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 are sentences of the form

\[ \forall \vec{x}. \left( \varphi \rightarrow \exists \vec{v}. \psi \right) \]

where \( \varphi \) (body) and \( \psi \) (head) are conjunctions of atoms.
Existential Rules are sentences of the form

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

where $\varphi$ (body) and $\psi$ (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 are sentences of the form

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

where $\varphi$ (body) and $\psi$ (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, …
Example:  

\[
\text{Bicycle}(a) \quad (1)
\]
\[
\text{Bicycle}(x) \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \quad (2)
\]
\[
\text{Wheel}(x) \rightarrow \exists w. \text{hasPart}(x, w) \land \text{Spoke}(w) \quad (3)
\]
\[
\text{Spoke}(x) \rightarrow \exists u. \text{partOf}(x, u) \land \text{Bicycle}(u) \quad (4)
\]
\[
\text{hasPart}(x, y) \rightarrow \text{partOf}(y, x) \quad (5)
\]
\[
\text{partOf}(x, y) \land \text{partOf}(y, z) \rightarrow \text{partOf}(x, z) \quad (6)
\]

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

Example:

\[
\begin{align*}
\text{Bicycle}(a) & \quad \text{(1)} \\
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \quad \text{(2)} \\
\text{Wheel}(x) & \rightarrow \exists w. \text{hasPart}(x, w) \land \text{Spoke}(w) \quad \text{(3)} \\
\text{Spoke}(x) & \rightarrow \exists u. \text{partOf}(x, u) \land \text{Bicycle}(u) \quad \text{(4)} \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \quad \text{(5)} \\
\text{partOf}(x, y) \land \text{partOf}(y, z) & \rightarrow \text{partOf}(x, z) \quad \text{(6)}
\end{align*}
\]

(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:**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Bicycle(a)</td>
</tr>
<tr>
<td>(2)</td>
<td>Bicycle(x) → ∃v. hasPart(x, v) ∧ Wheel(v)</td>
</tr>
<tr>
<td>(3)</td>
<td>Wheel(x) → ∃w. hasPart(x, w) ∧ Spoke(w)</td>
</tr>
<tr>
<td>(4)</td>
<td>Spoke(x) → ∃u. partOf(x, u) ∧ Bicycle(u)</td>
</tr>
<tr>
<td>(5)</td>
<td>hasPart(x, y) → partOf(y, x)</td>
</tr>
<tr>
<td>(6)</td>
<td>partOf(x, y) ∧ partOf(y, z) → partOf(x, z)</td>
</tr>
</tbody>
</table>

*(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:

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicycle</td>
<td>$\exists v. \text{hasPart}(x, v) \land \text{Wheel}(v)$ (1)</td>
</tr>
<tr>
<td>Wheel</td>
<td>$\exists w. \text{hasPart}(x, w) \land \text{Spoke}(w)$ (2)</td>
</tr>
<tr>
<td>Spoke</td>
<td>$\exists u. \text{partOf}(x, u) \land \text{Bicycle}(u)$ (3)</td>
</tr>
<tr>
<td>hasPart</td>
<td>$\text{hasPart}(x, y) \rightarrow \text{partOf}(y, x)$ (5)</td>
</tr>
<tr>
<td>partOf</td>
<td>$\text{partOf}(x, y) \land \text{partOf}(y, z) \rightarrow \text{partOf}(x, z)$ (6)</td>
</tr>
</tbody>
</table>

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

Bottom-up model construction: “chasing the rules”

$\ a : \text{Bicycle} \quad \text{w}(a) : \text{Wheel}$

\[
\begin{array}{c}
\text{hasPart} \\
\end{array}
\]
The Chase

Example:

\[ \text{Bicycle}(a) \]  
\[ \text{Bicycle}(x) \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \]  
\[ \text{Wheel}(x) \rightarrow \exists w. \text{hasPart}(x, w) \land \text{Spoke}(w) \]  
\[ \text{Spoke}(x) \rightarrow \exists u. \text{partOf}(x, u) \land \text{Bicycle}(u) \]  
\[ \text{hasPart}(x, y) \rightarrow \text{partOf}(y, x) \]  
\[ \text{partOf}(x, y) \land \text{partOf}(y, z) \rightarrow \text{partOf}(x, z) \]

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

Bottom-up model construction: “chasing the rules”

\( a : \text{Bicycle} \quad w(a) : \text{Wheel} \quad s(w(a)) : \text{Spoke} \)

- \( \bullet \) hasPart \( \rightarrow \bullet \) hasPart \( \rightarrow \bullet \)
Example:

\[ \text{Bicycle}(a) \]  

\[ \text{Bicycle}(x) \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \]  

\[ \text{Wheel}(x) \rightarrow \exists w. \text{hasPart}(x, w) \land \text{Spoke}(w) \]  

\[ \text{Spoke}(x) \rightarrow \exists u. \text{partOf}(x, u) \land \text{Bicycle}(u) \]  

\[ \text{hasPart}(x, y) \rightarrow \text{partOf}(y, x) \]  

\[ \text{partOf}(x, y) \land \text{partOf}(y, z) \rightarrow \text{partOf}(x, z) \]  

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

Bottom-up model construction: “chasing the rules”

\( a : \text{Bicycle} \quad w(a) : \text{Wheel} \quad s(w(a)) : \text{Spoke} \quad b(s(w(a))) : \text{Bicycle} \)

\[ \bullet \quad \text{hasPart} \rightarrow \bullet \quad \text{hasPart} \rightarrow \bullet \quad \text{partOf} \rightarrow \bullet \]
The Chase

Example:

\[
\begin{align*}
\text{Bicycle}(a) & \quad (1) \\
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \quad (2) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{hasPart}(x, w) \land \text{Spoke}(w) \quad (3) \\
\text{Spoke}(x) & \rightarrow \exists u. \text{partOf}(x, u) \land \text{Bicycle}(u) \quad (4) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \quad (5) \\
\text{partOf}(x, y) \land \text{partOf}(y, z) & \rightarrow \text{partOf}(x, z) \quad (6)
\end{align*}
\]

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

Bottom-up model construction: “chasing the rules”

\[
\begin{align*}
a : \text{Bicycle} & \quad w(a) : \text{Wheel} & \quad s(w(a)) : \text{Spoke} & \quad b(s(w(a))) : \text{Bicycle}
\end{align*}
\]
The Chase

Example:

\[
\begin{align*}
  \text{Bicycle}(a) & \quad (1) \\
  \text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \quad (2) \\
  \text{Wheel}(x) & \rightarrow \exists w. \text{hasPart}(x, w) \land \text{Spoke}(w) \quad (3) \\
  \text{Spoke}(x) & \rightarrow \exists u. \text{partOf}(x, u) \land \text{Bicycle}(u) \quad (4) \\
  \text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \quad (5) \\
  \text{partOf}(x, y) \land \text{partOf}(y, z) & \rightarrow \text{partOf}(x, z) \quad (6)
\end{align*}
\]

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

Bottom-up model construction: “chasing the rules”

\(a\) : Bicycle \quad \(w(a)\) : Wheel \quad \(s(w(a))\) : Spoke \quad \(b(s(w(a)))\) : Bicycle

\[\text{partOf} \quad \text{partOf} \quad \text{hasPart} \quad \text{hasPart} \quad \text{partOf} \quad \text{partOf}\]
The Chase

Example:

\[
\begin{align*}
\text{Bicycle}(a) \quad & \quad (1) \\
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \quad (2) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{hasPart}(x, w) \land \text{Spoke}(w) \quad (3) \\
\text{Spoke}(x) & \rightarrow \exists u. \text{partOf}(x, u) \land \text{Bicycle}(u) \quad (4) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \quad (5) \\
\text{partOf}(x, y) \land \text{partOf}(y, z) & \rightarrow \text{partOf}(x, z) \quad (6)
\end{align*}
\]

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

Bottom-up model construction: “chasing the rules”

\[a : \text{Bicycle} \quad w(a) : \text{Wheel} \quad s(w(a)) : \text{Spoke} \quad b(s(w(a))) : \text{Bicycle}\]
The Chase

**Example:**

1. \( \text{Bicycle}(a) \)

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

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

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

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

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

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

**Bottom-up model construction:** “chasing the rules”

- \( a : \text{Bicycle} \)
- \( w(a) : \text{Wheel} \)
- \( s(w(a)) : \text{Spoke} \)
- \( b(s(w(a))) : \text{Bicycle} \)

Construction does not terminate . . .
Stopping the Chase

**Restricted Chase:**

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

**Restricted Chase:**

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

**Restricted chase computation:**

$a : \text{Bicycle}$

$\bullet$
Stopping the Chase

Restricted Chase:

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

Restricted chase computation:

$a : \text{Bicycle} \quad w(a) : \text{Wheel}$

$\rightarrow$ \hspace{1cm} \text{hasPart} \quad \rightarrow$

Markus Krötzsch, 21 Feb 2018
Recent Advances in Reasoningwith Existential Rules
Stopping the Chase

**Restricted Chase:**

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

**Restricted chase computation:**

\[ a : \text{Bicycle} \quad w(a) : \text{Wheel} \]

\[
\begin{array}{c}
\text{hasPart} \\
\text{partOf}
\end{array}
\]

Markus Krötzsch, 21 Feb 2018

Recent Advances in Reasoning with Existential Rules
Stopping the Chase

**Restricted Chase:**
- Apply rules $\forall \vec{x} \cdot (\varphi \rightarrow \exists \vec{v} \cdot \psi)$ with substitution $\sigma$ only if $\exists \vec{v} \cdot \psi \sigma$ is not entailed already.
- Apply $\exists$-free rules first.

**Restricted chase computation:**

- $a : \text{Bicycle}$
- $w(a) : \text{Wheel}$
- $s(w(a)) : \text{Spoke}$

Diagram:
- $a$ hasPart $w(a)$
- $w(a)$ hasPart $s(w(a))$
- $s(w(a))$ hasPart $b$

Markus Krötzsch, 21 Feb 2018
Recent Advances in Reasoning with Existential Rules
**Stopping the Chase**

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

### Restricted chase computation:

Let $a : \text{Bicycle}$, $w(a) : \text{Wheel}$, and $s(w(a)) : \text{Spoke}$.

- $a$ hasPart $w(a)$.
- $w(a)$ hasPart $s(w(a))$.
- $a$ partOf $w(a)$.
- $w(a)$ partOf $s(w(a))$.

The restricted chase terminates, producing a finite model.
Stopping the Chase

**Restricted Chase:**
- Apply rules $\forall \vec{x} (\varphi \rightarrow \exists \vec{y}. \psi)$ with substitution $\sigma$ only if $\exists \vec{y}. \psi \sigma$ is not entailed already
- Apply $\exists$-free rules first

**Restricted chase computation:**

$a : \text{Bicycle}$

$w(a) : \text{Wheel}$

$s(w(a)) : \text{Spoke}$

\[\begin{array}{c}
\text{partOf} & \text{hasPart} & \text{partOf}
\end{array}\]

Markus Krötzsch, 21 Feb 2018 Recent Advances in Reasoning with Existential Rules slide 4 of 13
Stopping the Chase

Restricted Chase:

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

Restricted chase computation:

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

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

Restricted Chase:

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

Restricted chase computation:

$a : \text{Bicycle}$  $w(a) : \text{Wheel}$  $s(w(a)) : \text{Spoke}$

$\rightarrow$ restricted chase terminates, producing a finite model
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.
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 “⋆”) 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 “⋆”) 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 “$*$”) then it terminates on all sets of facts

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

$*$ : Bicycle, W., S.
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($\star$)

$\star$: Bicycle,W.,S.

$w(\star)$: Wheel

partOf

hasPart

hasPart

Markus Krötzsch, 21 Feb 2018

Recent Advances in Reasoning with Existential Rules

slide 5 of 13
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 “⋆”) then it terminates on all sets of facts

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

$\star : \text{Bicycle},W.,S.$  
$w(\star) : \text{Wheel}$  
$s(w(\star)) : \text{Spoke}$

![Diagram showing 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 "\(*\)"") then it terminates on all sets of facts

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

\[
\begin{align*}
\star &: \text{Bicycle,}W.,S. \\
w(\star) &: \text{Wheel} \\
s(w(\star)) &: \text{Spoke} \\
b(s(w(\star))) &: \text{Bicycle}
\end{align*}
\]

- hasPart
- partOf
- hasPart
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($\star$)

$\star$: Bicycle,W.,S. $w(\star)$: Wheel $s(w(\star))$: Spoke $b(s(w(\star)))$: Bicycle

Markus Krötzsch, 21 Feb 2018
Recent Advances in Reasoning with Existential Rules
slide 5 of 13
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 “⋆”) then it terminates on all sets of facts

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

- $\star$: Bicycle,W.,S.
- $w(\star)$: Wheel
- $s(w(\star))$: Spoke
- $b(s(w(\star)))$: Bicycle

Markus Krötzsch, 21 Feb 2018

Recent Advances in Reasoning with Existential Rules
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 “⋆”) then it terminates on all sets of facts

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

⋆ : Bicycle, W., S.

\[ w(\star) : \text{Wheel} \quad s(w(\star)) : \text{Spoke} \quad b(s(w(\star))) : \text{Bicycle} \]
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($\star$)

$\star$: Bicycle,W.,S.  
$w(\star)$: Wheel  
$s(w(\star))$: Spoke  
$b(s(w(\star)))$: Bicycle

$\leadsto$ 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

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(⋆)))`, then, certainly, we have previously derived facts `Wheel(w(⋆))`, `hasPart(w(⋆),s(w(⋆)))`, `Bicycle(⋆)`, `hasPart(⋆,w(⋆))`.

Moreover, applying all ∃-free rules to this, we also know that `partOf(w(⋆),⋆)`, `partOf(s(w(⋆)),w(⋆))`, and `partOf(s(w(⋆)),⋆)` 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.
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 ⋆))
then, certainly, we have previously derived facts Wheel (w (⋆)), hasPart (w (⋆), s (w (⋆))), Bicycle (⋆), hasPart (⋆, w (⋆)).

Moreover, applying all ∃-free rules to this, we also know that partOf (w (⋆), ⋆), partOf (s (w (⋆)), w (⋆)), and partOf (s (w (⋆)), ⋆) 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.
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(⋆)))` then, certainly, we have previously derived facts `Wheel(w(⋆))`, `hasPart(w(⋆),s(w(⋆)))`, `Bicycle(⋆)`, `hasPart(⋆,w(⋆))`. Moreover, applying all ∃-free rules to this, we also know that `partOf(w(⋆),⋆)`, `partOf(s(w(⋆)),w(⋆))`, and `partOf(s(w(⋆)),⋆)` 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 Spoke(s(w(★))) then, certainly, we have previously derived facts Wheel(w(★)), hasPart(w(★), s(w(★))), Bicycle(★), hasPart(★, w(★)).
Moreover, applying all ∃-free rules to this, we also know that partOf(w(★), ★), partOf(s(w(★)), w(★)), and partOf(s(w(★)), ★) 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 $\text{Spoke}(s(w(\star)))$ then, certainly, we have previously derived facts $\text{Wheel}(w(\star)), \text{hasPart}(w(\star), s(w(\star))), \text{Bicycle}(\star), \text{hasPart}(\star, w(\star))$. Moreover, applying all $\exists$-free rules to this, we also know that $\text{partOf}(w(\star), \star), \text{partOf}(s(w(\star)), w(\star))$, and $\text{partOf}(s(w(\star)), \star)$ must hold true.

Restricted Model Faithful Acyclicity (RMFA):

• Perform a chase-like construction on the critical instance
• Only apply an $\exists$-rule with substitution $\sigma$ if it is not blocked:
  1. find minimal amount of certain knowledge required for match $\sigma$;
  2. check if this minimal knowledge already entails the rule head.
• Give up if procedure does not stop before a cyclic term occurs
Theorem [IJCAI’17]: Deciding if a set of rules is RMFA is $2\text{ExpTime}$-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 ($\text{ExpTime}$)
- 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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MFA (skolem chase termination)</td>
<td>884</td>
<td>(72.5%)</td>
</tr>
<tr>
<td>RMFA (restricted chase termination)</td>
<td>936</td>
<td>(76.7%)</td>
</tr>
<tr>
<td>RMFC (restricted chase non-termination)</td>
<td>239</td>
<td>(19.6%)</td>
</tr>
<tr>
<td>Termination not decided by our methods</td>
<td>45</td>
<td>(3.6%)</td>
</tr>
<tr>
<td>MFA + 52</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 DBpedia, 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.
Since February 2018, VLog supports existential rule reasoning [unpublished]:

- Two chase variants: skolem chase and (1-parallel) restricted chase
- Restricted chase gives priority to the execution of $\exists$-free rules

$\rightarrow$ 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
