Defeasible AceRules: A Prototype

Martin Diller
Faculty of Informatics
Technical University of Vienna, Austria
`mdiller@kr.tuwien.ac.at`

Adam Wyner
Department of Computing Science
University of Aberdeen, Scotland, UK
`azwyner@abdn.ac.uk`

Hannes Strass
Computer Science Institute
Leipzig University, Germany
`strass@informatik.uni-leipzig.de`

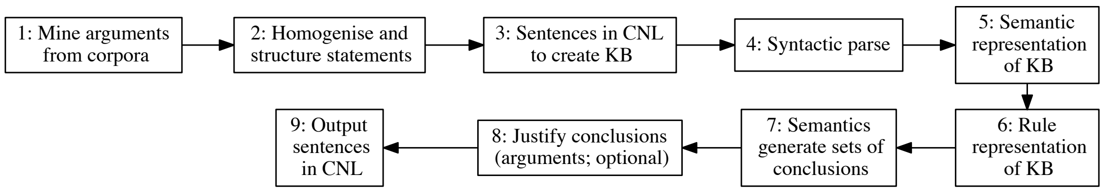
Abstract

The paper adapts the syntax of the AceRules system, an existing controlled natural language (CNL), and pipes the resulting output rules to a defeasible inference engine. With this, we can represent and reason with knowledge bases (KB) with strict and defeasible rules. For strict rules, the consequent of a strict rule must hold if the premises hold; for a defeasible rule, the consequent of a defeasible rule ought to hold if the premises hold. Strict rules are always applied whenever applicable, while defeasible rules are maximally applied (in terms of rule subsets), subject to consistency. In the CNL, defeasible rules are expressed in terms of what is "usual". The CNL expressions of rules are parsed and semantically represented so as to be used by an inference engine to accurately generate possible states of affairs. A verbaliser expresses the results in natural language, which facilitates understanding. The approach to defeasible rules is contrasted with available approaches, e.g. negation-as-failure, exceptions, and the use of abnormality predicates. The work is broadly set in the context of *argumentation*. The paper overviews the underlying formalism and the CNL; it provides an extended example.

1 Introduction

We set this work in the context of argumentation, where a central aim is to reason from premises to a conclusion using a rule, e.g. the reasoning pattern *Modus Ponens* is an argument in classic Propositional Logic. Recent efforts to formalise and instantiate "abstract" argumentation have focused on reasoning with defeasible knowledge bases (KBs) (Dung, 1995; Prakken, 2010 among others). Another development is argument mining, which attempts to extract arguments, e.g. rules as well as the premises and conclusions of the rules, from large textual corpora and structure them (Lippi and Torroni, 2016). Between abstract argumentation and argument mining, there is a substantial gap: while abstract argumentation can reason with defeasible KBs, its abstraction from linguistic information limits its applicability; argument mining can extract information from text, yet it does not extract fine-grained, highly structured semantic representations that can be used for inference. Motivated by this gap, the paper begins to address it using a controlled natural language (CNL) interface to a formal, implemented theory of reasoning from defeasible KBs. A CNL takes an *engineering approach* to natural language. On the one hand, a CNL can be used as a "target" for homogenisation and structuring of mined information, and on the other hand, it can be used as an input tool to provide well-structured KBs for a defeasible inference engine.

In this context, the paper, which develops from the works of Strass and Wyner (2017) and Wyner and Strass (2017), contributes a CNL that is augmented with a defeasible sentential operator *it is usual that* and integrated with an implemented defeasible inference engine: the system enables the input of defeasible KBs in natural language, reasoning with the knowledge base, then output of consistent answer sets in natural language. The presentation is grounded with an extended example. Schematically, the flow of analysis from source text, to inference, to natural language output consists of steps 1–9 as illustrated below. For the purposes of this paper, we focus on steps 3–7 and 9, presuming steps 1 and 2.


1: Mine arguments from corpora
2: Homogenise and structure statements
3: Sentences in CNL to create KB
4: Syntactic parse
5: Semantic representation of KB
6: Rule representation of KB
7: Semantics generate sets of conclusions
8: Justify conclusions (arguments; optional)
9: Output sentences in CNL

In Section 2, we briefly sketch some of the main component approaches to argumentation analysis, mining, and CNLs. We outline the defeasible semantics in Section 3. The main body of the paper is in Sections 4 and 5, where we first discuss how we have adapted a CNL to work with defeasible rules, and then we present examples and the inferences we draw. Finally, Section 6 closes with some general observations and future work.

2 Background

In this section, we briefly outline some of the context of our contribution, leaving as out of scope aspects of argumentation such as typologies of arguments or dialogue.

Abstract argumentation A central problem in formal reasoning is the treatment of inconsistency and non-monotonicity. Abstract argumentation (Dung, 1995) has emerged as an integrated approach. In abstract argumentation, arguments are "nodes" in a graph, where "attacks" are directed arcs that represent inconsistency. The semantics are used, essentially, to derive consistent subsets of arguments. Thus, from a globally inconsistent knowledge base, we can derive consistent information and reason further. *Instantiated argumentation* (e.g. ASPIC+; Prakken, 2010) realises the arguments as structures of strict and defeasible rules from a knowledge base. Once constructed as a graph, reasoning proceeds as in abstract argumentation. However, the construction of arguments in instantiated argumentation (e.g. in ASPIC+) itself leads to issues of overgeneration and opacity (Strass and Wyner, 2017).

Argument mining To use formal argumentation, significant KBs need to be created. Automatically mining arguments from large corpora of natural language texts has developed as a way to create corpora, primarily using machine learning approaches (Lippi and Torroni, 2016). While there have been advances, such approaches often treat the annotated passages as atomic propositions (Toni and Torroni, 2011) or without regard to linguistic structure (Cabrio and Villata, 2012), thus missing semantically meaningful, structured information that is relevant for fine-grained inference in formal logic. Rule-based approaches (Wyner et al., 2015), which can extract arguments and some of the information within sentences, are not yet sufficiently well-developed to extract highly structured information for a KB.

CNLs Controlled natural languages (CNLs) are engineered languages with finite lexicons and fixed grammatical constructions (Kuhn, 2014). Among the variety of purposes and applications, we focus on CNLs which provide unambiguous translations to machine readable semantic representations of First-order Logic such as Attempto Controlled English (ACE) with associated Prolog inference engine RACE (Fuchs et al., 2008; Fuchs, 2016) or Processable English (PENG^{ASP}; Guy and Schwitter, 2017) with associated inference engine in answer set programming (ASP). Both ACE and PENG^{ASP} provide some facility for non-monotonic reasoning using negation-as-failure, which we discuss further later. The constraints of CNLs are useful, particularly to control ambiguity and help to focus attention on particular phenomena. CNLs often have useful auxiliary functionalities such as input editors and verbalisers from semantic representations. As neither RACE nor PENG^{ASP} are open source, we work with the ACE related open source tool AceRules (Kuhn, 2007). While there are powerful, wide-coverage tools for parsing and semantic representation, e.g. C&C/Boxer (Bos, 2008), which could be used as CNLs, they require restraint in application and lack the useful auxiliary functionalities. In other related work, Gervasi and Zowghi (2005) use logical reasoning on (controlled) natural language for inconsistency management in software requirement analysis; the approach is based on Poole's THEORIST (1988), which is much like our direct-stable semantics, but for formulas instead of rules (Strass and Wyner, 2017).

3 Defeasible theories and reasoning

In this section, we give a brief formal overview of our approach to defeasible theories and defeasible reasoning, which underpins our implementation of defeasible reasoning.[1] The rationale for our approach to defeasiblity is discussed in Sections 4 and 5.

Defeasible theories For a set $\mathcal{P}$ of atomic propositions, the set of its literals is $\mathcal{L}_{\mathcal{P}} = \mathcal{P} \cup \{\neg p \mid p \in \mathcal{P}\}$. A *rule* over $\mathcal{L}_{\mathcal{P}}$ is a pair (B, h) where the finite set $B \subseteq \mathcal{L}_{\mathcal{P}}$ is called the *body* (premises) and the literal $h \in \mathcal{L}_{\mathcal{P}}$ is called the *head* (conclusion). For $B = \{b_1, \ldots, b_k\}$ with $k \in \mathbb{N}$, we can write rules thus: a *strict* rule is of the form "$b_1, \ldots, b_k \rightarrow h$"; a *defeasible* rule is of the form "$b_1, \ldots, b_k \Rightarrow h$". In case $k = 0$ we call "$\rightarrow h$" a *fact* and "$\Rightarrow h$" an *assumption*. The intuitive meaning of a rule (B, h) is that whenever we are in a state of affairs where all literals in B hold, then also literal h (always/usually, depending on the type of rule) holds. A *defeasible theory* is a tuple $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ where $\mathcal{P}$ is a set of atomic propositions, $\mathcal{S}$ is a set of strict rules over $\mathcal{L}_{\mathcal{P}}$, and $\mathcal{D}$ is a set of defeasible rules over $\mathcal{L}_{\mathcal{P}}$. Note that there is no *negation-as-failure* in this approach; we contrast our approach with one that uses *negation-as-failure* in Section 5.1. In this paper, we will consider defeasible theories with first-order predicates, variables, and constants, treating predicates on variables or constants as short-hand versions of their ground instantiations. More details can be found in the work of Strass and Wyner (2017).

The semantics of defeasible theories are a topic of ongoing work in argumentation theory (Caminada and Amgoud, 2007; Amgoud and Besnard, 2013; Strass, 2013; Wyner et al., 2015). For the purposes of this paper, we express no preference, abstracting away from any concrete manifestations of existing approaches. For our purposes here, we make a few (mild) assumptions about the approach to assigning semantics to defeasible theories that is used to draw inferences (the "back-end"). More specifically, we assume that the reasoning back-end:

1. accepts a defeasible theory $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ as input. An additional step might be needed to transform $\mathcal{T}$ into the reasoner's native input format, a point we raise again later.

2. can produce "interpretations" (consistent viewpoints, e.g. extensions) and/or (sets of) credulous/sceptical conclusions of the defeasible theory with respect to one or more semantics, e.g. stable, complete, preferred, grounded (Dung, 1995). We comment further on this below.

3. can produce graph-based justifications for its conclusions as a derivation of that literal as obtained from an argument extension.

It may be more or less straightforward to lift these restrictions, depending on the concrete approaches. Our assumptions cover considerable common ground of the various approaches in the literature; they are a meaningful and non-trivial starting point for our own work.

As mentioned above, defeasible theories can be interpreted with respect to the direct-stable semantics as defined by Strass and Wyner (2017). The direct-stable semantics is defined for the grounded instances of defeasible theories with variables, i.e. the result of replacing each rule in the defeasible theories with all possible rules that can be obtained by substituting the variables occurring in the rules with all possible constants. The result of evaluating a grounded defeasible theory with respect to the direct-stable semantics consists in sets of grounded literals, the stable sets of the theory. Informally, stable sets are consistent sets of literals where *all strict rules hold*, a *subset-maximal set of defeasible rules* hold subject to consistency, and every literal can be derived from the facts and assumptions, which are strict or defeasible rules without bodies respectively, using the remaining rules of the theory in a non-circular manner.

[1]This section is adapted from (Wyner and Strass, 2017).

More concretely, for a stable set to be consistent means that no two dual literals (i.e. an atom and its negation) can be in the set. That a rule "holds" in a set means that whenever the literals in the body of the rule are in the set, so is its head. All strict rules must hold and a $\subseteq$-maximal subset of the defeasible rules should hold without making the set inconsistent. Finally, there being a derivation of each literal from the facts and assumptions means that each literal is either a fact or an assumption or in the head of some rule in the theory having the property that each literal in the body of the rule can itself be derived from the facts and assumptions. A derivation of a literal should use the minimal set of rules needed to derive the literal. The requirement that the derivation be non-circular is cashed out by imposing a partial order on the rules used in the derivation.

4 Adding defeasibility to AceRules

In this section, we discuss the main advance reported in the paper, which is to adapt a CNL to accept natural language expressions of defeasible rules which are then provided to a defeasible reasoning engine. We adopt the Attempto Controlled English (ACE) framework (Fuchs et al., 2008), which has a parser (APE). In particular, we have adapted the AceRules tool (Kuhn, 2007), which is open source and processes ACE compatible parsed text into representations suitable for rule-based reasoners. We outline the implementation. The specific adaptation is to add defeasibility to the ACE language, then pipe the semantic representation to an implementation of the direct-stable semantics for defeasible theories of Strass and Wyner (2017). In what follows we sketch out what this means. However, due to space limitations, we do not rehearse the complex specifics or ideosyncracies of ACE other than to say it provides a fixed lexicon and grammar which provides unambiguous semantic representations. Some specific, relevant issues are detailed.

AceRules builds on APE, which parses ACE input sentences into discourse representation structures (DRSs) (Blackburn and Bos, 2005), a syntactical variant of first-order logic that supports the representation of aspects of discourse. AceRules disallows DRSs from APE which cannot be represented in the given rule language. An important component of AceRules is thus the process of "intelligent grouping" (Kuhn, 2007), the transformation of DRSs generated by APE which are not directly compliant with a rule language into DRSs that are compliant with the rule language. Grouping works by "aggregating" predicates, replacing complex expressions with equivalent simpler expressions, e.g. expressions inside of negation or in the head of a conditional. Moreover, grouping often also involves removing implicitly quantified variables (see the work of Kuhn, 2007, for examples).

Target rule languages supported by AceRules are, on the one hand, logic programs with strong as well as default negation (negation-as-failure) and, on the other hand, acyclic logic programs with both forms of negation as well as priorities over rules ("override" statements). The first are intended to be interpreted with respect to stable semantics of Gelfond and Lifschitz (1990), while the second with respect to the courteous semantics of Grosof (1997). AceRules relies on external tools (e.g. SModels and LParse) for interpreting programs with respect to stable semantics, while the courteous semantics is implemented natively. Crucially, AceRules also includes a verbalisation component, whereby the outputs of the interpreter-component are recast in DRSs which can then be verbalised by (the paraphrasing mechanism of) APE.

In contrast to the AceRules supported rule languages, in our rule language we disallow default negation and priorities over rules, while extending the input language accepted by AceRules with the constructs "*It is usual that* ..." and "*If* ... *then it is usual that* ...", which are respectively interpreted as assumptions and defeasible rules. For a discussion of the linguistic semantics see the discussions of Lewis (1975) and Kratzer (2012); however, while loosely related, we do not explicitly tie our analysis to that discussion. From the point of view of the interpretation of the CNL in a formal language, i.e. at the level of the DRSs, we treat defeasible rules analogously to strict rules and hence are able to make use of the techniques for translating from the output of APE (the DRSs) to the rule languages accepted by AceRules. The crucial difference is that we distinguish between the types of rules for the purposes of reasoning with the resulting defeasible theory, which amounts to having two forms of conditionals also at the level of the DRSs.

To exercise our approach we have implemented a script[2] that makes use of AceRules (and APE) as well as the answer set encodings for interpreting defeasible theories and the answer set solver (clingo) as described by Strass and Wyner (2017). More concretely, we make separate use of the AceRule parser and the verbaliser components, thus enabling our approach to be implemented by a simple interleaving of calls to the AceRules (and APE) parser, transformation to rules, solver for the direct-stable semantics (via encodings and a solver for answer set programming), and finally the AceRules (and APE) verbaliser.

Crucially, we pre-process the input text removing all constructs indicating defeasibility and make use of the AceRules parser "as if" all rules in the input were strict, but at the same time externally tracking which rules are defeasible and which are not. By differentiating the rules in this way, we are able to use the encodings for the direct-stable semantics later on in the pipeline. At the level of the stable sets, the distinction between strict and defeasible rules is irrelevant; and we are, hence, also able to make direct use of the AceRules (and APE) verbaliser component with the caveats that we mention later on in Section 5.2. We have followed this rather quick strategy for the implementation because our objective to this point was to have a prototype testing-ground for our approach as well as to develop an in depth understanding of the issues before us. Clearly, this work demonstrates that we can adapt AceRules to work with defeasible rules and to tie it in to the encodings for the direct-stable semantics.

5 Natural language interface

In this section, we discuss an extended working example, then address some of the subtleties. Our main aim in this section is to demonstrate by example the added value of defeasible rules and reasoning using a CNL. We only discuss the examples, results, and issues; the DRSs (without "usual") can all be viewed using ACE's online APE webclient.[3]

5.1 Working example: extending AceWiki with defeasible rules

We now exemplify and further motivate our approach by showing its use in the context of AceWiki (Kuhn, 2009),[4] a prototype of an encyclopedia in the style of the popular Wikipedia,[5] but where articles are written using ACE rather than unrestricted natural language. The advantage to using ACE in a wiki is that non-expert users can edit AceWiki entries, while at the same time users can use complex question answering and draw inferences. As it currently stands, AceWiki can represent a consistent KB about some domain and uses only strict rules.

Although the use of full ACE in the context of AceWiki is desirable, the undecidability of ACE (see Fuchs et al., 2008) also means that it is not feasible in practice. Restricting ACE to efficiently decidable fragments, e.g. via translation to a form of rule language, provides a more promising way forward. We base our example on current entries in the AceWiki about geographical information,[6] which have been restricted to a variant of ACE that can be translated into the rule language OWL 2 RL, and thus also, in principle, into the fragment of ACE admitted by AceRules. However, AceWiki could similarly be deployed in other fields such as Biology, Medicine, or more generally in any context where a structured KB would be useful.

As a motivating example, consider the entry for *island* in the geographical AceWiki. Some straightforward statements pertaining to the strict definition of *island* appear, e.g. *Every island is a land-mass* and *Every island is surrounded by a body of water*. Using ACE, such statements can be written in a

[2]The script together with other necessary files as well as the examples in this paper is available at `https://www.dbai.tuwien.ac.at/proj/adf/dAceRules`.

[3]APE webclient: `http://attempto.ifi.uzh.ch/site/resources/`

[4]AceWiki can be accessed at `http://attempto.ifi.uzh.ch/acewiki/`.

[5]`https://www.wikipedia.org/`

[6]`http://attempto.ifi.uzh.ch/webapps/acewikigeo/`

straightforward manner and are automatically translated to a rule language:[7]

```
(1) Every island is a land-mass.
(2) If X is an island then a body-of-water surrounds X.
```

Statement (1) leads to a rule like *island*$(x) \rightarrow$ *land-mass*(x) with a first-order variable x (cf. Section 3). Due to a lack of space, we will not explicitly present further rules in the paper; in any case, they can be obtained from the presented text via AceRules.

ACE enables the addition of lexical entries, such as proper names *Mainland-Shetland* or *St-Ninians-Isle*. Moreover, ACE is often able to deduce the word class for words that are not in its lexicon from the context. There are some interactions in ACE/AceRules in relation to the verb form, quantifier scope, and the verbaliser (among other subtleties) such that, for example, we have represented (2) as a rule; we suppress further such incidental comments. We do however further note that rule (2) illustrates the situation where the input text introduces an implicitly quantified variable in the head of a rule (here, referring to a body-of-water), which needs to be treated by the grouping-mechanism of AceRules we alluded to in Section 4.

The problem, which we develop, is to add a new entry for *tied-island* to this AceWiki. However, as we show, this would lead to inconsistency were we to only have strict rules. According to Wikipedia, tied islands "are landforms consisting of an island that is connected to land only by a tombolo: a spit of beach materials connected to land at both ends."[8] With slight simplification, this definition can be written into AceWiki as follows:

```
(3) Every tied-island is an island.
(4) Every tied-island attaches-to a land-mass.
```

A prominent example of a *tied-island* according to the Wikipedia entry is *St. Ninian's Isle*, which is attached to *Mainland Shetland*, the largest of the Shetland Islands off the coast of Scotland. Thus, entries for St. Ninian's Isle and Mainland Shetland in AceWiki would be:

```
(5) Mainland-Shetland is an island.
(6) St-Ninians-Isle is a tied-island.
(7) St-Ninians-Isle is a part of the Shetland-Islands.
```

According to the Wikipedia entry for St. Ninian's Isle, during the winter strong wave action removes sand from the tombolo that connects St. Ninian to Mainland Shetland such that the tombolo is usually covered at high tide and occasionally throughout the tidal cycle. Hence, simply stating that *St. Ninian's Isle attaches to Mainland Shetland* would be incorrect. Spelling out the exact conditions under which St. Ninian's Isle is connected to Mainland-Shetland, which corresponds to using exceptions in strict rules, seems quite difficult if even possible (or desirable) and would be rather uncommon for an application like AceWiki. Rather, an easy solution is provided by the use of the predicate *it is usual that* applied to a statement:

```
(8) It is usual that St-Ninians-Isle attaches-to Mainland-Shetland.
```

Let us now turn to a more fundamental reason for being able to distinguish between defeasible and strict statements in a CNL. Consider now the result of having all of the above statements in the AceWiki together with the following fairly uncontroversial statements referring to the meanings of *being attached to a land mass*, *being surrounded by water*, and *being a part of*. We initially highlight the issue using further strict rules. In particular, statements (12) and (13) are needed too, because we need to define in the wiki's KB that St. Ninian's Isle is attached to exactly one land mass, namely Mainland Shetland.

[7]We extended the lexicon of ACE with some further terms, e.g. *land-mass* and *body-of-water*, but suppress further discussion.

[8]`https://en.wikipedia.org/wiki/Tied_island` (accessed on 4/4/2017)

```
(9) If X attaches-to a land-mass then it is false that a body-of-water
    surrounds X.
(10) If a body-of-water surrounds X then it is false that X
     attaches-to a land-mass.
(11) If St-Ninians-Isle attaches-to Mainland-Shetland then St-Ninians-Isle
     is a part of Mainland-Shetland.
(12) If St-Ninians-Isle attaches-to Mainland-Shetland then St-Ninians-Isle
     attaches-to a land-mass.
(13) If St-Ninians-Isle attaches-to a land-mass then St-Ninians-Isle
     attaches-to Mainland-Shetland.
```

Since according to (6) St. Ninian's Isle is a tied island, and according to (3) every tied island is an island, and both (3) as well as (6) are strict rules, the direct stable semantics forces one to conclude that St. Ninian's Isle is an island. Now, because St. Ninian's Isle is an island and following (2), we conclude that a body of water surrounds St. Ninian's Isle. But from the fact that St. Ninian's is also a tied island and (4), St. Ninian's Isle attaches to a land mass. This leads to a contradiction according to statements (9) and (10). Hence, the entire AceWiki is deemed inconsistent and further reasoning is invalidated.

Note that the AceWiki remains inconsistent even after removing statement (8); the reason for the apparent contradiction in the Wiki is the fact, as is stated in the Wikipedia entry referring to St. Ninian's Isle,[9] that "[d]epending on the definition used, St. Ninian's is [. . .] either an island, or a peninsula." This reveals that the definition for *tied-island* in (3) should also be *defeasible*. However, in contrast to the reasons for the defeasiblity of (8), this is now due to the fact that there is no consensus on the meaning of *tied island*. Thus, we replace (3) with the more accurate statement:

```
(3') If X is a tied-island then it is usual that X is an island.
```

The consequence is that there is now one stable set:

```
ANSWER-TEXT #1:

There is a body-of-water X1.
St-Ninians-Isle is a tied-island.
Mainland-Shetland is a land-mass.
Mainland-Shetland is an island.
St-Ninians-Isle is a part of Shetland-Islands.
St-Ninians-Isle is a part of Mainland-Shetland.
St-Ninians-Isle attaches-to a land-mass.
The body-of-water X1 surrounds Mainland-Shetland.
St-Ninians-Isle attaches-to Mainland-Shetland.
It is false that Mainland-Shetland attaches-to a land-mass.
It is false that a body-of-water surrounds St-Ninians-Isle.
```

Here the conclusion is that St. Ninian's Isle is a tied island that is attached to Mainland Shetland, while nothing can be said regarding whether St. Ninian's is also an island or not. The reason is that since statement (4) is strict, (8) is also effectively interpreted as a strict rule; that is, (8) strictly holds. To make (4) consistent with the *intended reading* of (8), (4) should be replaced with:

```
(4') If X is a tied-island then it is usual that X attaches-to a
     land-mass.
```

The result is that there are now two stable sets (answer-texts), which have in common the statements:

[9]`https://en.wikipedia.org/wiki/St_Ninian's_Isle` (accessed on 4.4.2017)

```
There is a body-of-water X1.
St-Ninians-Isle is a tied-island.
Mainland-Shetland is a land-mass.
Mainland-Shetland is an island.
St-Ninians-Isle is a part of Shetland-Islands.
The body-of-water X1 surrounds Mainland-Shetland.
It is false that Mainland-Shetland attaches-to a land-mass.
```

One stable set contains the following statements in addition to the common statements:

```
St-Ninians-Isle is a part of Mainland-Shetland.
St-Ninians-Isle attaches-to a land-mass.
St-Ninians-Isle attaches-to Mainland-Shetland.
It is false that a body-of-water surrounds St-Ninians-Isle.
```

The other stable set contains the following statements in addition to the common statements:

```
There is a body-of-water X2.
St-Ninians-Isle is a land-mass.
St-Ninians-Isle is an island.
The body-of-water X2 surrounds St-Ninians-Isle.
It is false that St-Ninians-Isle attaches-to a land-mass.
```

The interpretation of the latter set of statements is that St. Ninian's Isle is a tied island, but can only be called an island when it is not attached to Mainland-Shetland. Also relaxing the definition of *island* by changing (2) to

```
(2') If X is an island then it is usual that a body-of-water
     surrounds X.
```

has the consequence that St. Ninian's Isle can also (always) be considered an island, despite the fact that the isle is not always surrounded by water.

Summarizing, we have shown that by distinguishing between defeasible and strict statements, we can resolve apparent inconsistencies such as might arise, in our example, because of the use of generic statements that allow for exceptions or because different meanings can be attached to certain words.

However, our approach does not require explicit statement of exceptions or alternatives. Using non-artificial, specific exceptions together with negation-as-failure in strict rules is often not feasible nor desirable. More fundamentally, using artificial exceptions, e.g. *abnormality* predicates specific to each rule, will usually not lead to a satisfactory result. Consider, for instance the effect of having the statement (8") below rather than the statement (8) mentioned previously, while replacing (3) with (3") rather than (3'), (4) with (4") rather than (4'), as well as (2) with (2") rather than (2').

```
(8'')If it is not provable that it is false that St-Ninians-Isle attaches-to
     Mainland-Shetland then St-Ninians-Isle attaches-to Mainland-Shetland.
(3'') If X is a tied-island and it is not provable that X is not an
      island then X is an island.
(4'') If X is a tied-island and it is not provable that it is false that
      X attaches-to a land-mass then X attaches-to a land-mass.
(2'') If X is an island and it is not provable that it is false that a
      body-of-water surrounds X then a body-of-water surrounds X.
```

The resulting text does not have any answer set under the standard stable semantics for logic programs.[10] Interpreting the text under the courteous semantics does produce a unique answer set, but this approach

[10]This is not to say that it is not possible to simulate the evaluation of ACE texts under the direct-stable semantics by using logic programs without the defeasible conditional; in fact the encodings by Strass and Wyner (2017) provide such a simulation (via disjunctive logic programs). On the other hand, the complexity results by Strass and Wyner (2017) also suggest that any such simulation via normal or extended logic programs will involve a worst-case exponential blow-up in general (unless the polynomial hierarchy collapses to its first level), at least for ACE texts which can be parsed as *grounded* defeasible theories.

is unsatisfactory in general. First, because the rules must be acyclic and second because the resulting answer is often uninformative or somewhat arbitrary. In the current case, the rules are in fact cyclic and hence no answer is produced.

An auxiliary point is that the discussion above shows the utility of the tool, for it allows us to experiment using natural language with alternative inputs to determine alternative outputs (and compare semantics between them). From such alternatives, we can identify our preferred inputs, semantics, and outputs, which are intuitively plausible and computationally feasible.

5.2 Subtleties of defeasibility

In the previous section, we have seen what follows from different statements about what is defeasible. Which statements are strict or defeasible is not always explicit in natural language statements, but might be implicit or contextual. There appears to be no lexical distinction between strict and defeasible conditionals. Yet such a distinction seems essential in a CNL, which requires some explicit representations of the input. This is similar to the issue of whether negation in natural language out to be represented as strict or as negation-as-failure. Making an explicit distinction between strict and defeasible rules is especially pressing in the context of an application like AceWiki, where the claim can be made that, given the possibility of error, (virtually) all statements be considered defeasible.

We now consider another more fundamental issue that arises when introducing defeasibility in a controlled natural language. This is an issue that arises with similar constructs such as "it is false that ..." and "it is possible that ..." and is the question of the *semantic scope* of the operator *usual*. Consider the following two sentences in ACE extended with *usual*:

```
It is usual that Mainland-Shetland is damp and eerie and desolate.
Mainland-Shetland is not eerie.
```

In our current approach using AceWiki, the first sentence in the text above gets parsed as the conjunction of "it is usual that Mainland-Shetland is damp", "it is usual that Mainland-Shetland is eerie", and "it is usual that Mainland-Shetland is desolate", the only stable set hence being one where Mainland Shetland is damp and desolate, but not eerie.

Although this approach of having *usual* "distribute" over complex expressions within its scope seems satisfactory in the above example, further study is needed to determine whether this distribution always respects the intended meaning. As one possible counter-example, consider the following fragment of text, where the scope of *usual* in the second sentence includes an indefinite pronoun.

```
It is usual that Mainland-Shetland is inaccessible from Aberdeen.
It is usual that a ferry that starts in Aberdeen services Mainland-Shetland.
If a ferry that starts in Aberdeen services Mainland-Shetland then
Mainland-Shetland is not inaccessible from Aberdeen.
```

The result of evaluating this fragment of text is that there are four stable-sets, one in which Mainland Shetland is accessible via a ferry that starts in Aberdeen, while in the context of the remaining three stable sets Mainland Shetland is not accessible from Aberdeen. In each of these latter stable sets Mainland Shetland is inaccessible because: there is a ferry that starts in Aberdeen, but it does not service Mainland-Shetland; there is a ferry that services Mainland-Shetland, but it does not start in Aberdeen; and finally and anomalously, there simply is no ferry (while, at the same time, it being the case that were there a ferry, it would start in Aberdeen and service Mainland-Shetland).

The issue here is the particularities of AceRules parsing and semantic representation. The sentence "It is usual that a ferry that starts in Aberdeen services Mainland-Shetland" gets parsed as the conjunction of "It is usual that there is a ferry X1", "It is usual that the ferry X1 services Mainland-Shetland", and "It is usual that the ferry X1 starts in Aberdeen". However, one could argue that the result of the parse should be "it is usual that there is a ferry X1 that starts in Aberdeen" and "it is usual that the ferry X1 services Mainland-Shetland".

Indeed, this issue can be taken as support for our approach in that we can, first of all, enforce our intended reading by input of:

```
There is a ferry that starts in Aberdeen.
It is usual that the ferry services Mainland-Shetland.
```

Secondly, one can envision instances in which it is not desirable that "there being a ferry X1" grammatically associates with "starting in Aberdeen". For example in the following structurally very similar text:

```
It is usual that a ferry that starts early services Mainland-Shetland.
It is usual that a ferry that starts late services Mainland-Shetland.
If a ferry that starts early services Mainland-Shetland then it is
false that a ferry that starts late services Mainland-Shetland.
If a ferry that starts late services Mainland-Shetland then it is
false that a ferry that starts early services Mainland-Shetland.
```

one would expect it be possible to derive, concordantly with the direct stable semantics, that there is a ferry that services Mainland-Shetland, independently of whether the ferry in question starts late or early.

In our current implementation, we issue a warning whenever a statement corresponding to a defeasible rule is "split" or "distributed" into several defeasible rules. This is particularly important as such a distribution can lead to errors when verbalising the resulting answer sets. In particular, the mentioned fourth anomalous stable-set mentioned previously cannot be verbalised (in the current implementation) because of the fact that the stable set contains facts referring to something starting in Aberdeen and servicing Mainland-Shetland while at the same time there being no concrete object to which this "something" can be associated with (i.e. the ferry). The impact of these observations are that one must take care to integrate the syntax, semantic representation, and inference engine in order to obtain accurate and plausible (if not intended) output.

6 Conclusion

The paper has motivated the development of a CNL with defeasible reasoning, adapting the AceRules system with defeasible rules. We have provided background on defeasibility and reasoning, along with a discussion about how defeasible rules are introduced to AceRules. We have an extended example to exercise the tool, showing the utility and advantages of representing and reasoning with defeasible rules.

There are numerous opportunities for further development. ACE's APE parser can be augmented with capacities to represent tense, additional verb constructions, and subordinate clauses for justifications such as *because*. The example can be incrementally extended, testing the output results to ensure they comply with intuitions. There are a range of issues to address about the verbalisation, most importantly to provide some means to reconstruct claims and justifications in the output results, e.g. "X because Y", rather than producing a list of statements. A possible starting point is the direct-stable semantics implementation's ability to produce derivations of concluded literals.

More fundamentally, the interactions between the rule expressions output by AceRules, the rule language of the inference engine, and the verbaliser need greater study and development. As it is, AceRules uses grouping to adjust DRSs to the rule language, which raises several complexities; whether this can be relaxed to taken advantage of recent advances in rules languages remains to be seen. Relatedly, further work needs to be carried out on the complexity of grounding of variables and optimisations to limit blow-up, which has not been discussed in this paper. Other avenues of investigation might be rule decomposition, which could help to optimise grounding. More generally, we expect to incorporate other natural language expressions of defeasibility or genericity into a CNL with defeasible rules. Clearly, the approach we have developed opens a range of avenues for future research.

References

Amgoud, L. and P. Besnard (2013). A formal characterization of the outcomes of rule-based argumentation systems. In *SUM*, Volume 8078 of *LNCS*, pp. 78–91. Springer.

Blackburn, P. and J. Bos (2005). *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Publications.

Bos, J. (2008). Wide-coverage semantic analysis with Boxer. In J. Bos and R. Delmonte (Eds.), *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pp. 277–286. College Publications.

Cabrio, E. and S. Villata (2012). Natural language arguments: A combined approach. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pp. 205–210.

Caminada, M. and L. Amgoud (2007). On the evaluation of argumentation formalisms. *Artificial Intelligence 171*(5–6), 286–310.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence 77*(2), 321–358.

Fuchs, N. E. (2016). Reasoning in Attempto Controlled English: Non-monotonicity. In B. Davis, G. J. Pace, and A. Wyner (Eds.), *Controlled Natural Language – 5th International Workshop, CNL 2016, Aberdeen, UK, July 25-27, 2016, Proceedings*, Volume 9767 of *Lecture Notes in Computer Science*, pp. 13–24. Springer.

Fuchs, N. E., K. Kaljurand, and T. Kuhn (2008). Attempto Controlled English for knowledge representation. In *Reasoning Web*, pp. 104–124.

Gelfond, M. and V. Lifschitz (1990). Logic programs with classical negation. In D. H. D. Warren and P. Szeredi (Eds.), *Logic Programming, Proceedings of the Seventh International Conference, Jerusalem, Israel, June 18-20, 1990*, pp. 579–597. MIT Press.

Gervasi, V. and D. Zowghi (2005). Reasoning about inconsistencies in natural language requirements. *ACM Trans. Softw. Eng. Methodol. 14*(3), 277–330.

Grosof, B. N. (1997). Prioritized conflict handling for logic programs. In J. Maluszynski (Ed.), *Logic Programming, Proceedings of the 1997 International Symposium, Port Jefferson, Long Island, NY, USA, October 13-16, 1997*, pp. 197–211. MIT Press.

Guy, S. and R. Schwitter (2017). The PENG^{ASP} system: Architecture, language and authoring tool. *Language Resources and Evaluation 51*(1), 67–92.

Kratzer, A. (2012). Modals and conditionals.

Kuhn, T. (2007). AceRules: Executing rules in controlled natural language. In *Web Reasoning and Rule Systems, First International Conference, RR 2007, Innsbruck , Austria, June 7-8, 2007, Proceedings*, Volume 4524 of *Lecture Notes in Computer Science*, pp. 299–308. Springer.

Kuhn, T. (2009). AceWiki: A semantic wiki using controlled English. In *Proceedings of the Poster Session at the 6th European Semantic Web Conference (ESWC09).*

Kuhn, T. (2014). A survey and classification of controlled natural languages. *Computational Linguistics 40*(1), 121–170.

Lewis, D. (1975). Adverbs of quantification. In *Formal Semantics of Natural Language*, pp. 178–188. Cambridge University Press.

Lippi, M. and P. Torroni (2016). Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology 16*(2), 10:1–10:25.

Poole, D. (1988). A logical framework for default reasoning. *Artificial Intelligence 36*(1), 27–47.

Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument & Computation 1*(2), 93–124.

Strass, H. (2013, September). Instantiating knowledge bases in Abstract Dialectical Frameworks. In *Proceedings of the Fourteenth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XIV)*, Volume 8143 of *LNCS*, pp. 86–101. Springer.

Strass, H. and A. Wyner (2017, February). On automated defeasible reasoning with controlled natural language and argumentation. In R. Barták, T. L. McCluskey, and E. Pontelli (Eds.), *Proceedings of the Second International Workshop on Knowledge-based Techniques for Problem Solving and Reasoning (KnowProS)*.

Toni, F. and P. Torroni (2011). Bottom-up argumentation. In *Theorie and Applications of Formal Argumentation - First International Workshop, TAFA 2011. Barcelona, Spain, July 16-17, 2011, Revised Selected Papers*, pp. 249–262.

Wyner, A., T. Bench-Capon, P. Dunne, and F. Cerutti (2015). Senses of 'argument' in instantiated argumentation frameworks. *Argument & Computation 6*(1), 50–72.

Wyner, A. Z., W. Peters, and D. Price (2015). Argument discovery and extraction with the argument workbench. In *Proceedings of the 2nd Workshop on Argumentation Mining, ArgMining@HLT-NAACL 2015, June 4, 2015, Denver, Colorado, USA*, pp. 78–83.

Wyner, A. Z. and H. Strass (2017). dARe – Using argumentation to explain conclusions from a controlled natural language knowledge base. In S. Benferhat, K. Tabia, and M. Ali (Eds.), *Proceedings of the Thirtieth International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017*, Volume 10351 of *Lecture Notes in Computer Science*, pp. 328–338. Springer.