $\mathcal{I}, \mathcal{Z} \models B$ for some variable assignment $\mathcal{Z}$. Then also $\mathcal{I}, \mathcal{Z} \models B_3$ as $B_3 \subseteq B$, and hence $\mathcal{I}, \mathcal{Z} \models Y(t)$. But then $\mathcal{I}, \mathcal{Z} \models B_2$ and hence $\mathcal{I}, \mathcal{Z} \models X(x)$. This in turn shows that $\mathcal{I}, \mathcal{Z} \models B_1$ and thus $\mathcal{I}, \mathcal{Z} \models H$ as required.

For the other direction, consider some interpretation $\mathcal{I}$ such that $\mathcal{I} \models B \to H$. Then there is some interpretation $\mathcal{I}'$ with $\mathcal{I}' \models B \to H$, and such that $Y^{\mathcal{I}'} = \{\delta \in \Delta^{\mathcal{I}'} \mid \mathcal{I}', \mathcal{Z} \models B_3$ for some variable assignment $\mathcal{Z}$ with $t^{\mathcal{I}', \mathcal{Z}} = \delta\}$ and $X^{\mathcal{I}'} = \{\delta \in \Delta^{\mathcal{I}'} \mid Y^{\mathcal{I}'} \neq \emptyset\}$. A suitable $\mathcal{I}'$ can be obtained from $\mathcal{I}$ by minimising the extent of X and Y while preserving all other aspects of the interpretation, which can be done since X, Y are fresh. Note that $\mathcal{I}' \models B_3 \to Y(t)$ and $\mathcal{I}' \models B_2 \to X(x)$ by definition. We claim that $\mathcal{I}' \models B_1 \to H$. Thus assume that $\mathcal{I}', \mathcal{Z} \models B_1$ for some variable assignment $\mathcal{Z}$. Then $\mathcal{I}', \mathcal{Z} \models X(x)$. By the definition of $X^{\mathcal{I}'}$ and $Y^{\mathcal{I}'}$, we find that there is some variable assignment $\mathcal{Z}'$ such that $\mathcal{I}', \mathcal{Z}' \models B_3$. By construction, B_3 and B_1 contain no common variables. Thus there is some variable assignment $\mathcal{Z}''$ such that $\mathcal{Z}''(x) = \mathcal{Z}(x)$ for any variable x in B_1 and $\mathcal{Z}''(x) = \mathcal{Z}'(x)$ for any variable x in B_3. But then $\mathcal{I}', \mathcal{Z}'' \models B_1 \cup B_3$. As defined in (1), $(B_1 \cup B_2) \supseteq B$ and thus $\mathcal{I}', \mathcal{Z}'' \models B$, and we can conclude $\mathcal{I}', \mathcal{Z}'' \models H$ since $\mathcal{I}' \models B \to H$. By definition, $\mathcal{Z}$ and $\mathcal{Z}''$ agree on all terms in H and thus we obtain $\mathcal{I}', \mathcal{Z} \models H$ as required. Since $\mathcal{Z}$ was arbitrary, this shows that $\mathcal{I} \models B_1 \to H$ as required.

The cases (2)–(4) can be treated in a similar fashion.

Summing up, we find that RB_4 semantically emulates RB, can be computed in polynomially many steps w.r.t. the size of RB, and contains at most three variables per rule. To obtain the required rule base RB', we replace concept atoms of the form $\exists R.\mathsf{Self}(x)$ and $\exists R.\{a\}(x)$ by role atoms $R(x, x)$ and $R(x, a)$, respectively. Note that this suffices to establish the form of Proposition 8.5.2, and that this operation does not introduce additional variables. □

We can combine the previous results to obtain the desired complexity result.

Theorem 9.4.4 *The problem of deciding satisfiability of an extended* ELP *rule base is* P*-complete w.r.t. the size of the rule bases.*

Proof. By Lemma 9.4.3, every extended ELP rule base RB is semantically emulated by an extended ELP rule base RB′ with at most three variables per rule. Thus, the size of ground(RB′) is polynomial w.r.t. the size of RB, and it is equisatisfiable to RB′ by Proposition 9.2.4. Using the construction in Definition 8.5.3, a datalog program P(ground(RB′)) is obtained that also has at most three variables per rule. By Theorem 8.5.6, P(ground(RB′)) is satisfiable iff ground(RB′) is. The result follows since satisfiability of datalog programs with at most three variables per rule can be decided in polynomial time (Fact 4.1.4), combined with the fact that all of the relevant transformations are polynomial. □

9.5 Summary

In this chapter, we have introduced the notion of DL-safe variables as a basis for combining the established formalism of DL-safe rules with the new approaches on DL Rules as discussed in Chapter 8. The resulting formalism of DL+safe rules lead to a new class of decidable fragments of SWRL that generalise both DL-safe rules and DL Rules. It could be shown that satisfiability checking is decidable in all DL+safe rule languages that are based on a description logic for which knowledge base satisfiability is decidable.

The decidability proof for DL+safe rules is based on the *grounding* of DL-safe variables, which leads to an equisatisfiable but exponentially large set of DL Rules. Our further investigations have shown that this exponential blow-up may not lead to a corresponding increase in worst-case complexity of reasoning. Indeed, the worst-case complexity in the case of $\mathcal{SROIQ}$+safe rules was found to be the same as for $\mathcal{SROIQ}$ since the additional ground rules did not increase the *depth* of the knowledge base. It is known that DL-safe rules do not increase the reasoning complexity of $\mathcal{SHIQ}$, and we therefore conjecture that a similar result could be obtained for $\mathcal{SHIQ}$+safe rules.

However, reasoning in DL+safe rule languages is necessarily intractable since they encompass DL-safe rules, and hence the extension of $\mathcal{SROEL}(\sqcap_s, \times)$ rules to $\mathcal{SROEL}(\sqcap_s, \times)$+safe rules does not preserve tractability. Yet, we were able to extend $\mathcal{SROEL}(\sqcap_s, \times)$ rules with DL-safe variables without loosing tractability. The resulting formalism ELP uses conditions that resemble the structural requirements for DL Rules in order to enforce an acyclic dependency structure between all variables in rule bodies. This contrasts with the earlier definition of DL+safe rules where DL-safe variables were treated like constant symbols that are hardly affected by the structural restrictions that are imposed on a rule.[4]

All of our proofs eventually used grounding for reducing a DL+safe rule base to a DL rule base. While convenient for obtaining complexity results, this method may not be most adequate for practical implementations, even if it is deferred as in the case of ELP until all rules have been decomposed to limit the resulting increase in the size of the rule base. Since ELP can be transformed to equisatisfiable datalog, it might be more promising to simply keep DL-safe variables, together with non-DL atoms of the form $O(x)$ to restrict their possible values. It is very likely that optimised datalog engines will typically show better performance on such inputs than on the corresponding grounding. Indeed, grounding can still be performed by the datalog engine if considered suitable, whereas ground rules can hardly be generalised again to obtain a more compact representation.

An interesting perspective on DL-safe variables is to view them as "variable nominals" that represent one of a finite number of nominal classes. In contrast to disjunctions of nominals, the value that is chosen for DL-safe variables must be the same in all occurrences of this variable. Based on the observation that grounding leads to a highly regular knowledge base, one might conjecture that such variable nominals could be processed more efficiently when introducing a suitable DL construct that allows DL-safe variables to be expressed more naturally in terms of DL axioms. The study of according extensions of existing inferencing algorithms is an interesting area of future research.

9.6 Related Work

DL-safe rules have originally been proposed in [MSS05], where they were obtained as a natural extension of the resolution-based *KAON2* algorithm for translating $\mathcal{SHIQ}$ knowledge bases to equisatisfiable datalog programs [Mot06]. Reasoning support for DL-safe rules is currently available in the original KAON2 system [MS06] and in Pellet [KPS06, SPG+07]. In addition, OWL 2 introduces a simple data integration mechanism based on *keys* that allows reasoners to infer the identity of two individuals whenever they share the same values on certain roles (in OWL: "properties") [HKP+09]. Since OWL 2 keys are only applicable to named individuals, they are closely related to DL-safe rules, and indeed every OWL 2 key axiom is equivalent to a (slightly modified kind of) DL-safe rule with an equality statement as its head [MPSP09].

[4]Cases where the presence of constants is the reason why a SWRL rule is not a DL Rule can only occur in DLs without inverses. For example, $R(a, x) \to C(x)$ is not an $\mathcal{EL}$ rule.

9.6 Related Work

An earlier approach for combining description logics with datalog was $\mathcal{AL}$-log [DLNS98]. This hybrid approach restricts the interaction of datalog and description logics by restricting to unary DL atoms, disallowing DL atoms in rule heads, and requiring all rules to be DL-safe (though this term has not been used there). DL atoms are thus considered as constraints that additionally restrict the applicability of datalog rules, which in turn can be considered to operate “on top” of a given DL knowledge base. The DL considered in [DLNS98] is $\mathcal{ALC}$, so the $\mathcal{SHIQ}$-based DL-safe rules of [MSS05] can be viewed as a proper extension of $\mathcal{AL}$-log.

A more general perspective is provided in [LR98] where the CARIN family of knowledge representation languages is considered. These formalisms are generally based on a combination of datalog and the description logic $\mathcal{ALCNR}$, which we prefer to call $\mathcal{ALCN}(\sqcap)$ according to the nomenclature of Chapter 5. The approach allows both role and concept atoms in rules but it disallows DL atoms in rule heads. Besides the fact that the UNA is adopted for constant symbols, the semantics of CARIN agrees with the semantics of SWRL. It is shown that conjunctive query answering is decidable for $\mathcal{ALCN}(\sqcap)$, from which decidability of non-recursive CARIN rule bases can be derived. Based on advances in conjunctive querying, this form of non-recursive CARIN has recently been extended to more expressive DLs [Ort08].

Since reasoning tasks become undecidable in unrestricted recursive rule bases, two types of restrictions are studied in [LR98]. First of all, it is shown that decidability can be regained by restricting the expressive features of $\mathcal{ALCN}(\sqcap)$ in suitable ways. As a second approach that is closer related to this work, a notion of *role safety* is introduced, requiring that at least one of the variables in each role atom in the body of a rule is DL-safe in a strong sense: it is required to occur in a non-DL body atom the predicate of which does not occur in the head of any rule. It is easy to see that a SWRL rule that is role-safe in this sense is a DL+safe rule in any description logic that has inverse roles and that satisfies the basic conditions of Definition 8.4.2. For this to be true, all binary atoms that occur in the head of some rule need to be declared as roles, but otherwise the result is straightforward. Role safety precludes the occurrence of role chains, yet chains may be introduced when connecting the rule body with the universal role U. To prevent that the resulting rule base formally violates regularity restrictions, the DL should also support role conjunction and concept products, so that all roles can be declared simple and rules can be simplified as in Fig. 8.4. This allows us to conclude that $\mathcal{ALCNI}(\sqcap,\times)$+safe rules subsume role-safe CARIN-$\mathcal{ALCN}(\sqcap)$, and in particular that the decidability of the latter is a corollary of the results of Chapter 5, 8, and 9.

Another generalisation of DL-safe rules and $\mathcal{AL}$-log is provided by the framework of $\mathcal{DL}$*+log* when considered under its first-order semantics [Ros06]. This approach encompasses datalog with disjunctions and negations, and imposes a weaker requirement for DL-safety that requires only the variables that occur in rule heads to be DL-safe. On the one hand, this is more restrictive than DL+safe rules which also allow variables that are not DL-safe in the rule heads. On the other hand, $\mathcal{DL}$+*log* is more general than DL+safe rules since it allows rule bodies to contain non-tree-shaped dependencies between variables that are not DL-safe, as long as those variables do not occur in rule heads. Clearly, reasoning in $\mathcal{DL}$+*log* subsumes some forms of conjunctive query answering for the underlying DL, and indeed it was shown in [Ros06] that satisfiability of $\mathcal{DL}$+*log* rule bases is decidable iff the containment problem for (unions of) conjunctive queries is decidable for the underlying description logic. This is a significantly stronger requirement than the one that has been given in Proposition 9.2.4 for ensuring decidability of DL+safe rules.

Chapter 10

Conclusions

The objective of this work was to advance the development of hybrid knowledge representation formalisms that combine aspects of rules and description logics. We conclude by summing up the results that have been accomplished toward that goal (Section 10.1), and by discussing their significance for applied and foundational research (Section 10.2). Finally, we give an extended overview of future research questions that arise from our work (Section 10.3).

10.1 Summary of the Results

To summarise and discuss the results of this work, we refer to the three main goals as specified in Section 1.4.

10.1.1 Decidable Fragments of SWRL

In Chapter 8, we have introduced *DL Rules* as a new family of decidable SWRL fragments. The defining feature of DL Rules is that they can be semantically emulated by knowledge bases of an underlying description logic, and that the computation of these knowledge bases is possible in polynomial time (actually even in logarithmic space). Although computationally simple, however, the required translation is not necessarily obvious since it combines expressive features of $\mathcal{SROIQ}$ in a rather unusual way. Moreover, the resulting DL representations of SWRL rules involve multiple auxiliary axioms that are harder to manipulate and maintain than the original rule. Thus, even though DL Rules can only express logical sentences that could also be captured by DL knowledge bases, the rule-based perspective is arguably more adequate for modelling certain kinds of information.

We have not made any attempt to arrive at maximal (in any sense) DL Rules languages herein. Based on our experiences in maximising DLP in Chapter 7, we expect any such attempt to lead to prohibitively complex syntactic descriptions due to the intricate interplay of various DL features. Therefore, the definition of DL Rules was rather designed to allow for an easy generalisation to a large class of description logics, allowing us to transfer numerous complexity results from DL to fragments of SWRL. This approach also encompasses DLs with additional role constructors as studied in Chapter 5 which are of natural interest when studying SWRL. As shown in this chapter, certain logical operations on roles can be allowed without increasing the worst-case complexity of reasoning, thus providing interesting extensions of DLs in their own right. The use of these operations in Chapter 8 illustrates that especially conjunctions of simple roles and (simple or non-simple) concept products allow a DL to express more SWRL rules.

In addition to their utility for providing a rule-based view on description logics, DL Rules constitute a powerful vehicle for re-using decidability and complexity results that have been established for DLs. This has been illustrated in Chapter 9 where *DL+safe rules* have been defined as a new class of decidable SWRL fragments that extend both DL Rules and the known class of DL-safe rules [MSS05]. This was established by introducing DL-safe variables that, in effect, can assume only values that are represented by some individual name. In this sense, DL+safe rules are compact representations of the DL Rules obtained by grounding DL-safe variables. Yet, it can be argued that they truly extend the expressivity of DLs since this grounding leads to an exponential number of rules.

A major insight of this approach was that DL-safe rules can be considered as an abbreviation for an exponential number of ground DL Rules. The same is true for recursive role-safe CARIN [LR98], which has hitherto been incomparable to DL-safe rules and other approaches [Mot06, Ros06]. DL+safe rules thus provide a common conceptual framework for DL Rules, DL-safe rules, and role-safe CARIN. Moreover, DL+safe rules can easily be further extended to accommodate future extensions of DL expressiveness by adopting the modular definition of DL Rules as illustrated for the case of role constructors in Chapter 8. For example, conjunctions of non-simple roles, regular expressions on roles, or other new modelling primitives as in [TSS09] could be exploited. In this way, one could even try to obtain (monotonic) $\mathcal{DL}$*+log* as a special case [Ros06].

While the exponential grounding of DL+safe rules may incur an exponential increase in reasoning complexity, we have shown that this is not the case for $\mathcal{SROIQ}$+safe rules. For tractable rule languages such as $\mathcal{SROEL}(\sqcap_s, \times)$+safe rules, however, the DL-safe component does lead to higher complexities, and the tractable formalism ELP has been introduced as a response.

10.1.2 Rule Fragments of Description Logics

The study of DL fragments that share some properties with first-order Horn logic has mainly been conducted in Chapter 6 and 7. The former is based on the definition of Horn-$\mathcal{SHIQ}$ which it generalised to arbitrary fragments of $\mathcal{SROIQ}$. While Horn-$\mathcal{SROIQ}$ as such is not studied, all of its features occur in some of the Horn DLs that are. The main results of Chapter 6 beyond the general definition of Horn DLs are a number of complexity results for various Horn description logics. It has been well-known that reasoning in all fragments of Horn-$\mathcal{SHIQ}$ can be achieved in time polynomial in the number of atomic ABox axioms (data complexity), but no results on combined complexities had been established yet.

The main conclusion of these complexity studies is that reasoning in Horn logics becomes intractable even for very simple DLs: PSpace-complete for all DLs between Horn-$\mathcal{FL}^-$ and Horn-$\mathcal{FLOH}^-$, and ExpTime-complete for all DLs between Horn-$\mathcal{FLE}$ and Horn-$\mathcal{SHIQ}$. This might indicate a slight decrease for Horn-$\mathcal{FL}^-$ since reasoning for $\mathcal{FL}^-$ is ExpTime-complete, but overall these intractabilities mostly serve to complete our understanding of Horn DLs rather than hinting at practically useful DL fragments. Another important result of Chapter 6 is in the proofs themselves which establish intractability in a direct way while using only very little expressive features. Theorem 7.2.7 (page 122) gives an example of how these techniques can be re-used to establish proofs in other contexts.

In Chapter 7, we have focussed on the study of the principal relationship between DL and datalog, seeking a maximal fragment of $\mathcal{SROIQ}$ that can be semantically emulated in datalog. A first contribution has been to define this task in a rigorous way, using the new notion of *structurality* to ensure that the problem can have a solution. We have then explicitly defined $\mathcal{DLP}$ as the maximal fragment of $\mathcal{SROIQ}$ that satisfies our design principles, and shown that (1) it can indeed be expressed in datalog, and (2) no larger DL has this property. The encodings required for (1) have been surprisingly intricate – this was also reflected in the definition of $\mathcal{DLP}$ –, but the most complex proof was required for showing maximality (2).

In conclusion, the result of Chapter 7 is not so much the (necessarily complex) definition of a maximal DLP language, but rather the development and application of proof techniques for establishing such results at all. Another conclusion of this work is that the syntactic complexity of DL can impose a real barrier for relating it to other logical formalisms. Nevertheless, the complexity of $\mathcal{DLP}$ can not be attributed to this characteristic of DL only; rather it also reflects the fundamental difference between the paradigms of Horn logic and description logic. In this sense, Chapter 7 also truly increases our understanding of the relationship between these formalisms.

Finally, it should be noted that a major difference between Horn DLs and DLP is that the former, in essence, refers to first-order Horn logic *with* function symbols, while the latter excludes function symbols. This is apparent from the fact that Horn DLs do not restrict the use of existential quantifiers, while DLP supports existentials only on the left-hand side of GCIs. Thus, while DLP in the sense of Chapter 7 appears as a Horn DL, it still belongs to a more specific class of logics for which stricter properties must hold. This is also reflected in the fact that the model-theoretic properties of datalog that are exploited in Chapter 7 are not obtained as special cases of well-known closure of first-order Horn sentences under *reduced products*, but require the use of sub-model constructions that are related to universal logic [CK90].

10.1.3 Tractable Knowledge Representation Languages

Our research on tractable knowledge representation formalisms has led to positive and – just as important – negative results. The latter includes new intractability results for logics for which one might have hoped for polynomial-time inferencing procedures. Such results have specifically been obtained in the framework of Horn DLs as studied in Chapter 6. Whereas the original motivation for introducing Horn DLs was their reduced data complexity, we have shown that reasoning in Horn DLs is still intractable with respect to the overall size of the knowledge base, even when restricting to very small DLs such as Horn-$\mathcal{FL}^-$. The only exception is Horn-$\mathcal{FL}_0$ for which reasoning is possible in polynomial time, which is essentially a known result due to the close relationship of Horn-$\mathcal{FL}_0$ to DLP (in the sense of [Vol04]). Further intractability results have been established for Horn-$\mathcal{ELF}$, Horn-$\mathcal{FL}\circ^-$, and Horn-$\mathcal{FLI}^-$, all of which turn out to be ExpTime-hard.

Yet, the fact that no new tractable fragments could be discovered by studying Horn DLs does not indicate that Horn restrictions are not relevant in this context. Namely, we simply did not discover *new* tractable Horn DLs, but all known tractable DLs are also Horn in the sense of our definition.

Conversely, we have also obtained a number of positive results which established the tractability of new and extended formalisms. The first result of this kind is the tractability of the description logic $\mathcal{SROEL}(\sqcap_s, \times)$ in Chapter 5. This result is not unexpected, since $\mathcal{SROEL}(\sqcap_s, \times)$ is closely related to the tractable OWL EL profile of the OWL 2 standard [MCH+09], which it extends with conjunctions of simple roles and concept products on the left-hand side of RIAs. Yet, it seems that no proof of this tractability has been given in the literature yet, and – more importantly – no reasoning algorithm that specifically addresses this DL has been published.

The tractability result for $\mathcal{SROEL}(\sqcap_s, \times)$ depends on the correctness of a datalog reduction for $\mathcal{SROEL}(\sqcap_s, \times)$ rules that is given in Chapter 8. While the proof of this result is rather lengthy, the resulting datalog translation is indeed very easy and can be performed in logarithmic space. This line of research has been further extended in Chapter 9, where ELP was introduced as a tractable extension of $\mathcal{SROEL}(\sqcap_s, \times)$ rules with DL-safe variables. Tractability in this case is not obvious since direct grounding of DL-safe variables would lead to an exponential increase of the rule base, while standard rolling-up techniques are not applicable due to the lack of inverse roles in $\mathcal{SROEL}(\sqcap_s, \times)$.

Interestingly, ELP also accommodates the expressive power of two of the most important tractable DLs: $\mathcal{EL}^{++}$ (and our extension $\mathcal{SROEL}(\sqcap_s, \times)$) and DLP (and our extension $\mathcal{RL}$). Although it is known that the union of these DLs is intractable, ELP can still support all logical inferences of DLP knowledge bases by considering DLP axioms as DL-safe. Since ELP also subsumes $\mathcal{SROEL}(\sqcap_s, \times)$, we thus obtain a tractable formalism that supports all individual consequences of the two DLs, and some (but not all) consequences of their union.

10.2 Significance of the Results

The successful adoption of Semantic Web technologies in many areas of application leads to new challenges for the underlying knowledge representation formalisms. Description logics have traditionally played a major rôle in ontological modelling but they are faced with new challenges as the focus of applications shifts from schema information toward instance data. And indeed, recent years have seen a massive increase in the amount of data that is published in machine-readable formats on the Semantic Web – now often called the *Web of Data* – while large parts of this semantic information refer to instances.[1] Rule languages, e.g. from logic programming or deductive databases, can help to address these challenges, but their combination with DLs remains an open problem.

This work has addressed this practically relevant challenge by investigating combinations of DLs and rule languages that allow for a tight semantic integration in the framework of SWRL, with the goal of extending expressivity of DLs and of improving the interoperability between rule-based and DL-based models and tools. A significant contribution toward these goals was the identification of *DL Rules* in Chapter 8 as a new class of decidable SWRL fragments that provides an alternative to the known *DL-safe rules*. By combining both approaches in *DL+safe rules* in Chapter 9, we were able to reconcile a number of hitherto incomparable DL rule extensions within a single conceptual framework. The modular definition of DL+safe rules allows us to instantiate them for a broad class of DLs, and it highlights ways for incorporating possible future extensions. We therefore believe

[1]This trend is supported by the increased adoption of Semantic Web technologies in "Web 2.0" scenarios [AKTV08], e.g. in semantic wikis [KVV+07, KVV06], where structured data is exploited for knowledge management and syndication.

that DL+safe rules are an important contribution for understanding first-order DL rule extensions.

Standard reasoning tasks in many DLs, and therefore also in many DL+safe rule languages, have very high worst-case complexities. Another major contribution of this work therefore is to propose light-weight formalisms that allow for polynomial-time inferencing while still providing additional expressiveness for DLs. The peak of this development in this work is the hybrid DL rule language ELP that integrates the expressiveness of the OWL 2 profiles OWL EL and OWL RL within a single rule-based formalism. The practical significance of this insight is that it opens a way for supporting multiple OWL 2 profiles in a single system, in spite of the fact that the unrestricted union of these profiles would lead to a highly intractable ontology language.

This outcome of this work has influenced language design and ongoing tool development in ontology-driven applications. In particular, the new v2.0 revision of the *Web Service Modeling Language* WSML[2] bases its sublanguage WSML-DL on ELP, thus establishing basic interoperability both with OWL 2 and with other rule-based sub-languages of WSDL [BFH+09]. ELP arguably is also an attractive formalism for implementers since it allows a single implementation to support a number of ontology languages. This is reflected by the recent effort of researchers at Semantic Technology Institute Innsbruck to develop an ELP reasoner *ELLY*[3] based on the datalog engine *IRIS*[4] [BF08].

Another software project that is based on the algorithms for ELP is the *Orel* ontology management system developed at Karlsruhe Institute of Technology.[5] This system focusses on large-scale ontology management and inferencing using secondary storage such as an on-disk database instead of executing inferences in primary memory. The goal of such an approach is to increase the scalability of reasoning by reducing the memory requirements and exploring the use of mechanisms for distribution, optimisation, and parallelisation that exist for databases. At the time of this writing, Orel is a very recent prototype. Yet it is able to classify the large OWL EL ontology SNOMED CT using a standard MySQL storage backend.

These ongoing implementation efforts also take advantage of the datalog reduction that we have developed for ELP (and thus, in particular, for the DL underlying the OWL EL profile). This outcome illustrates that the increased interoperability between rules and ontologies that has been established in this work is not merely of interest for improving the capabilities of modelling languages, but that it also enables the re-use of tools and algorithms available in both fields.

[2] `http://www.wsmo.org/wsml/wsml-syntax`
[3] `http://elly.sourceforge.org`
[4] `http://www.iris-reasoner.org`
[5] `http://code.google.com/p/orel/`

Besides this practical impact of our work, we have also advanced the understanding of the elementary relationship of first-order Horn logic and description logics in general. The insight that DL Rules can indirectly be expressed in description logics is relevant for ontology engineering, but it also has a didactic dimension in explaining the "hidden" expressiveness of DLs. The latter aspect is exploited, e.g., in [HKR09] to provide a textbook introduction to rules in the context of Semantic Web technologies.

Nevertheless, many of our insights about the relationship of rules and DLs are more foundational in nature. In particular, this applies to our characterisation of $\mathcal{DLP}$ as a datalog-expressible fragment of description logics that is maximal in a concrete sense. The significance of these results is not so much the actual definition of this fragment – its grammatical structure is rather too complex to suggest direct practical usage – but the development of paradigms and methods for investigating (maximal) *syntactic* fragments that are characterised by *semantic* criteria. This work can also be considered in the context of Lindström-type model-theoretic characterisations of fragments of first-order logic, though our study adds an additional syntactic twist based on the new notion of name separation. Considering emulation instead of equivalence complicates matters further. Yet we are convinced that investigations of the relationship between knowledge representation formalisms should in general be based on variants of emulation or conservative extension, since such notions can capture the practical requirements of semantic interoperability in a more precise way.

10.3 Future Work

The results of this work can, in essence, be extended in two ways: by further advancing the theoretical insights about the investigated logics and logical fragments, and by focussing on the practical application of our results by developing optimised algorithms and software tools.

Various open questions on the theoretical side have already been discussed in the respective chapters. From our point of view, the following research questions are specifically interesting:

- How can DL+safe rules be further generalised and extended? An obvious path for doing so is the use of additional expressive features of DLs to encompass more SWRL rules by extended normalisations. More interesting, however, is the question how new decidability results can be obtained based on the original rule form of SWRL, for example by making connections to decidable fragments of first-order Horn logic. In this context, one could also incorporate structural properties that have been studied for logic programs, e.g. *stratification*, *linearity*, or the *polynomial fringe property* [DEGV01].

- How can regularity and simplicity restrictions on DLs be weakened while preserving decidability and implementability? Both types of structural restrictions directly affect the admissibility of DL rule bases, and we have already presented some measures to overcome problems related to simplicity by means of concept products in Chapter 8. Another related question is how to weaken the simplicity restrictions on roles in role conjunctions so as to further extend the results of [GK08]. Relevant contributions for weakening regularity have been made in [Kaz09b].
- How can universal function-free first-order Horn logic be characterised by model-theoretic properties? We are aware of an according result for Horn logic with function symbols, but not of any such work on datalog. Chapter 7 provides certain necessary conditions that turned out to be sufficient for the "datalog fragment" of $\mathcal{SROIQ}$, but the complex constructions that were required to show this for the relevant cases do not allow for an easy generalisation to arbitrary first-order logic formulae with the respective model-theoretic properties.
- What is the "intersection" of other interesting fragments of first-order logic? Chapter 7 showed that "intersection" is rather not an appropriate term since the question of expressibility of one logic in terms of another is not symmetric but depends on the direction of this embedding. Yet, determining maximal structural, modular sub-logics that can be semantically emulated in some other formalism can be a worthwhile endeavour, especially if the related logics do not have the unusual syntactic complexity that DLs have. Candidates of such fragments include Guarded Fragments [AvBN98], modal logics [BvBW06], or the two-variable fragment with counting quantifiers C^2 [PH05].

Further questions could of course be raised, but the above are most directly related to the research reported herein, while being significant and complex enough to provide a basis for independent research efforts.

Regarding the practical application of our results, we have already mentioned ongoing implementation efforts for ELP in the previous section. Further efforts are required, however, to support the adoption of rule-based DL extensions in applied contexts. At least three different topic areas have to be addressed in this respect:

(1) inference engines and rule base management systems,

(2) rule editors and rule-enabled ontology engineering environments,

(3) establishing standards for serialising and interchanging rule bases.

The aforementioned ELLY and Orel reasoners aim at item (1) by exploiting the datalog reduction. Various optimisations are essential to achieve efficient processing in practice. Besides well-known optimisation techniques for datalog, such as magic sets (see, e.g., [AHV94]) or incremental materialisation (see, e.g., [GM99]), it is also necessary to apply optimisations to equality reasoning and DL-safety. For clarity, we have used a general-purpose axiomatisation of equality, but the inspection of the datalog programs obtained for ELP reveals that only specific inferences need to be computed from equality statements in this case, so that the use of a simplified equality theory would be feasible (see [Mot06] for a related discussion). Regarding DL-safety, it is clearly desirable to directly take DL-safe variables into account, e.g. when computing unifications for applying datalog rules, instead of using a general purpose algorithm that considers all possible instantiations even for DL-safe variables.

Reasoning with other types of DL+safe rules could be based on existing implementations of DL inference engines, which would require suitable extensions of their current algorithms for that purpose. Directly using rules internally promises better performance than translating rules to auxiliary knowledge bases, since the axioms of the latter admit more unintended interpretations than the original form. Managing rules is already required for handling DL-safe rules and has been addressed by various commonly used APIs such as the *KAON2 API* [Mot06] and the popular *OWL API* [HBN07, GHPPS09]. Further research is needed, however, to develop and evaluate suitable implementation techniques for handling DL+safe rules efficiently. Recent works have shown that some rule-like features can be addressed in tableaux algorithms in a more direct fashion [TSS09]. Another promising approach is to integrate the handling of DL-safe variables into inference algorithms, so as to avoid the unnecessary computation of ground rules.

Item (2) above is essential to enable the creation of rule-based data models in the first place. The integration of rule modelling into ontology editing environments have been attempted previously, but more work is needed to establish this modelling paradigm in application areas. A prototypical plug-in for graphically editing DL rules in the ontology editor Protégé [KFNM04] has been presented in [GH08]. Another related approach is pursued in the development of the NeOn Toolkit which can be used for creating OWL ontologies as well as F-Logic rule bases [HLS+08]. More work is required to establish a tight integration of OWL and rules in these cases, but the existing implementations indicate the feasibility of and potential demand for such approaches.

Item (3) is closely related to both of the other aspects, and may even be the essential component for connecting editors and reasoners, or – in other terms – creators and users of DL rule bases. Two main approaches provide promising foundations for exchanging rules: the SWRL proposal and its recent extensions, and the *Rule Interchange Format* (RIF) developed at W3C. SWRL is the syntactic form that is most widely used and supported in DL-based applications today, e.g. in Pellet [SPG+07] or KAON2 [MS06]. Further extensions and alternative serialisations have recently been proposed for a better integration of SWRL with

OWL 2 and related tools [GHPPS09]. RIF, in contrast, takes a more rule-centric perspective but includes a specification for combining RIF rules with OWL ontologies that semantically resembles SWRL while using the different RIF syntax [dB09]. Future work is needed in both cases to elaborate and explore these approaches in application scenarios, since it is not clear yet which exchange syntax for SWRL-like rule bases will be used in the future.

In summary, this work opens up a wide range of possible research directions both on the applied and on the foundational side. The separation of both aspects in the above discussion should not be misunderstood: we are convinced that the fruitful interplay of theory and practice is vital for ensuring the healthy future of this field of research.

Bibliography

[ACN+07] Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors. *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, volume 4825 of *LNCS*. Springer, 2007.

[AdR02] Carlos Areces and Maarten de Rijke. From description to hybrid logics, and back. In Wolter et al. [WWdRZ02], pages 17–36.

[AHV94] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1994.

[AKTV08] Anupriya Ankolekar, Markus Krötzsch, Duc Thanh Tran, and Denny Vrandečić. The two cultures: mashing up Web 2.0 and the Semantic Web. *Journal of Web Semantics*, 6(1), 2008.

[AvBN98] Hajnal Andréka, Johan F. A. K. van Benthem, and István Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[Baa03] Franz Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 325–330. Morgan Kaufmann, 2003.

[BB08] Dave Beckett and Jeen Broekstra, editors. *SPARQL Query Results XML Format*. W3C Recommendation, 15 January 2008. Available at `http://www.w3.org/TR/rdf-sparql-XMLres/`.

[BBH+90] Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, and Hans-Jürgen Profitlich. Terminological knowledge representation: A proposal for a terminological logic. Technical memo, DFKI GmbH, Saarbrücken, Germany, 1990.

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Kaelbling and Saffiotti [KS05], pages 364–369.

[BBL08] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In Clark and Patel-Schneider [CPS08].

[BBMR89] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori A. Resnik. CLASSIC: a structural data model for objects. In James Clifford, Bruce G. Lindsay, and David Maier, editors, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, volume 18 of *SIGMOD Record*, pages 59–67, 1989.

[BCM+07] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[BDS93] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[Bec04] Dave Beckett, editor. *RDF/XML Syntax Specification (Revised)*. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-syntax-grammar/`.

[BF08] Barry Bishop and Florian Fischer. IRIS – Integrated Rule Inference System. In Frank van Harmelen, Andreas Herzig, Pascal Hitzler, Zuoquan Lin, Ruzica Piskac, and Guilin Qi, editors, *Proceedings of the Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (ARea'08)*, volume 350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[BFH+09] Barry Bishop, Florian Fischer, Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Yiorgos Trimponias, and Gulay Unel. Defining the features of the WSML-DL v2.0 language. Deliverable, SOA4All, 2009. Available at `http://www.soa4all.eu/file-upload.html?func=startdown&id=82`.

[BG98] Leo Bachmair and Harald Ganzinger. Equational reasoning in saturation-based theorem proving. In Wolfgang Bibel and Peter H.

Schmitt, editors, *Automated Deduction – A Basis for Applications*, volume I, chapter 11, pages 353–397. Kluwer, 1998.

[BG04] Dan Brickley and Ramanathan V. Guha, editors. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-schema/`.

[BH91] Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In Harold Boley and Michael M. Richter, editors, *Proceedings of the Workshop on Processing Declarative Knowledge (PDK'91)*, volume 567 of *LNCS*, pages 67–86. Springer, 1991.

[BHK+09] Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynolds, editors. *RIF Core Dialect*. W3C Candidate Recommendation, 1 October 2009. Available at `http://www.w3.org/TR/rif-core/`.

[BK09] Harold Boley and Michael Kifer, editors. *RIF Basic Logic Dialect*. W3C Candidate Recommendation, 1 October 2009. Available at `http://www.w3.org/TR/rif-bld/`.

[BL84] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In Ronald J. Brachman, editor, *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI'84)*, pages 34–37. AAAI Press, 1984.

[BL08] Gerhard Brewka and Jérôme Lang, editors. *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*. AAAI Press, 2008.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, pages 96–101, May 2001.

[BLM08] Franz Baader, Carsten Lutz, and Boris Motik, editors. *Proceedings of the 21st International Workshop on Description Logics (DL'08)*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[Bor96] Alex Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[Bou09] Craig Boutilier, editor. *Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI'09)*. IJCAI, 2009.

[BPL85] Ronald J. Brachman, Victoria Pigman Gilbert, and Hector J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts in KRYPTON. In Aravind K. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI'85)*, pages 532–539. Morgan Kaufmann, 1985.

[BS85] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[BST07] Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logics*, 5(3):392–420, 2007.

[BT98] Patrick Blackburn and Miroslava Tzakova. Hybridizing concept languages. *Annals of Mathematics and Artificial Intelligence*, 24(1–4):23–49, 1998.

[BvBW06] Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier Science, 2006.

[CDA$^+$06] Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors. *Proceedings of the 5th International Semantic Web Conference (ISWC'06)*, volume 4273 of *LNCS*. Springer, 2006.

[CEO07] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 391–396. AAAI Press, 2007.

[CEO09] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In Boutilier [Bou09], pages 714–720.

[CFT08] Kendall G. Clark, Lee Feigenbaum, and Elias Torres, editors. *SPARQL Protocol for RDF*. W3C Recommendation, 15 January 2008. Available at http://www.w3.org/TR/rdf-sparql-protocol/.

[CGL$^+$07] Diego Calvanese, Guiseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[CHMS09] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors. *Proceedings of the 22nd International Workshop on Description Logics (DL'09)*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[CHPPS05] Bernardo Cuenca Grau, Ian Horrocks, Bijan Parsia, and Peter F. Patel-Schneider, editors. *Proceedings of the OWLED 2005 Workshop on OWL: Experiences and Directions*, volume 188 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[CK90] Chen Chung Chang and H. Jerome Keisler. *Model Theory*, volume 73 of *Studies in Logic and the Foundations of Mathematics*. North Holland, third edition, 1990.

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[CM03] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog: Using the ISO Standard.* Springer, 5th edition, 2003.

[CPS08] Kendall G. Clark and Peter F. Patel-Schneider, editors. *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, volume 496 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[Cyc02] Cycorp. *Ontological Engineer's Handbook, version 0.7*, 2002. Available online at http://www.cyc.com/doc/handbook/oe/oe-handbook-toc-opencyc.html, accessed Oct 2009.

[dB09] Jos de Bruijn, editor. *RIF RDF and OWL Compatibility*. W3C Candidate Recommendation, 1 October 2009. Available at http://www.w3.org/TR/rif-rdf-owl/.

[dCHS$^+$04] Sherri de Coronado, Margaret W. Haber, Nicholas Sioutos, Mark S. Tuttle, and Lawrence W. Wright. NCI Thesaurus: Using science-based terminology to integrate cancer research results. In Marius Fieschi, Enrico Coiera, and Yu-Chan Jack Li, editors, *Proceedings of the 11th World Congress on Medical Informatics (MEDINFO'04)*, pages 33–37. IOS Press, 2004.

[DEDC96] Pierre Deransart, AbdelAli Ed-Dbali, and Laurent Cervoni. *Prolog: The Standard. Reference Manual.* Springer, 1996.

[DEGV01] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[DLNS96] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language, and Information, pages 193–238. CLSI Publications, 1996.

[DLNS98] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: Integrating datalog and description logics. *Journal of Intelligent and Cooperative Information Systems*, 10(3):227–252, 1998.

[DMW06] Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors. *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06).* AAAI Press, 2006.

[DN05] Stéphane Demri and Hans Nivelle. Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language and Information*, 14(3):289–329, 2005.

[DR06] John Day-Richter, editor. *The OBO Flat File Format Specification, version 1.2*. The Gene Ontology Consortium, 2006. Available at `http://www.geneontology.org/GO.format.obo-1_2.shtml`.

[dSMPH09] Christian de Sainte Marie, Adrian Paschke, and Gary Hallmark, editors. *RIF Production Rule Dialect*. W3C Candidate Recommendation, 1 October 2009. Available at `http://www.w3.org/TR/rif-prd/`.

[EGOS08] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Simkus. Query answering in the description logic Horn-$\mathcal{SHIQ}$. In Hölldobler et al. [HLW08], pages 166–179.

[EIST05] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In Kaelbling and Saffiotti [KS05], pages 90–96.

[ELOS09] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Simkus. Query answering in description logics with transitive roles. In Boutilier [Bou09], pages 759–764.

[ELST04] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the Semantic Web. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR'04)*, pages 141–151. AAAI Press, 2004.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 2nd edition, 1996.

[FMC$^+$09] Peter Fox, Deborah McGuinness, Luca Cinquini, Patrick West, Jose Garcia, James L. Benedict, and Don Middleton. Ontology-supported scientific data frameworks: The virtual solar-terrestrial observatory experience. *Computers & Geosciences*, 35(4):724–738, 2009.

[For82] Charles Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[GB92] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.

[Gen00] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[GF92] Michael R. Genesereth and Richard E. Fikes. Knowledge Interchange Format, version 3.0 reference manual. Technical report, Computer Science Department, Stanford University, Stanford, CA, USA, 1992.

[GGPS03] Carole Goble, Chris Greenhalgh, Steve Pettifer, and Robert Stevens. Knowledge integration: *In Silico* experiments in bioinformatics. In Ian Foster and Carl Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, chapter 9. Morgan Kaufmann, second edition, 2003.

[GH08] Francis Gasse and Volker Haarslev. DLRule: A rule editor plug-in for Protégé. In Clark and Patel-Schneider [CPS08].

[GHPPS09] Birte Glimm, Matthew Horridge, Bijan Parsia, and Peter F. Patel-Schneider. A syntax for rules in OWL 2. In Patel-Schneider and Hoekstra [PSH09].

[GHS08] Birte Glimm, Ian Horrocks, and Ulrike Sattler. Unions of conjunctive queries in $\mathcal{SHOQ}$. In Brewka and Lang [BL08], pages 252–262.

[GHVD03] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: combining logic programs with description logic. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, pages 48–57. ACM, 2003.

[GK08] Birte Glimm and Yevgeny Kazakov. Role conjunctions in expressive description logics. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *Proceedings of the 15th International Conference on Logic for Programming and Automated Reasoning (LPAR'08)*, volume 5330 of *LNCS*, pages 391–405. Springer, 2008.

[GL94] Guiseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, volume 1, pages 205–212. AAAI Press, 1994.

[GLHS08] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. Answering conjunctive queries in the SHIQ description logic. *Journal of Artificial Intelligence Research*, 31:150–197, 2008.

[GM99] Ashish Gupta and Inderpal Singh Mumick, editors. *Materialized Views: Techniques, Implementations, and Applications*. MIT Press, 1999.

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

[Gog06] Joseph Goguen. Information integration in institutions. Draft of a chapter for Jon Barwise memorial volume, edited by Larry Moss, Indiana University Press; available at `http://cseweb.ucsd.edu/~goguen/pps/ifi04.pdf`, 2006.

[Goo05] John Goodwin. Experiences of using OWL at the Ordnance Survey. In Cuenca Grau et al. [CHPPS05].

[GR09] Birte Glimm and Sebastian Rudolph. Conjunctive query entailment: Decidable in spite of $\mathcal{O}$, $\mathcal{I}$, and $\mathcal{Q}$. In Cuenca Grau et al. [CHMS09].

[Grä98] Erich Grädel. Description logics and guarded fragments of first-order logic: A perspective for new description logics? In Enrico Franconi, Giuseppe De Giacomo, Robert M. MacGregor, Werner Nutt, and Christopher A. Welty, editors, *Proceedings of the 1998 International Workshop on Description Logics (DL'98)*, volume 11 of *CEUR Workshop Proceedings*. CEUR-WS.org, 1998.

[GSFA08] Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors. *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*. IOS Press, 2008.

[GSH08] Francis Gasse, Ulrike Sattler, and Volker Haarslev. Rewriting rules into $\mathcal{SROIQ}$ axioms. In Baader et al. [BLM08].

[GW97] Bernhard Ganter and Rudolph Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1997.

[GZB06] Christine Golbreich, Songmao Zhang, and Oliver Bodenreider. The foundational model of anatomy in OWL: Experience and perspectives. *Journal of Web Semantics*, 4(3):181–195, 2006.

[Hay04] Patrick Hayes, editor. *RDF Semantics*. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-mt/`.

[HBN07] Matthew Horridge, Sean Bechhofer, and Olaf Noppens. Igniting the OWL 1.1 touch paper: The OWL API. In Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, editors, *Proceedings* *of the OWLED 2007 Workshop on OWL: Experiences and Directions*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[HKP+09] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[HKR09] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[HKRS08] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web – Grundlagen*. eXamen.press. Springer, 2008.

[HKS05] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The irresistible $\mathcal{SRIQ}$. In Cuenca Grau et al. [CHPPS05].

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In Doherty et al. [DMW06], pages 57–67.

[HLS+08] Peter Haase, Holger Lewen, Rudi Studer, Duc Thanh Tran, Michael Erdmann, Mathieu d'Aquin, and Enrico Motta. The NeOn ontology engineering toolkit, 2008. Presentation at the Developers Track of the 17th International Conference on World Wide Web (WWW'08), synopsis available at `http://www.aifb.uni-karlsruhe.de/WBS/pha/publications/neon-toolkit.pdf`.

[HLW08] Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors. *Proceedings of the 11th European Conference on Logics in Artificial Intelligence (JELIA'08)*, volume 5293 of *LNAI*. Springer, 2008.

[HM01] Volker Haarslev and Ralf Möller. Racer system description. In Rajeev Gor, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR'01)*, volume 2083 of *LNCS*, pages 701–705. Springer, 2001.

[HMS05] Ulrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In Kaelbling and Saffiotti [KS05], pages 466–471.

[HPS04] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*, pages 723–731. ACM, 2004.

[HPS08] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in OWL. In Sheth et al. [SSD+08], pages 323–338.

[HPSB+04] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin N. Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language*. W3C Member Submission, 21 May 2004. Available at `http://www.w3.org/Submission/SWRL/`.

[HPSBT05] Ian Horrocks, Peter F. Patel-Schneider, Sean Bechhofer, and Dmitry Tsarkov. OWL Rules: A proposal and prototype implementation. *Journal of Web Semantics*, 3(1):23–40, 2005.

[HS04] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 160(1):79–104, 2004.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David A. McAllester, and Andrei Voronkov, editors, *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *LNCS*, pages 161–180. Springer, 1999.

[ISO07] ISO/IEC 24707:2007. *Information technology – Common Logic (CL): a framework for a family of logic-based languages*. International Organization for Standardization, 2007. Publicly available at `http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007%28E%29.zip`.

[JS08] Andrew G. James and Kent A. Spackman. Representation of disorders of the newborn infant by SNOMED CT. In Stig Kjær Andersen, Gunnar O. Klein, Stefan Schulz, and Jos Aarts, editors, *Proceedings of the 21st International Congress of the European Federation for Medical Informatics (MIE'08)*, pages 833–838, 2008.

[KAH08] Matthias Knorr, José J. Alferes, and Pascal Hitzler. A coherent well-founded model for hybrid MKNF knowledge bases. In Ghallab et al. [GSFA08], pages 99–103.

[Kal06] Aditya Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, University of Maryland College Park, USA, 2006.

[Kaz06] Yevgeny Kazakov. *Saturation-Based Decision Procedures for Extensions of the Guarded Fragment*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2006.

[Kaz08] Yevgeny Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In Brewka and Lang [BL08], pages 274–284.

[Kaz09a] Yevgeny Kazakov. Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies. In Boutilier [Bou09], pages 2040–2045.

[Kaz09b] Yevgeny Kazakov. An extension of regularity conditions for complex role inclusion axioms. In Cuenca Grau et al. [CHMS09].

[KBR86] Thomas S. Kaczmarek, Raymond Bates, and Gabriel Robins. Recent developments in NIKL. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI'86)*, volume 2, pages 978–985. Morgan Kaufmann, 1986.

[KC04] Graham Klyne and Jeremy J. Carroll, editors. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-concepts/`.

[Ken09] Robert E. Kent. System consequence. In Sebastian Rudolph, Frithjof Dau, and Sergei O. Kuznetsov, editors, *Proceedings of the 17th International Conference on Conceptual Structures (ICCS'09)*, volume 5662 of *Lecture Notes in Computer Science*, pages 201–218. Springer, 2009.

[KFNM04] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL Plugin: An open development environment for Semantic Web applications. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference (ISWC'04)*, volume 3298 of *LNCS*, pages 229–243. Springer, 2004.

[KG09] Markus Krötzsch and Bernhard Ganter. A brief introduction to formal concept analysis. In Pascal Hitzler and Henrik Schärfe, editors, *Conceptual Structures in Practice*, chapter 1. Chapman & Hall/CRC, 2009.

[KHVS06] Markus Krötzsch, Pascal Hitzler, Denny Vrandečić, and Michael Sintek. How to reason with OWL in a logic programming system. In *Proceedings of the 2nd International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML'06)*. IEEE Computer Society Press, 2006.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[KPS06] Vladimir Kolovski, Bijan Parsia, and Evren Sirin. Extending the $\mathcal{SHOIQ}(d)$ tableaux with DL-safe rules: First results. In Bijan Parsia, Ulrike Sattler, and David Toman, editors, *Proceedings of the 19th International Workshop on Description Logics (DL'06)*, volume 198 of *CEUR WS Proceedings*. CEUR-WS.org, 2006.

[KR07] Markus Krötzsch and Sebastian Rudolph. Conjunctive queries for $\mathcal{EL}$ with role composition. In Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris, editors, *Proceedings of the 20th International Workshop on Description Logics (DL'07)*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[KRH06] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. On the complexity of Horn description logics. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *Proceedings of the 2nd Workshop on OWL: Experiences and Directions*, volume 216 of *CEUR WS Proceedings*. CEUR-WS.org, 2006.

[KRH07a] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for Horn description logics. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 452–457. AAAI Press, 2007.

[KRH07b] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In Aberer et al. [ACN$^+$07], pages 310–323.

[KRH08a] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description logic rules. In Ghallab et al. [GSFA08], pages 80–84.

[KRH08b] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. ELP: Tractable rules for OWL 2. In Sheth et al. [SSD+08], pages 649–664.

[Krö10] Markus Krötzsch. Efficient inferencing for OWL EL. In Tomi Janhunen and Ilkka Niemelä, editors, *Proceedings of the 12th European Conference on Logics in Artificial Intelligence (JELIA'10)*, LNAI. Springer, 2010. To appear.

[KRS10] Markus Krötzsch, Sebastian Rudolph, and Peter H. Schmitt. On the semantic relationship between datalog and description logics. In *Proceedings of the 4th Interational Conference on Web Reasoning and Rule Systems (RR'10)*, LNCS. Springer, 2010. To appear.

[KS05] Leslie Pack Kaelbling and Alessandro Saffiotti, editors. *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. Professional Book Center, 2005.

[KVV06] Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic MediaWiki. In Cruz et al. [CDA+06], pages 935–942.

[KVV+07] Markus Krötzsch, Denny Vrandečić, Max Völkel, Heiko Haller, and Rudi Studer. Semantic Wikipedia. *Journal of Web Semantics*, 5(4):251–261, 2007.

[LG90] Doug Lenat and Ramanathan V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, 1990.

[Llo88] John W. Lloyd. *Foundations of Logic Programming*. Springer, 1988.

[LR98] Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in carin. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[LS02] Carsten Lutz and Ulrike Sattler. The complexity of reasoning with Boolean modal logics. In Wolter et al. [WWdRZ02], pages 329–348.

[LS05] Carsten Lutz and Ulrike Sattler. Description Logics. Lecture at the ICCL Summer School 2005, Dresden, Germany. Course material available at `http://www.computational-logic.org/content/events/iccl-ss-2005/`, 2005.

[Lut08] Carsten Lutz. The complexity of conjunctive query answering in expressive description logics. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR'08)*, number 5195 in LNAI, pages 179–193. Springer, 2008.

[LW05] Carsten Lutz and Dirk Walther. PDL with negation of atomic programs. *Journal of Applied Non-Classical Logics*, 15(2):189–213, 2005.

[LWW07] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In Veloso [Vel07], pages 453–458.

[MB87] Robert MacGregor and Raymond Bates. The LOOM knowledge representation language. Technical report, Information Sciences Institute, The University of Southern California, Los Angeles, CA, USA, 1987.

[MCH+09] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz, editors. *OWL 2 Web Ontology Language: Profiles*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-profiles/`.

[MHRS06] Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler. Can OWL and logic programming live together happily ever after? In Cruz et al. [CDA+06], pages 501–514.

[Min74] Marvin Minsky. A framework for representing knowledge. Artificial intelligence memo, A.I. Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[MM04] Frank Manola and Eric Miller, editors. *Resource Description Framework (RDF): Primer*. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-primer/`.

[Mot06] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe (TH), Germany, 2006.

[MPSC09] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau, editors. *OWL 2 Web Ontology Language: Direct Semantics*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-direct-semantics/`.

[MPSP09] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-syntax/`.

[MR07] Boris Motik and Riccardo Rosati. A faithful integration of description logics with logic programming. In Veloso [Vel07], pages 477–482.

[MS06] Boris Motik and Ulrike Sattler. A comparison of reasoning techniques for querying large description logic ABoxes. In Miki Hermann and Andrei Voronkov, editors, *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligencen, and Reasoning (LPAR'01)*, volume 4246 of *LNCS*, pages 227–241. Springer, 2006.

[MSH07] Boris Motik, Rob Shearer, and Ian Horrocks. Optimized reasoning in description logics using hypertableaux. In Frank Pfenning, editor, *Proceedings of the 21st Conference on Automated Deduction (CADE'07)*, volume 4603 of *LNAI*, pages 67–83. Springer, 2007.

[MSH08] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau reasoning for description logics. Submitted to a journal, available at `http://www.hermit-reasoner.com/publications/`, 2008.

[MSS05] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.

[ORS10] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning (KR'10)*, pages 269–279. AAAI Press, 2010.

[Ort08] Magdalena Ortiz. Extending CARIN to the description logics of the $\mathcal{SH}$ family. In Hölldobler et al. [HLW08], pages 324–337.

[OWL09] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[PH05] Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14:369–395, 2005.

[PS08] Eric Prud'hommeaux and Andy Seaborne, editors. *SPARQL Query Language for RDF*. W3C Recommendation, 15 January 2008. Available at `http://www.w3.org/TR/rdf-sparql-query/`.

[PSG+05] Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, Edna Ruckhaus, and Daniel Hewlett. Cautiously approaching SWRL. Preprint submitted to Elsevier Science, available at `http://www.mindswap.org/papers/CautiousSWRL.pdf`, 2005.

[PSH09] Peter F. Patel-Schneider and Rinke Hoekstra, editors. *Proceedings of the OWLED 2009 Workshop on OWL: Experiences and Directions*, volume 529 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, editors. *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[PSM09] Peter F. Patel-Schneider and Boris Motik, editors. *OWL 2 Web Ontology Language: Mapping to RDF Graphs*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-mapping-to-rdf/`.

[PSS93] Peter F. Patel-Schneider and Bill Swartout. Description-logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort. Available at `http://www.bell-labs.com/user/pfps/papers/krss-spec.ps`, 1993.

[QK90] Joachim Quantz and Carsten Kindermann. Implementation of the BACK-system version 4. KIT Report, Department of Computer Science, Technische Universität Berlin, Germany, 1990.

[Qui68] M. Ross Quillian. Semantic memory. In Marvin Minsky, editor, *Semantic Information Processing*, chapter 4, pages 227–270. MIT Press, 1968.

[RGG+94] Alan Rector, Also Gangemi, Elena Galeazzi, Andrzej J. Glowinski, and Aangelo Rossi-Mori. The GALEN CORE model schemata for anatomy: Towards a re-usable application-independent model of medical concepts. In Pedro Barahona, Mario Veloso, and Jeremy Bryant, editors, *Proceedings of the 12th International Congress of the European Federation for Medical Informatics (MIE'94)*, pages 229–233, 1994.

[RKH08a] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. All elephants are bigger than all mice. In Baader et al. [BLM08].

[RKH08b] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Cheap Boolean role constructors for description logics. In Hölldobler et al. [HLW08], pages 362–374.

[RKH08c] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Description logic reasoning with decision diagrams: Compiling $\mathcal{SHIQ}$ to disjunctive datalog. In Sheth et al. [SSD+08], pages 435–450.

[RKH08d] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Terminological reasoning in $\mathcal{SHIQ}$ with ordered binary decision diagrams. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 529–534. AAAI Press, 2008.

[RN03] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, second edition, 2003.

[Ros06] Riccardo Rosati. $\mathcal{DL}$*+log*: A tight integration of description logics and disjunctive datalog. In Doherty et al. [DMW06], pages 68–78.

[RP05] Robert G. Raskin and Michael J. Pan. Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). *Journal of Computers and Geosciences*, 31(9):1119–1125, 2005.

[Rud06] Sebastian Rudolph. *Relational Exploration – Combining Description Logics and Formal Concept Analysis for Knowledge Specification.* PhD thesis, Technische Universität Dresden, Germany, 2006.

[SAR$^+$07] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J. Mungall, The OBI Consortium, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H. Scheuermann, Nigam Shah, Patricia L. Whetzeland, and Suzanna Lewis. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25:1251–1255, 2007.

[SBLH06] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[SCC97] Kent A. Spackman, Keith E. Campbell, and Roger A. Côté. SNOMED RT: A reference terminology for health care. In Daniel R. Masys, editor, *Proceedings of the 1997 AMIA Annual Fall Symposium*, Journal of the American Medial Informatics Association, Symposium Supplement, pages 640–644. Hanley & Belfus, 1997.

[Sch91] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 466–471. Morgan Kaufmann, 1991.

[Sch94] Andrea Schaerf. Reasoning with individuals in concept languages. *Data Knowledge Engineering*, 13(2):141–176, 1994.

[Sch09a] Peter H. Schmitt. Notes on conservative extensions. Personal communication, 2009.

[Sch09b] Michael Schneider, editor. *OWL 2 Web Ontology Language: RDF-Based Semantics*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-rdf-based-semantics/`.

[Ser07] Baris Sertkaya. *Formal Concept Analysis Methods for Description Logics*. PhD thesis, Technische Universität Dresden, Germany, 2007.

[SGA07] Rudi Studer, Stephan Grimm, and Andreas Abecker, editors. *Semantic Web Services: Concepts, Technologies, and Applications*. Springer, 2007.

[SHKG09] Michael Smith, Ian Horrocks, Markus Krötzsch, and Birte Glimm, editors. *OWL 2 Web Ontology Language: Conformance*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-conformance/`.

[Sow00] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[SS89] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431. Morgan Kaufmann, 1989.

[SSD+08] Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors. *Proceedings of the 7th International Semantic Web Conference (ISWC'08)*, volume 5318 of *LNCS*. Springer, 2008.

[SSS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Journal of Artificial Intelligence*, 48:1–26, 1991.

[ST07] Renate A. Schmidt and Dmitry Tishkovsky. Using tableau to decide expressive description logics with role negation. In Aberer et al. [ACN+07], pages 438–451.

[SWE] Jet Propulsion Laboratory, California Institute of Technology. *Semantic Web for Earth and Environmental Terminology (SWEET)*. `http://sweet.jpl.nasa.gov/`, accessed Oct 2009.

[Tes01] Sergio Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, United Kingdom, 2001.

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *LNCS*, pages 292–297. Springer, 2006.

[Tob01] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.

[TSS09] Dmitry Tsarkov, Ulrike Sattler, and Robert Stevens. A solution for the Man-Man Problem in the family history knowledge base. In Patel-Schneider and Hoekstra [PSH09].

[Tur37] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1937.

[UD07] Mathias Uslar and Nikolai Dahlem. Semantic web technologies for power grid management. In Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, and Marc Ronthaler, editors, *Informatik 2007: Informatik trifft Logistik, Beiträge der 37. Jahrestagung der Gesellschaft für Informatik, Bremen, Germany*, volume 1 of *GI Proceedings 109*, pages 242–246. Köllen Druck & Verlag GmbH, 2007.

[UG07] Mathias Uslar and Fabian Grüning. Zur semantischen Interoperabilität in der Energiebranche: CIM IEC 61970. *Wirtschaftsinformatik*, 49(4):295–303, 2007.

[Vel07] Manuela M. Veloso, editor. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*. IJCAI, 2007.

[Vol04] Raphael Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, Universität Karlsruhe (TH), Germany, 2004.

[Wes01] Michael Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In Carole A. Goble, Deborah L. McGuinness, Ralf Möller, and Peter F. Patel-Schneider, editors, *Working Notes of the 2001 International Description Logics Workshop (DL'01)*, volume 49 of *CEUR WS Proceedings*. CEUR-WS.org, 2001.

[WWdRZ02] Frank Wolter, Heinrich Wansing, Maarten de Rijke, and Michael Zakharyaschev, editors. *Proceedings of the 3rd International Conference on Advances in Modal Logic (AiML'00)*, volume 3 of *Advances in Modal Logics*. World Scientific Publishing Company, 2002.

Index