TECHNISCHE UNIVERSITÄT DRESDEN

RECENT ADVANCES IN REASONING WITH EXISTENTIAL RULES

Markus Krötzsch

Knowledge-Based Systems

reporting joint work with

David Carral (TUD), Irina Dragoste (TUD), Ceriel Jacobs (VUE), Jacopo Urbani (VUE)

EMCL Workshop, 21 Feb 2018

Existential Rules

Existential Rules are sentences of the form

$$\forall \vec{x}.\, \left(\varphi \;\to\; \exists \vec{v}.\, \psi\right)$$

where φ (body) and ψ (head) are conjunctions of atoms.

Existential Rules

Existential Rules are sentences of the form

$$\forall \vec{x}. \left(\varphi \to \exists \vec{v}. \psi\right)$$

where φ (body) and ψ (head) are conjunctions of atoms.

What do we want to do?

- Main reasoning tasks on rules: answering conjunctive queries
- Challenge: this is undecidable in general

Existential Rules

Existential Rules are sentences of the form

$$\forall \vec{x}.\, \left(\varphi \;\rightarrow\; \exists \vec{v}.\, \psi\right)$$

where φ (body) and ψ (head) are conjunctions of atoms.

What do we want to do?

- Main reasoning tasks on rules: answering conjunctive queries
- Challenge: this is undecidable in general

Why?

- Rules are a powerful data query paradigm (Datalog!) – applications in data management, program analysis, business analytics, social network analysis, . . .
- Existential rules are a powerful ontology language – generalising Horn ontologies, lightweight OWL profiles, knowledge graph formalisms, . . .

The Chase

Example:

$$\text{Bicycle}(a) \tag{1}$$

$$\text{Bicycle}(x) \to \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \tag{2}$$

$$\text{Wheel}(x) \to \exists w.\text{hasPart}(x, w) \wedge \text{Spoke}(w) \tag{3}$$

$$\text{Spoke}(x) \to \exists u.\text{partOf}(x, u) \wedge \text{Bicycle}(u) \tag{4}$$

$$\text{hasPart}(x, y) \to \text{partOf}(y, x) \tag{5}$$

$$\text{partOf}(x, y) \wedge \text{partOf}(y, z) \to \text{partOf}(x, z) \tag{6}$$

(**Notes:** (1) ∀ are tacitly omitted; (2) these rules could be expressed in description logic)

The Chase

Example:

$$\mathsf{Bicycle}(a) \tag{1}$$

$$\mathsf{Bicycle}(x) \to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \tag{2}$$

$$\mathsf{Wheel}(x) \to \exists w.\mathsf{hasPart}(x, w) \wedge \mathsf{Spoke}(w) \tag{3}$$

$$\mathsf{Spoke}(x) \to \exists u.\mathsf{partOf}(x, u) \wedge \mathsf{Bicycle}(u) \tag{4}$$

$$\mathsf{hasPart}(x, y) \to \mathsf{partOf}(y, x) \tag{5}$$

$$\mathsf{partOf}(x, y) \wedge \mathsf{partOf}(y, z) \to \mathsf{partOf}(x, z) \tag{6}$$

(**Notes:** (1) $\forall$ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: "chasing the rules"

The Chase

Example:

$$\text{Bicycle}(a) \tag{1}$$

$$\text{Bicycle}(x) \to \exists v.\text{hasPart}(x, v) \land \text{Wheel}(v) \tag{2}$$

$$\text{Wheel}(x) \to \exists w.\text{hasPart}(x, w) \land \text{Spoke}(w) \tag{3}$$

$$\text{Spoke}(x) \to \exists u.\text{partOf}(x, u) \land \text{Bicycle}(u) \tag{4}$$

$$\text{hasPart}(x, y) \to \text{partOf}(y, x) \tag{5}$$

$$\text{partOf}(x, y) \land \text{partOf}(y, z) \to \text{partOf}(x, z) \tag{6}$$

(**Notes:** (1) ∀ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: “chasing the rules”

a : Bicycle

The Chase

Example:

$$\text{Bicycle}(a) \tag{1}$$

$$\text{Bicycle}(x) \to \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \tag{2}$$

$$\text{Wheel}(x) \to \exists w.\text{hasPart}(x, w) \wedge \text{Spoke}(w) \tag{3}$$

$$\text{Spoke}(x) \to \exists u.\text{partOf}(x, u) \wedge \text{Bicycle}(u) \tag{4}$$

$$\text{hasPart}(x, y) \to \text{partOf}(y, x) \tag{5}$$

$$\text{partOf}(x, y) \wedge \text{partOf}(y, z) \to \text{partOf}(x, z) \tag{6}$$

(**Notes:** (1) $\forall$ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: "chasing the rules"

The Chase

Example:

$$\text{Bicycle}(a) \tag{1}$$

$$\text{Bicycle}(x) \to \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \tag{2}$$

$$\text{Wheel}(x) \to \exists w.\text{hasPart}(x, w) \wedge \text{Spoke}(w) \tag{3}$$

$$\text{Spoke}(x) \to \exists u.\text{partOf}(x, u) \wedge \text{Bicycle}(u) \tag{4}$$

$$\text{hasPart}(x, y) \to \text{partOf}(y, x) \tag{5}$$

$$\text{partOf}(x, y) \wedge \text{partOf}(y, z) \to \text{partOf}(x, z) \tag{6}$$

(**Notes:** (1) $\forall$ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: “chasing the rules”

The Chase

Example:

$$\text{Bicycle}(a) \tag{1}$$

$$\text{Bicycle}(x) \rightarrow \exists v.\text{hasPart}(x, v) \land \text{Wheel}(v) \tag{2}$$

$$\text{Wheel}(x) \rightarrow \exists w.\text{hasPart}(x, w) \land \text{Spoke}(w) \tag{3}$$

$$\text{Spoke}(x) \rightarrow \exists u.\text{partOf}(x, u) \land \text{Bicycle}(u) \tag{4}$$

$$\text{hasPart}(x, y) \rightarrow \text{partOf}(y, x) \tag{5}$$

$$\text{partOf}(x, y) \land \text{partOf}(y, z) \rightarrow \text{partOf}(x, z) \tag{6}$$

(**Notes:** (1) ∀ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: “chasing the rules”

$$a : \text{Bicycle} \xrightarrow{\text{hasPart}} w(a) : \text{Wheel} \xrightarrow{\text{hasPart}} s(w(a)) : \text{Spoke} \xrightarrow{\text{partOf}} b(s(w(a))) : \text{Bicycle}$$

The Chase

Example:

$$\text{Bicycle}(a) \quad (1)$$

$$\text{Bicycle}(x) \to \exists v.\text{hasPart}(x, v) \land \text{Wheel}(v) \quad (2)$$

$$\text{Wheel}(x) \to \exists w.\text{hasPart}(x, w) \land \text{Spoke}(w) \quad (3)$$

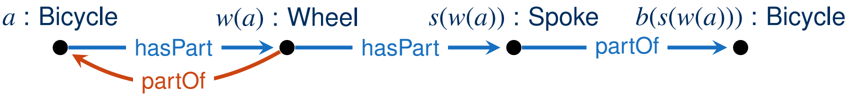
$$\text{Spoke}(x) \to \exists u.\text{partOf}(x, u) \land \text{Bicycle}(u) \quad (4)$$

$$\text{hasPart}(x, y) \to \text{partOf}(y, x) \quad (5)$$

$$\text{partOf}(x, y) \land \text{partOf}(y, z) \to \text{partOf}(x, z) \quad (6)$$

(**Notes:** (1) ∀ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: "chasing the rules"



The Chase

Example:

$$\mathsf{Bicycle}(a) \tag{1}$$

$$\mathsf{Bicycle}(x) \to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \tag{2}$$

$$\mathsf{Wheel}(x) \to \exists w.\mathsf{hasPart}(x, w) \wedge \mathsf{Spoke}(w) \tag{3}$$

$$\mathsf{Spoke}(x) \to \exists u.\mathsf{partOf}(x, u) \wedge \mathsf{Bicycle}(u) \tag{4}$$

$$\mathsf{hasPart}(x, y) \to \mathsf{partOf}(y, x) \tag{5}$$

$$\mathsf{partOf}(x, y) \wedge \mathsf{partOf}(y, z) \to \mathsf{partOf}(x, z) \tag{6}$$

(**Notes:** (1) $\forall$ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: "chasing the rules"

The Chase

Example:

$$\mathsf{Bicycle}(a) \tag{1}$$

$$\mathsf{Bicycle}(x) \to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \tag{2}$$

$$\mathsf{Wheel}(x) \to \exists w.\mathsf{hasPart}(x, w) \wedge \mathsf{Spoke}(w) \tag{3}$$

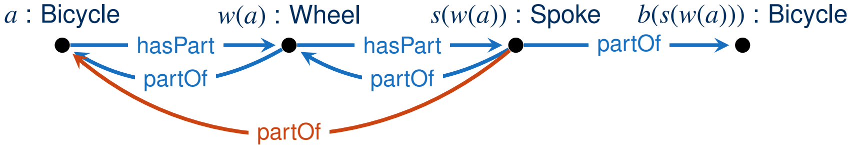
$$\mathsf{Spoke}(x) \to \exists u.\mathsf{partOf}(x, u) \wedge \mathsf{Bicycle}(u) \tag{4}$$

$$\mathsf{hasPart}(x, y) \to \mathsf{partOf}(y, x) \tag{5}$$

$$\mathsf{partOf}(x, y) \wedge \mathsf{partOf}(y, z) \to \mathsf{partOf}(x, z) \tag{6}$$

(**Notes:** (1) ∀ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: “chasing the rules”



The Chase

Example:

$$\text{Bicycle}(a) \quad (1)$$

$$\text{Bicycle}(x) \to \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \quad (2)$$

$$\text{Wheel}(x) \to \exists w.\text{hasPart}(x, w) \wedge \text{Spoke}(w) \quad (3)$$

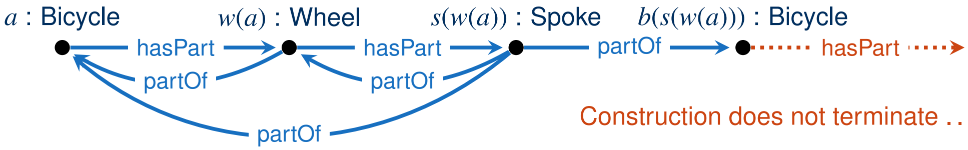
$$\text{Spoke}(x) \to \exists u.\text{partOf}(x, u) \wedge \text{Bicycle}(u) \quad (4)$$

$$\text{hasPart}(x, y) \to \text{partOf}(y, x) \quad (5)$$

$$\text{partOf}(x, y) \wedge \text{partOf}(y, z) \to \text{partOf}(x, z) \quad (6)$$

(**Notes:** (1) ∀ are tacitly omitted; (2) these rules could be expressed in description logic)

Bottom-up model construction: “chasing the rules”



Construction does not terminate . . .

Stopping the Chase

Restricted Chase:

- Apply rules $\forall\vec{x}.(\varphi \to \exists\vec{v}.\psi)$ with substitution σ only if $\exists\vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Stopping the Chase

Restricted Chase:

- Apply rules $\forall\vec{x}.(\varphi \to \exists\vec{v}.\psi)$ with substitution σ only if $\exists\vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:

Restricted Chase:

- Apply rules $\forall \vec{x}.(\varphi \to \exists \vec{v}.\psi)$ with substitution σ only if $\exists \vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:

Restricted Chase:

- Apply rules $\forall\vec{x}.(\varphi \to \exists\vec{v}.\psi)$ with substitution σ only if $\exists\vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

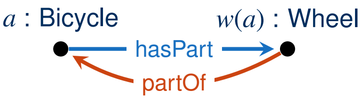
Restricted chase computation:

Stopping the Chase

Restricted Chase:

- Apply rules $\forall \vec{x}.(\varphi \to \exists \vec{v}.\psi)$ with substitution σ only if $\exists \vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:

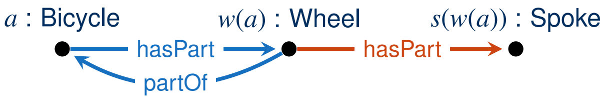


Stopping the Chase

Restricted Chase:

- Apply rules $\forall \vec{x}.(\varphi \to \exists \vec{v}.\psi)$ with substitution σ only if $\exists \vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:

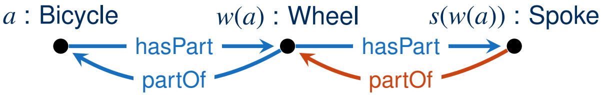


Stopping the Chase

Restricted Chase:

- Apply rules $\forall\vec{x}.(\varphi \to \exists\vec{v}.\psi)$ with substitution σ only if $\exists\vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:

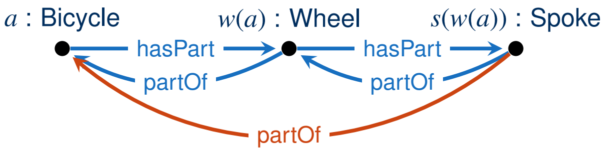


Stopping the Chase

Restricted Chase:

- Apply rules $\forall\vec{x}.(\varphi \to \exists\vec{v}.\psi)$ with substitution σ only if $\exists\vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:

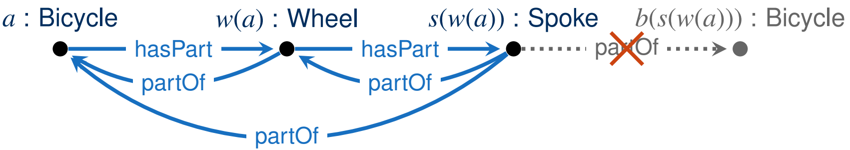


Stopping the Chase

Restricted Chase:

- Apply rules $\forall\vec{x}.(\varphi \to \exists\vec{v}.\psi)$ with substitution σ only if $\exists\vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:



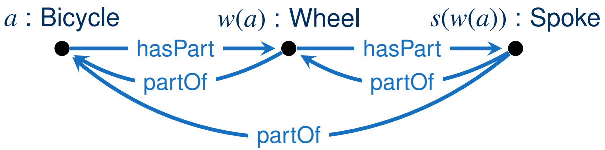
$\models \exists u.\text{partOf}(s(w(a)), u) \land \text{Bicycle}(u)$

Stopping the Chase

Restricted Chase:

- Apply rules $\forall \vec{x}.(\varphi \to \exists \vec{v}.\psi)$ with substitution σ only if $\exists \vec{v}.\psi\sigma$ is not entailed already
- Apply $\exists$-free rules first

Restricted chase computation:



$\rightsquigarrow$ restricted chase terminates, producing a finite model

Detecting Termination

Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Detecting Termination

Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise $\exists$, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

Detecting Termination

Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise $\exists$, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle($\star$)

Detecting Termination

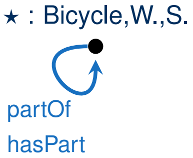
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise ∃, perform chase, check if it stops;
  give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "⋆") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle(⋆)



Detecting Termination

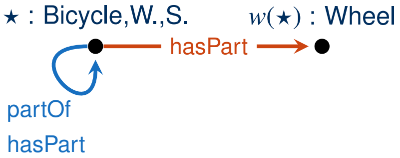
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise $\exists$, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle($\star$)



Detecting Termination

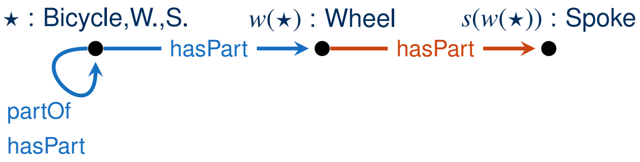
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise $\exists$, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle($\star$)



Detecting Termination

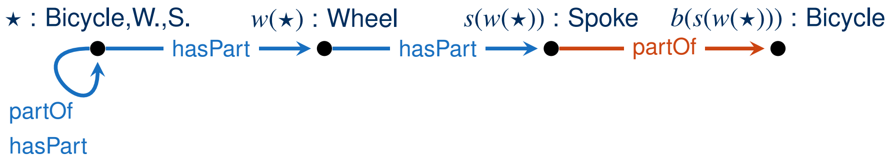
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise ∃, perform chase, check if it stops;
  give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle($\star$)



Detecting Termination

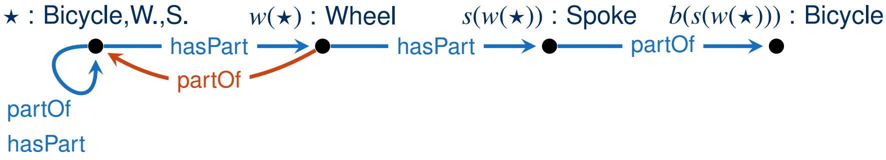
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise $\exists$, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle($\star$)



Detecting Termination

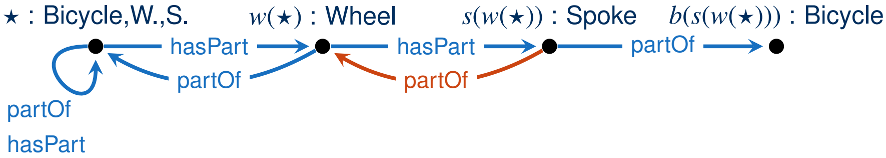
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise $\exists$, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle($\star$)



Detecting Termination

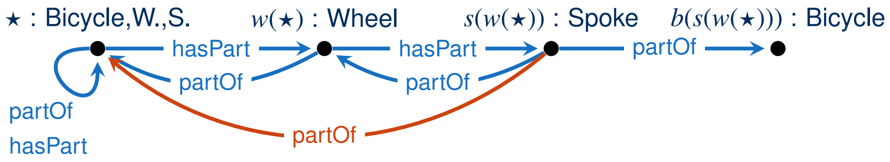
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise ∃, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "⋆") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle(⋆)



Detecting Termination

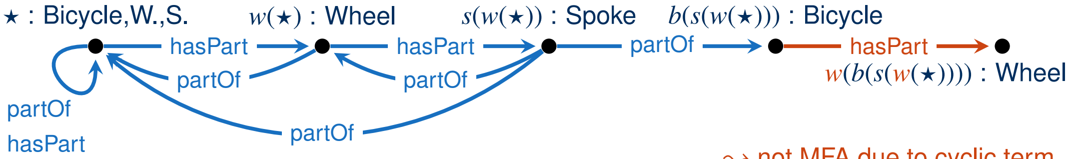
Fact: Whether the restricted chase will terminate on a set of rules is undecidable.

Many decidable and sufficient (but not necessary) criteria were proposed: acyclicity

Model-Faithful Acyclicity (MFA):

- Approach: skolemise ∃, perform chase, check if it stops; give up if a cyclic skolem term (with a repeated function symbol) appears
- Termination may depend on given facts, but: if the approach terminates on the critical instance (the set of all possible facts using a single constant "$\star$") then it terminates on all sets of facts

MFA check for the example: we show only derivations from Bicycle($\star$)



$\rightsquigarrow$ not MFA due to cyclic term

Restricted Acyclicity [IJCAI'17]

How to check (universal) termination for the restricted chase?

- Problem: restricted chase termination is not monotone

Restricted Acyclicity [IJCAI'17]

How to check (universal) termination for the restricted chase?

- Problem: restricted chase termination is not monotone
- In particular: it always terminates on the critical instance!

Restricted Acyclicity [IJCAI'17]

How to check (universal) termination for the restricted chase?

- Problem: restricted chase termination is not monotone
- In particular: it always terminates on the critical instance!

Idea: for each fact in the chase sequence, we can re-trace a weakest set of premises that must have been given to derive the fact

Restricted Acyclicity [IJCAI'17]

How to check (universal) termination for the restricted chase?

- Problem: restricted chase termination is not monotone
- In particular: it always terminates on the critical instance!

Idea: for each fact in the chase sequence, we can re-trace a weakest set of premises that must have been given to derive the fact

Example: If we see a fact Spoke$(s(w(\star)))$ then, certainly, we have previously derived facts Wheel$(w(\star))$, hasPart$(w(\star), s(w(\star)))$, Bicycle$(\star)$, hasPart$(\star, w(\star))$. Moreover, applying all ∃-free rules to this, we also know that partOf$(w(\star), \star)$, partOf$(s(w(\star)), w(\star))$, and partOf$(s(w(\star)), \star)$ must hold true.

Restricted Acyclicity [IJCAI'17]

How to check (universal) termination for the restricted chase?

- Problem: restricted chase termination is not monotone
- In particular: it always terminates on the critical instance!

Idea: for each fact in the chase sequence, we can re-trace a weakest set of premises that must have been given to derive the fact

Example: If we see a fact $\mathsf{Spoke}(s(w(\star)))$ then, certainly, we have previously derived facts $\mathsf{Wheel}(w(\star))$, $\mathsf{hasPart}(w(\star), s(w(\star)))$, $\mathsf{Bicycle}(\star)$, $\mathsf{hasPart}(\star, w(\star))$. Moreover, applying all ∃-free rules to this, we also know that $\mathsf{partOf}(w(\star), \star)$, $\mathsf{partOf}(s(w(\star)), w(\star))$, and $\mathsf{partOf}(s(w(\star)), \star)$ must hold true.

Restricted Model Faithful Acyclicity (RMFA):

- Perform a chase-like construction on the critical instance
- Only apply an ∃-rule with substitution σ if it is not blocked:
  (1) find minimal amount of certain knowledge required for match σ;
  (2) check if this minimal knowledge already entails the rule head.
- Give up if procedure does not stop before a cyclic term occurs

Theorem and Practice

Theorem [IJCAI'17]: Deciding if a set of rules is RMFA is 2ExpTime-complete even if the arity of predicates or the number of variables per rule is bounded.

- One can obtain slightly better bounds for DL ontologies (ExpTime)
- Criteria for making this tractable have been studied elsewhere, and seem to apply in many cases [ISWC'17]

Practice: We did not encounter major performance issues even for a prototype implementation. They arose mostly for rule sets that are artificially constructed to be "unreasonably" hard.

Real-World Coverage

RMFA succeeds in detecting that our example has a finite restricted chase.

How about other practical rule sets?

- OWL ontologies can often be transformed into existential rules
- We studied 1220 ontologies obtained from two sources (MOWLcorp and Oxford ontology corpus)
- We also applied a new sufficient criterion RMFC that shows non-termination

Real-World Coverage

RMFA succeeds in detecting that our example has a finite restricted chase.

How about other practical rule sets?

- OWL ontologies can often be transformed into existential rules
- We studied 1220 ontologies obtained from two sources (MOWLcorp and Oxford ontology corpus)
- We also applied a new sufficient criterion RMFC that shows non-termination

MFA (skolem chase termination)	884	(72.5%)	
RMFA (restricted chase termination)	936	(76.7%)	MFA + 52
RMFC (restricted chase non-termination)	239	(19.6%)	
Termination not decided by our methods	45	(3.6%)	

VLog: A Column-Based Rule Engine [AAAI'16]

VLog is an efficient implementation for large-scale rule reasoning

- Free and open source (C++)
- Command-line client and web interface
- Fully in-memory or using database back-end for input facts
- Supports existential quantifiers and arbitrary predicate arities

`https://github.com/karmaresearch/vlog`

Main reasoning algorithm: Bottom-up materialisation (chase)

- Semi-naive evaluation: only apply rules to matches that involve newly derived facts
- Column-store technology: store predicates in compressed vertical data structures
- Optimisations: highly efficient joins, redundancy avoidance, pre-computation, . . .

VLog: A Column-Based Rule Engine [AAAI'16]

VLog is an efficient implementation for large-scale rule reasoning

- Free and open source (C++)
- Command-line client and web interface
- Fully in-memory or using database back-end for input facts
- Supports existential quantifiers and arbitrary predicate arities

`https://github.com/karmaresearch/vlog`

Main reasoning algorithm: Bottom-up materialisation (chase)

- Semi-naive evaluation: only apply rules to matches that involve newly derived facts
- Column-store technology: store predicates in compressed vertical data structures
- Optimisations: highly efficient joins, redundancy avoidance, pre-computation, ...

Performance example: We extracted a Datalog rule set of 9,396 rules from DB-pedia, and applied it to a set of 112M facts from the same source. On a laptop, VLog computes 33M derived facts in 20sec, using 585MiB of RAM.

Restricted Chase in VLog

Since February 2018, VLog supports existential rule reasoning [unpublished]:

- Two chase variants: skolem chase and (1-parallel) resrticted chase
- Restricted chase gives priority to the execution of ∃-free rules

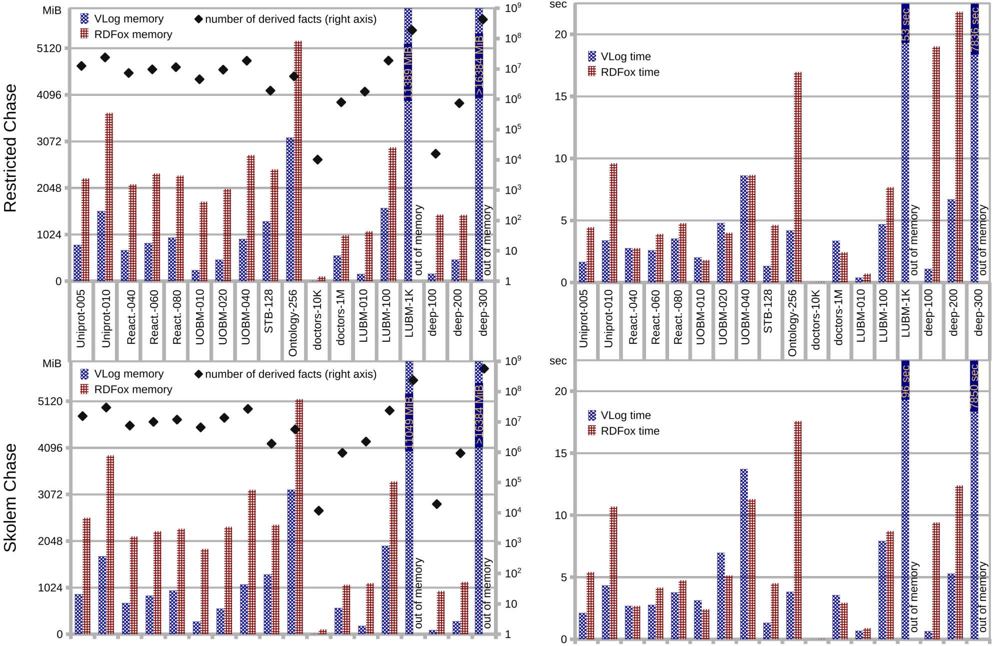
$\leadsto$ supports all rule sets that satisfy RMFA

We performed an extensive evaluation:

- using 18 challenging existential rule ontologies (many from a recent benchmark),
- producing several hundreds of billions of derived facts,
- on a laptop (2.2GHz Intel Core i7 CPU [4 cores], 16GB RAM 1600MHz DDR3).

We compare against RDFox, a leading rule engine

Results: Memory and Time



Conclusion

Existential rules are a powerful ontology and data analysis language

The chase is a versatile reasoning procedure, but it may not terminate

Summary of results:

- RMFA: the first criterion for restricted chase termination (TTBOOK)
- RMFC: the first criterion for non-termination of any chase (TTBOOK)
- VLog: a very memory-efficient and surprisingly fast existential rule reasoner

What's next? (potentially including student projects)

- Optimisations (we only do vanilla restricted chase so far)
- Applications, e.g., existential rule reasoning for automated deduction?
- Existential rules for enriched knowledge graphs/attributed logics?
- Adding numeric reasoning (linear programming, CSPs, . . .)
- Coping with (some types of) infinite models

References

[IJCAI'17] David Carral, Irina Dragoste, Markus Krötzsch: Restricted chase (non)termination for existential rules with disjunctions. In Carles Sierra, ed., Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), 922-928, 2017.

[ISWC'17] David Carral, Irina Dragoste, Markus Krötzsch: Tractable Query Answering for Expressive Ontologies and Existential Rules. In Claudia d'Amato et al., eds., Proceedings of the 16th International Semantic Web Conference (ISWC'17), volume 10587 of LNCS. Springer 2017.

[AAAI'16] Jacopo Urbani, Ceriel Jacobs, Markus Krötzsch: Column-Oriented Datalog Materialization for Large Knowledge Graphs. In Dale Schuurmans, Michael P. Wellman, eds., Proceedings of the 30th AAAI Conference on Artificial Intelligence, 258-264. AAAI Press 2016.