Capturing Homomorphism-Closed Decidable Queries with Existential Rules

As presented at the Journée de la Logique, Nov 17, 2021, Montpellier

Camille Bourgaux\textsuperscript{ENS}, David Carral\textsuperscript{LIRMM}, Markus Krötzsch\textsuperscript{TUD}, Sebastian Rudolph\textsuperscript{TUD}, Michaël Thomazo\textsuperscript{ENS}

\textbf{ENS}: ENS, CNRS, PSL University & Inria, Paris
\textbf{LIRMM}: LIRMM, Inria, University of Montpellier, CNRS
\textbf{TUD}: TU Dresden
Ontology-Based Query Answering: query results = logical entailments over databases
Ontology-Based Query Answering: query results = logical entailments over databases

\[ \forall x, y, z. \ p(x, y) \land q(z, y) \rightarrow p(y, x) \]

\[ \exists x, y. \ p(x, y) \land p(y, x) \]

Inference via logical entailments:

\[ D \cup \Sigma \models q \]
Ontology-Based Query Answering: query results $= \text{logical entailments over databases}$

\[
\begin{align*}
\forall \ x,y,z. & \ p(x,y) \land q(z,y) \implies p(y,x) \\
\end{align*}
\]

$\exists \ x,y. \ p(x,y) \land p(y,x)$

Sebastian Rudolph, TU Dresden
Ontology-Based Query Answering: query results = logical entailments over databases

Database $\mathcal{D}$

Logical theory $\Sigma$

Query $q$

\[ \forall x, y, z. \quad p(x, y) \land q(z, y) \quad \rightarrow \quad p(y, x) \]

\[ \exists x, y. \quad p(x, y) \land p(y, x) \]

**Logic**

- DL-Lite
- Datalog
- Disjunctive Datalog
- Existential Rules
- Disjunctive Exist. Rules

**Data Complexity**

- AC$_0$
- P
- coNP
- r.e.

**Example rule**

- $p(x, y) \land p(y, z) \rightarrow p(x, z)$
- $\text{vertex}(x) \rightarrow \text{red}(x) \lor \text{green}(x) \lor \text{blue}(x)$
- $\text{human}(x) \rightarrow \exists y. \text{mother}(x, y) \land \text{human}(y)$
Can Datalog express every query that can be decided in polynomial time?

No!

- Datalog has no negation
- All Datalog queries are closed under homomorphism (hom-closed)
- Datalog cannot even express all hom-closed queries in P
  (less obvious: Dawar & Kreutzer discovered a hom-closed PTime query that cannot be expressed in Datalog [ICALP'08])
Can Datalog express every query that can be decided in polynomial time?

Yes or No

Algorithm

Datalog has no negation

• All Datalog queries are closed under homomorphism (hom-closed)

• Datalog cannot even express all hom-closed queries in P

(less obvious: Dawar & Kreutzer discovered a hom-closed PTime query that cannot be expressed in Datalog [ICALP'08])
Can Datalog express every query that can be decided in polynomial time?

Yes or No

polytime Algorithm

Yes

or

No

polytime Algorithm
(Turing Machine)

p (10, 11) p (11, 01) q (01, 00)

Serialisation
Expressing Queries – Warm-Up

Can Datalog express every query that can be decided in polynomial time?

Yes or No

• Datalog has no negation
• All Datalog queries are closed under homomorphism (hom-closed)

• Datalog cannot even express all hom-closed queries in P
  (less obvious: Dawar & Kreutzer discovered a hom-closed PTime query that cannot be expressed in Datalog [ICALP'08])
Expressing Queries – Warm-Up

Can Datalog express every query that can be decided in polynomial time?

So can it?
Expressing Queries – Warm-Up

Can Datalog express every query that can be decided in polynomial time?

So can it? **No!**
Expressing Queries – Warm-Up

Can Datalog express every query that can be decided in polynomial time?

So can it?  **No!**

- Datalog has no negation
  - all Datalog queries are closed under homomorphism (hom-closed)
Expressing Queries – Warm-Up

Can Datalog express every query that can be decided in polynomial time?

So can it?  **No!**

- Datalog has no negation
  - ~ all Datalog queries are closed under homomorphism (hom-closed)
Can Datalog express every query that can be decided in polynomial time?

So can it? **No!**

- Datalog has no negation
  - So all Datalog queries are closed under homomorphism (hom-closed)

- Datalog cannot even express all hom-closed queries in P
  - (less obvious: Dawar & Kreutzer discovered a hom-closed PTime query that cannot be expressed in Datalog [ICALP’08])

Sebastian Rudolph, TU Dresden  
Capturing Homomorphism-Closed Decidable Queries with Existential Rules  
slide 3 of 20
Marnette [PODS 2009] showed the following general result:

**Theorem:** For every skolem-chase terminating $\Sigma$ and concrete database $D$, the skolem chase over $\Sigma$ and $D$ is polynomial in the size of $D$. The data complexity of BCQ entailment over any such $\Sigma$ is P-complete.
Marnette [PODS 2009] showed the following general result:

**Theorem:** For every skolem-chase terminating $\Sigma$ and concrete database $\mathcal{D}$, the skolem chase over $\Sigma$ and $\mathcal{D}$ is polynomial in the size of $\mathcal{D}$. The data complexity of BCQ entailment over any such $\Sigma$ is P-complete.

This insight can be taken further
[Krötzsch & R., IJCAI’11; Zhang, Zhang & You, AAAI’15]

**Theorem:** For every skolem-chase terminating $\Sigma$ and BCQ $q$, there is a set of Datalog rules $\Sigma'$ and BCQ $q'$ such that $\{D | D, \Sigma \models q\} = \{D | D, \Sigma' \models q'\}$. 
Toward Higher Expressivity

**Logic**

- Datalog
- Skolem-Chase Terminating Existential Rules

**Expressive power**

- \(\subseteq\) hom-closed P
- \(\subseteq\) hom-closed P

**Existential Rules**

- = hom-closed r.e.
- [R. & Thomazo, IJCAI’15]
Toward Higher Expressivity

**Logic**
- Datalog
- Skolem-Chase Terminating Existential Rules

**Expressive power**
- $\subseteq$ hom-closed P

**Existential Rules**
- $=$ hom-closed r.e.
  - [R. & Thomazo, IJCAI’15]

Extend the formalism to capture more queries.
We want decidability: require chase termination.
Toward Higher Expressivity

**Logic**

- Datalog
- Skolem-Chase Terminating Existential Rules
- Standard-Chase Terminating Existential Rules

**Expressive power**

- $\subseteq$ hom-closed P
- $\subseteq$ hom-closed P
- ?

**Existential Rules**

- $= \text{hom-closed r.e.}$
  
  [R. & Thomazo, IJCAI’15]

Extend the formalism to capture more queries.
We want decidability: require chase termination.
Toward Higher Expressivity

**Logic**
- Datalog
- Skolem-Chase Terminating Existential Rules
- Standard-Chase Terminating Existential Rules
- Core-Chase Terminating Existential Rules

**Expressive power**
- $\subseteq$ hom-closed $P$
- $\subseteq$ hom-closed $P$
- ?
- ?

Existential Rules = hom-closed r.e.  
[R. & Thomazo, IJCAI’15]

Extend the formalism to capture more queries.  
We want decidability: require chase termination.
Toward Higher Expressivity

**Logic**

- Datalog
- Skolem-Chase Terminating Existential Rules
- Standard-Chase Terminating Existential Rules
- Core-Chase Terminating Existential Rules
- Standard-Chase Terminating Disjunctive Exist. Rules
- Core-Chase Terminating Disjunctive Exist. Rules
- Existential Rules

**Expressive power**

- $\subseteq$ hom-closed P
- $\subseteq$ hom-closed P
- ?
- ?
- ?
- ?
- = hom-closed r.e.
  
  [R. & Thomazo, IJCAI’15]

Extend the formalism to capture more queries.
We want decidability: require chase termination.
## Toward Higher Expressivity

<table>
<thead>
<tr>
<th>Logic</th>
<th>Expressive power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datalog</td>
<td>$\subseteq$ hom-closed P</td>
</tr>
<tr>
<td>Skolem-Chase Terminating Existential Rules</td>
<td>$\subseteq$ hom-closed P</td>
</tr>
<tr>
<td>Standard-Chase Terminating Existential Rules</td>
<td>?</td>
</tr>
<tr>
<td>Core-Chase Terminating Existential Rules</td>
<td>?</td>
</tr>
<tr>
<td>Standard-Chase Terminating Disjunctive Exist. Rules</td>
<td>?</td>
</tr>
<tr>
<td>Core-Chase Terminating Disjunctive Exist. Rules</td>
<td>?</td>
</tr>
<tr>
<td>Existential Rules</td>
<td>= hom-closed r.e. [R. &amp; Thomazo, IJCAI’15]</td>
</tr>
</tbody>
</table>

Extend the formalism to capture more queries.
We want decidability: require chase termination.

**Standard Chase?**  **Disjunctive??**
Rule $\rho = \forall x. (\beta[x] \rightarrow \bigvee_{i=1}^{k} \exists y_i. \eta_i[x_i, y_i])$

**Definition:** Rule $\rho$ is applicable to $D$ if:

1. there is a function $h : x \rightarrow \text{dom}(D)$ such that $h(\varphi) \subseteq D$ (a match),
2. for every $i \in \{1, \ldots, k\}$, there is no function $h' : x \cup y_i \rightarrow \text{dom}(D)$ with $h'(x) = h(x)$ for all $x \in x$ and $h'(\eta_i) \subseteq D$.

$\{D_1, \ldots, D_k\}$ is the result of applying $\rho$ to $D$ under $h$ if $D_i = D \cup \hat{h}_i(\eta_i)$ and:

- $\hat{h}_i(x) = h(x)$ for all $x \in x$,
- $\hat{h}_i(y)$ is a fresh null for all $y \in y_i$.

- Chase sequence becomes chase tree giving rise to a universal model set containing every structures obtained from taking union over tree branches
- Query is entailed iff it matches every structure in this set
- Disjunctive chase terminates if the produced chase tree is finite
- Generalization of the (nondisjunctive) standard chase
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

\[
\begin{align*}
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w) \\
\text{properPartOf}(x, y) & \rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) & \rightarrow \text{hasPart}(y, x)
\end{align*}
\]
Will it terminate?

\[ D = \{ \text{Bicycle}(c) \} \]

\[
\begin{align*}
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w)
\end{align*}
\]

\[
\begin{align*}
\text{properPartOf}(x, y) & \rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) & \rightarrow \text{hasPart}(y, x)
\end{align*}
\]

Applying the standard chase may yield:
Will it terminate?

\[ D = \{ \text{Bicycle}(c) \} \]
\[ \text{Bicycle}(x) \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \]
\[ \text{Wheel}(x) \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w) \]
\[ \text{properPartOf}(x, y) \rightarrow \text{partOf}(x, y) \]
\[ \text{hasPart}(x, y) \rightarrow \text{partOf}(y, x) \]
\[ \text{partOf}(x, y) \rightarrow \text{hasPart}(y, x) \]

**Applying the standard chase may yield:**
\[ D_1 = D \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

**Applying the standard chase may yield:**

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

Applying the standard chase may yield:

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ \mathcal{D}_3 = \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]
Will it terminate?

\[ D = \{ \text{Bicycle}(c) \} \]

Bicycle\( (x) \) → ∃v. hasPart\( (x, v) \) ∧ Wheel\( (v) \)

Wheel\( (x) \) → ∃w. properPartOf\( (x, w) \) ∧ Bicycle\( (w) \)

properPartOf\( (x, y) \) → partOf\( (x, y) \)

hasPart\( (x, y) \) → partOf\( (y, x) \)

partOf\( (x, y) \) → hasPart\( (y, x) \)

Applying the standard chase may yield:

\[ D_1 = D \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ D_2 = D_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ D_3 = D_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]

\[ D_4 = D_3 \cup \{ \text{hasPart}(n_2, n_3), \text{Wheel}(n_3) \} \]
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

Applying the standard chase may yield:

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ \mathcal{D}_3 = \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]

\[ \mathcal{D}_4 = \mathcal{D}_3 \cup \{ \text{hasPart}(n_2, n_3), \text{Wheel}(n_3) \} \]

\[ \mathcal{D}_5 = \mathcal{D}_4 \cup \{ \text{partOf}(n_1, n_2) \} \]
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

Applying the standard chase may yield:

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ \mathcal{D}_3 = \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]

\[ \mathcal{D}_4 = \mathcal{D}_3 \cup \{ \text{hasPart}(n_2, n_3), \text{Wheel}(n_3) \} \]

\[ \mathcal{D}_5 = \mathcal{D}_4 \cup \{ \text{partOf}(n_1, n_2) \} \]

\[ \mathcal{D}_6 = \mathcal{D}_5 \cup \{ \text{hasPart}(n_2, n_1) \} \]
Will it terminate?

\[ \mathcal{D} = \{\text{Bicycle}(c)\} \]

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

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

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

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

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

Applying the standard chase may yield:

\[ \mathcal{D}_1 = \mathcal{D} \cup \{\text{hasPart}(c, n_1), \text{Wheel}(n_1)\} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{\text{partOf}(n_1, c)\} \]

\[ \mathcal{D}_3 = \mathcal{D}_2 \cup \{\text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2)\} \]

\[ \mathcal{D}_4 = \mathcal{D}_3 \cup \{\text{hasPart}(n_2, n_3), \text{Wheel}(n_3)\} \]

\[ \mathcal{D}_5 = \mathcal{D}_4 \cup \{\text{partOf}(n_1, n_2)\} \]

\[ \mathcal{D}_6 = \mathcal{D}_5 \cup \{\text{hasPart}(n_2, n_1)\} \]

\[ \mathcal{D}_7 = \mathcal{D}_6 \cup \{\text{partOf}(n_3, n_2)\} \]

\[ \mathcal{D}_8 = \mathcal{D}_7 \cup \{\text{partOf}(n_3, n_4), \text{Bicycle}(n_4)\} \]

The chase can continue forever...
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

**Applying the standard chase may yield:**

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ \mathcal{D}_3 = \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]

\[ \mathcal{D}_4 = \mathcal{D}_3 \cup \{ \text{hasPart}(n_2, n_3), \text{Wheel}(n_3) \} \]

\[ \mathcal{D}_5 = \mathcal{D}_4 \cup \{ \text{partOf}(n_1, n_2) \} \]

\[ \mathcal{D}_6 = \mathcal{D}_5 \cup \{ \text{hasPart}(n_2, n_1) \} \]

\[ \mathcal{D}_7 = \mathcal{D}_6 \cup \{ \text{partOf}(n_3, n_2) \} \]

\[ \mathcal{D}_8 = \mathcal{D}_7 \cup \{ \text{partOf}(n_3, n_4), \text{Bicycle}(n_4) \} \]
Will it terminate?

\[ D = \{ \text{Bicycle}(c) \} \]

\[
\begin{align*}
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w)
\end{align*}
\]

\[
\begin{align*}
\text{properPartOf}(x, y) & \rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) & \rightarrow \text{hasPart}(y, x)
\end{align*}
\]

Applying the standard chase may yield:

\[
\begin{align*}
D_1 = D \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} & \quad D_5 = D_4 \cup \{ \text{partOf}(n_1, n_2) \} \\
D_2 = D_1 \cup \{ \text{partOf}(n_1, c) \} & \quad D_6 = D_5 \cup \{ \text{hasPart}(n_2, n_1) \} \\
D_3 = D_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} & \quad D_7 = D_6 \cup \{ \text{partOf}(n_3, n_2) \} \\
D_4 = D_3 \cup \{ \text{hasPart}(n_2, n_3), \text{Wheel}(n_3) \} & \quad D_8 = D_7 \cup \{ \text{partOf}(n_3, n_4), \text{Bicycle}(n_4) \}
\end{align*}
\]

The chase can continue forever . . .
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ \mathcal{D}_3 = \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]

\[ \mathcal{D}_4 = \mathcal{D}_3 \cup \{ \text{partOf}(n_1, n_2) \} \]

\[ \mathcal{D}_5 = \mathcal{D}_4 \cup \{ \text{hasPart}(n_2, n_1) \} \]

No further rules are applicable. The chase terminates.
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:
Will it terminate?

\[ D = \{ \text{Bicycle}(c) \} \]

\[
\begin{align*}
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w)
\end{align*}
\]

\[
\begin{align*}
\text{properPartOf}(x, y) & \rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) & \rightarrow \text{hasPart}(y, x)
\end{align*}
\]

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:

\[ D_1 = D \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \quad \text{Bicycle}(x) \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \]

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

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

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

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

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

\[
\begin{align*}
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w) \\
\text{properPartOf}(x, y) & \rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) & \rightarrow \text{hasPart}(y, x)
\end{align*}
\]

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:

\[
\begin{align*}
\mathcal{D}_1 &= \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \}
\end{align*}
\]
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

\[
\begin{align*}
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w) \\
\text{properPartOf}(x, y) & \rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) & \rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) & \rightarrow \text{hasPart}(y, x)
\end{align*}
\]

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:

\[
\begin{align*}
\mathcal{D}_1 &= \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \\
\mathcal{D}_4 &= \mathcal{D}_3 \cup \{ \text{partOf}(n_1, n_2) \}
\end{align*}
\]
Will it terminate?

\[ D = \{ \text{Bicycle}(c) \} \]

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

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

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

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

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

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:

\[ D_1 = D \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ D_2 = D_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ D_3 = D_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]

\[ D_4 = D_3 \cup \{ \text{partOf}(n_1, n_2) \} \]

\[ D_5 = D_4 \cup \{ \text{hasPart}(n_2, n_1) \} \]

No further rules are applicable. The chase terminates.
Will it terminate?

\[ \mathcal{D} = \{ \text{Bicycle}(c) \} \]

\begin{align*}
\text{Bicycle}(x) & \rightarrow \exists v. \text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) & \rightarrow \exists w. \text{properPartOf}(x, w) \land \text{Bicycle}(w)
\end{align*}

\text{properPartOf}(x, y) \rightarrow \text{partOf}(x, y)

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

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

Prioritizing Datalog rules (aka the Datalog-first strategy) yields:

\[ \mathcal{D}_1 = \mathcal{D} \cup \{ \text{hasPart}(c, n_1), \text{Wheel}(n_1) \} \]

\[ \mathcal{D}_2 = \mathcal{D}_1 \cup \{ \text{partOf}(n_1, c) \} \]

\[ \mathcal{D}_3 = \mathcal{D}_2 \cup \{ \text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2) \} \]

\[ \mathcal{D}_4 = \mathcal{D}_3 \cup \{ \text{partOf}(n_1, n_2) \} \]

\[ \mathcal{D}_5 = \mathcal{D}_4 \cup \{ \text{hasPart}(n_2, n_1) \} \]

No further rules are applicable. The chase terminates.
Chase termination

Some observations:

- Termination is dependent on order of rule applications for standard chase (even when prioritizing Datalog), but not for skolem chase (and neither for core chase).
- Termination always depends on the concrete database instance.
Chase termination

Some observations:

- Termination is dependent on order of rule applications for standard chase (even when prioritizing Datalog), but not for skolem chase (and neither for core chase)
- Termination always depends on the concrete database instance

We can define rule classes based on their termination behaviour:

<table>
<thead>
<tr>
<th>Termination on . . .</th>
<th>instance $\mathcal{D}$</th>
<th>all instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skolem chase</td>
<td>$\text{CT}^{\text{sk}}_{\mathcal{D}}$</td>
<td>$\text{CT}^{\text{sk}}_{\forall}$</td>
</tr>
<tr>
<td>Standard chase (all/some strategies)</td>
<td>$\text{CT}^{\text{std}}<em>{\mathcal{D}} / \text{CT}^{\text{std}}</em>{\exists}$</td>
<td>$\text{CT}^{\text{std}}_{\forall / \exists}$</td>
</tr>
<tr>
<td>Datalog-first chase (all/some strategies)</td>
<td>$\text{CT}^{\text{dif}}<em>{\mathcal{D}} / \text{CT}^{\text{dif}}</em>{\exists}$</td>
<td>$\text{CT}^{\text{dif}}_{\forall / \exists}$</td>
</tr>
<tr>
<td>Core chase</td>
<td>$\text{CT}^{\text{cor}}_{\mathcal{D}}$</td>
<td>$\text{CT}^{\text{cor}}_{\forall}$</td>
</tr>
</tbody>
</table>

We focus on the all-instance case. The following syntacting inclusions are known:

$$\text{CT}^{\text{sk}}_{\forall} \subset \text{CT}^{\text{std}}_{\forall / \exists} \subset \text{CT}^{\text{dif}}_{\forall / \exists} = \text{CT}^{\text{std}}_{\forall / \exists} \subset \text{CT}^{\text{cor}}_{\forall}$$
### Main Result

<table>
<thead>
<tr>
<th>Logic</th>
<th>Expressive power</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logic</strong></td>
<td></td>
</tr>
<tr>
<td>Datalog</td>
<td>( \subseteq ) hom-closed P</td>
</tr>
<tr>
<td>Skolem-Chase Terminating Existential Rules</td>
<td>( CT^\text{sk}_\forall ) ( \subseteq ) hom-closed P</td>
</tr>
<tr>
<td>Standard-Chase Terminating Existential Rules</td>
<td>( CT^\text{std}_\forall ) ?</td>
</tr>
<tr>
<td>Core-Chase Terminating Existential Rules</td>
<td>( CT^\text{cor}_\forall ) ?</td>
</tr>
<tr>
<td>Standard-Chase Terminating Disjunctive Exist. Rules</td>
<td>( CT^\text{std,\lor}_\forall ) ?</td>
</tr>
<tr>
<td>Core-Chase Terminating Disjunctive Exist. Rules</td>
<td>( CT^\text{cor,\lor}_\forall ) ?</td>
</tr>
</tbody>
</table>
### Main Result

<table>
<thead>
<tr>
<th>Logic</th>
<th>Expressive power</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Datalog</strong></td>
<td>⊂ hom-closed $P$</td>
</tr>
<tr>
<td><strong>Skolem-Chase Terminating Existential Rules</strong></td>
<td>$CT_{sk}^\vee$ ⊂ hom-closed $P$</td>
</tr>
<tr>
<td><strong>Standard-Chase Terminating Existential Rules</strong></td>
<td>$CT_{std}^\vee = \text{hom-closed decidable}$</td>
</tr>
<tr>
<td><strong>Core-Chase Terminating Existential Rules</strong></td>
<td>$CT_{cor}^\vee ?$</td>
</tr>
<tr>
<td><strong>Standard-Chase Terminating Disjunctive Exist. Rules</strong></td>
<td>$CT_{std,\vee}^\vee ?$</td>
</tr>
<tr>
<td><strong>Core-Chase Terminating Disjunctive Exist. Rules</strong></td>
<td>$CT_{cor,\vee}^\vee ?$</td>
</tr>
</tbody>
</table>

**Main Theorem:** Existential rules for which the standard chase terminates (on every input and with every fair rule application order) can express all decidable hom-closed queries.
### Main Result

<table>
<thead>
<tr>
<th>Logic</th>
<th>Expressive power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datalog</td>
<td>$\subset \text{hom-closed P}$</td>
</tr>
<tr>
<td>Skolem-Chase Terminating Existential Rules</td>
<td>$\mathit{CT}_{\text{sk},V}$ $\subset \text{hom-closed P}$</td>
</tr>
<tr>
<td>Standard-Chase Terminating Existential Rules</td>
<td>$\mathit{CT}_{\text{std},V} = \text{hom-closed decidable}$</td>
</tr>
<tr>
<td>Core-Chase Terminating Existential Rules</td>
<td>$\mathit{CT}_{\text{cor},V} = \text{hom-closed decidable}$</td>
</tr>
<tr>
<td>Standard-Chase Terminating Disjunctive Exist. Rules</td>
<td>$\mathit{CT}_{\text{std},V} = \text{hom-closed decidable}$</td>
</tr>
<tr>
<td>Core-Chase Terminating Disjunctive Exist. Rules</td>
<td>$\mathit{CT}_{\text{cor},V} = \text{hom-closed decidable}$</td>
</tr>
</tbody>
</table>

**Main Theorem:** Existential rules for which the standard chase terminates (on every input and with every fair rule application order) can express all decidable hom-closed queries.

Hence, no hom-closed OBQA approach for which query answering is decidable can express more queries.
Proof Plan

(1) Disjunctive existential rules can express all decidable hom-closed queries

(2) Standard-chase terminating disjunctive existential rules can express all decidable hom-closed queries

(3) Standard-chase terminating (non-disjunctive) existential rules can express all decidable hom-closed queries
We can use disjunctions to guess how database elements are ordered:

\[
\text{elem}(x) \land \text{elem}(y) \rightarrow (x \approx y) \lor (x \not\approx y)
\]

\[
(x \not\approx y) \rightarrow (x < y) \lor (y < x)
\]

\ldots
Order!

We can use disjunctions to guess how database elements are ordered:

\[
\text{elem}(x) \land \text{elem}(y) \rightarrow (x \approx y) \lor (x \not\approx y)
\]

\[
(x \not\approx y) \rightarrow (x < y) \lor (y < x)
\]

\[
\ldots
\]

Many possible models arise:
We can use disjunctions to guess how database elements are ordered:

\[ \text{elem}(x) \land \text{elem}(y) \rightarrow (x \approx y) \lor (x \not\approx y) \]

\[ (x \not\approx y) \rightarrow (x < y) \lor (y < x) \]

\[ \ldots \]

Many possible models arise:
Order!

We can use disjunctions to guess how database elements are ordered:

\[ \text{elem}(x) \land \text{elem}(y) \rightarrow (x \approx y) \lor (x \not\approx y) \]
\[ (x \not\approx y) \rightarrow (x < y) \lor (y < x) \]

\[ \ldots \]

Many possible models arise:
We can use disjunctions to guess how database elements are ordered:

\[
\text{elem}(x) \land \text{elem}(y) \rightarrow (x \approx y) \lor (x \not\approx y)
\]

\[
(x \not\approx y) \rightarrow (x < y) \lor (y < x)
\]

\[
\ldots
\]

Many possible models arise:
We can use disjunctions to guess how database elements are ordered:

\[ \text{elem}(x) \land \text{elem}(y) \rightarrow (x \approx y) \lor (x \not\approx y) \]
\[ (x \not\approx y) \rightarrow (x < y) \lor (y < x) \]

Many possible models arise:
We can use disjunctions to guess how database elements are ordered:

\[ \text{elem}(x) \land \text{elem}(y) \rightarrow (x \approx y) \lor (x \not\approx y) \]

\[ (x \not\approx y) \rightarrow (x < y) \lor (y < x) \]

...
Completion

We need to know which relations **do not** hold, so we guess them too:

\[
\text{elem}(x_1) \land \text{elem}(x_2) \rightarrow p(x_1, x_2) \lor \bar{p}(x_1, x_2)
\]

\[
\text{elem}(x_1) \land \text{elem}(x_2) \rightarrow q(x_1, x_2) \lor \bar{q}(x_1, x_2)
\]

\[
\vdots
\]
Completion

We need to know which relations **do not** hold, so we guess them too:

\[
\text{elem}(x_1) \wedge \text{elem}(x_2) \rightarrow p(x_1, x_2) \lor \bar{p}(x_1, x_2)
\]

\[
\text{elem}(x_1) \wedge \text{elem}(x_2) \rightarrow q(x_1, x_2) \lor \bar{q}(x_1, x_2)
\]

\[
\ldots
\]

For each possible order, we obtain many possible completions:
We need to know which relations **do not** hold, so we guess them too:

\[
\begin{align*}
\text{elem}(x_1) \land \text{elem}(x_2) & \rightarrow p(x_1, x_2) \lor \overline{p}(x_1, x_2) \\
\text{elem}(x_1) \land \text{elem}(x_2) & \rightarrow q(x_1, x_2) \lor \overline{q}(x_1, x_2)
\end{align*}
\]

For each possible order, we obtain many possible completions:
This is not a suitable successor relation.
(since < is transitive)
This is not a suitable successor relation.
(since $<$ is transitive)

**Solution:** Construct a tree by tracing directed paths:
Tree to tape(s)
Sebastian Rudolph, TU Dresden

Capturing Homomorphism-Closed Decidable Queries with Existential Rules

slide 15 of 20
Tree to tape(s)
Tree to tape(s)
Tree to tape(s)
(1) Disjunctive existential rules can express all decidable hom-closed queries

(2) Standard-chase terminating disjunctive existential rules can express all decidable hom-closed queries

(3) Standard-chase terminating (non-disjunctive) existential rules can express all decidable hom-closed queries
Enforcing Termination

Some models are infinite:

But: We can write a Datalog query to detect when this occurs.
Enforcing Termination

Some models are infinite:

But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:
Enforcing Termination

Some models are infinite:

But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:
1. Prepare an “inactive” structure for which any further model expansion is redundant
Enforcing Termination

Some models are infinite:

But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:

1. Prepare an “inactive” structure for which any further model expansion is redundant
2. Connect all elements to this structure
Enforcing Termination

Some models are infinite:

But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:

1. Prepare an “inactive” structure for which any further model expansion is redundant
2. Connect all elements to this structure
3. Make the structure “active” when a termination problem is detected
Proof Plan

(1) Disjunctive existential rules can express all decidable hom-closed queries

(2) Standard-chase terminating disjunctive existential rules can express all decidable hom-closed queries

(3) Standard-chase terminating (non-disjunctive) existential rules can express all decidable hom-closed queries
Simulating Disjunctive Datalog in Existential Rules

Idea: Represent possible worlds with existentially introduced elements

\[ p(x) \rightarrow q(x) \lor r(x) \]

Challenge: Creation of fresh “worlds” must terminate

\[ \sim \text{ adapt chase-terminating set-modelling technique of } [\text{Krötzsch, Marx, R., ICDT 2019}] \]
Conclusions

Results:

- Chase-terminating existential rules characterise the decidable hom-closed queries.
- Neither disjunctions nor better chase algorithms can increase expressivity.
- New techniques to order databases, to enforce termination, and to simulate disjunctive reasoning with existential rules.

Bonus Theorem (not in the talk): If a language $Q$ of ontology-based queries captures all decidable hom-closed queries and query answering is decidable for $Q$, then $Q$ is not recursively enumerable.

And indeed universally standard chase-terminating existential rule sets are not \cite{Grahne & Onet, Fund. Inf. 2018}.

Open questions:

- Even if expressivity is the same, can some OBQA approaches lead to lower complexities for certain (PTime) queries?
- Natural characterisations for OBQA approaches with (semi-)decidable syntax?