Capturing Homomorphism-Closed Decidable Queries with Existential Rules

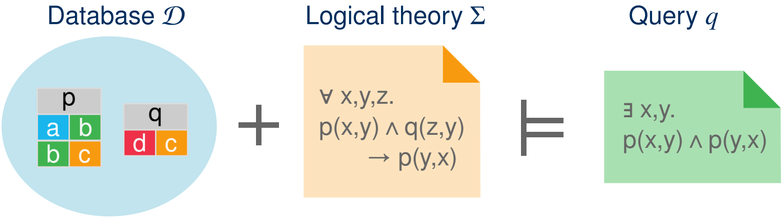
As presented at the Journée de la Logique, Nov 17, 2021, Montpellier

Camille Bourgaux[ENS], David Carral[LIRMM], Markus Krötzsch[TUD], Sebastian Rudolph[TUD], Michaël Thomazo[ENS]

ENS: ENS, CNRS, PSL University & Inria, Paris
LIRMM: LIRMM, Inria, University of Montpellier, CNRS
TUD: TU Dresden

DeciGUT
European Research Council
Established by the European Commission

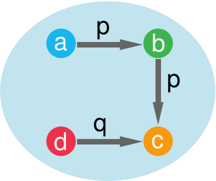
Ontology-Based Query Answering: query results = logical entailments over databases



KR for Data Access

Ontology-Based Query Answering: query results = logical entailments over databases

Database $\mathcal{D}$ — Logical theory Σ — Query q



$+$

$\forall$ x,y,z.
p(x,y) $\wedge$ q(z,y)
$\rightarrow$ p(y,x)

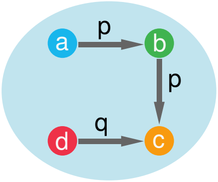
$\models$

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

KR for Data Access

Ontology-Based Query Answering: query results = logical entailments over databases

Database $\mathcal{D}$



Logical theory Σ

∀ x,y,z.
p(x,y) ∧ q(z,y)
→ p(y,x)

Query q

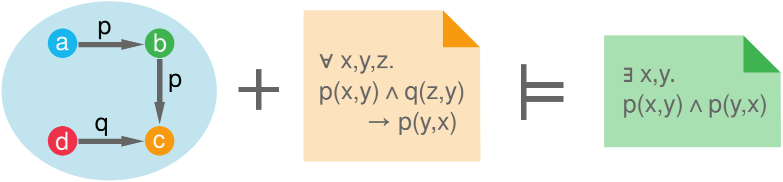
∃ x,y.
p(x,y) ∧ p(y,x)

$$\underbrace{\mathcal{D}}_{\textbf{input}} + \underbrace{\Sigma \models q}_{\textbf{fixed = OMQ}}$$

KR for Data Access

Ontology-Based Query Answering: query results = logical entailments over databases

Database $\mathcal{D}$ Logical theory Σ Query q



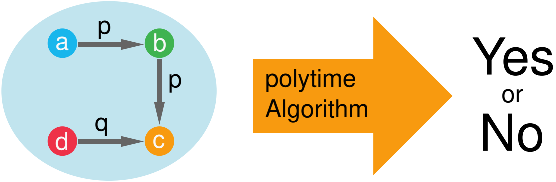
Logic	Data Complexity	Example rule
DL-Lite	AC_0	
Datalog	P	$\mathsf{p}(x,y) \land \mathsf{p}(y,z) \to \mathsf{p}(x,z)$
Disjunctive Datalog	coNP	$\mathsf{vertex}(x) \to \mathsf{red}(x) \lor \mathsf{green}(x) \lor \mathsf{blue}(x)$
Existential Rules	r.e.	$\mathsf{human}(x) \to \exists y.\mathsf{mother}(x,y) \land \mathsf{human}(y)$
Disjunctive Exist. Rules	r.e.	

Expressing Queries – Warm-Up

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

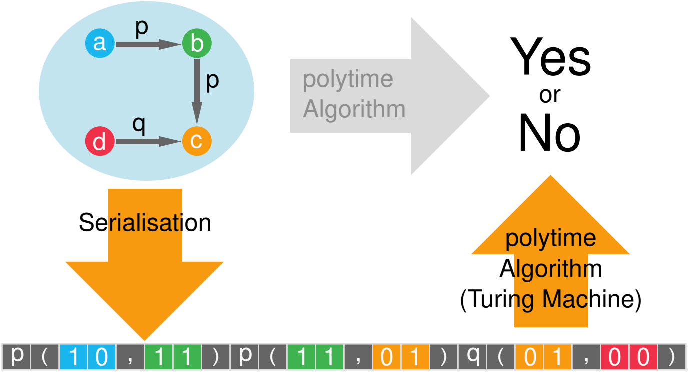
Expressing Queries – Warm-Up

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



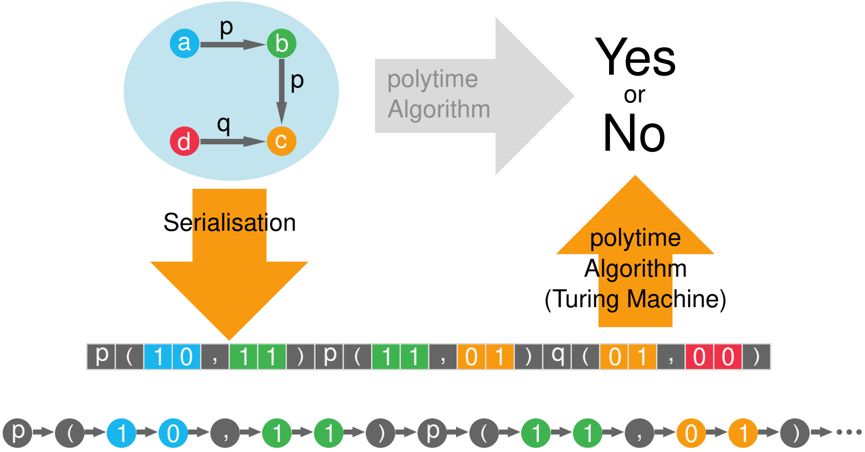
Expressing Queries – Warm-Up

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



Expressing Queries – Warm-Up

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



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!**

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

So can it? **No!**

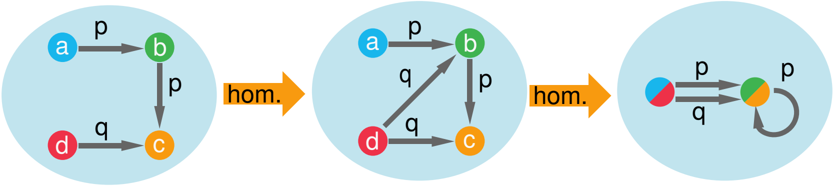
- Datalog has no negation
  $\rightsquigarrow$ all Datalog queries are closed under homomorphism (hom-closed)

Expressing Queries – Warm-Up

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

So can it? **No!**

- Datalog has no negation
  $\rightsquigarrow$ all Datalog queries are closed under homomorphism (hom-closed)

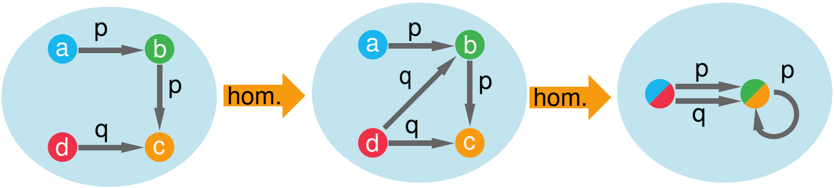


Expressing Queries – Warm-Up

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

So can it? **No!**

- Datalog has no negation
  $\rightsquigarrow$ 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])

Skolem-Chase Terminating Rules just as Expressive as Datalog

Marnette [PODS 2009] showed the following general result:

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

Skolem-Chase Terminating Rules just as Expressive as Datalog

Marnette [PODS 2009] showed the following general result:

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

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

Theorem: For every skolem-chase terminating Σ and BCQ q, there is a set of Datalog rules Σ' and BCQ q' such that $\{\mathcal{D} \mid \mathcal{D}, \Sigma \models q\} = \{\mathcal{D} \mid \mathcal{D}, \Sigma' \models q'\}$.

Toward Higher Expressivity

Logic	**Expressive power**
Datalog	$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\subset$ hom-closed P
Existential Rules	$=$ hom-closed r.e. [R. & Thomazo, IJCAI'15]

Toward Higher Expressivity

Logic	**Expressive power**
Datalog	$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\subset$ 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	**Expressive power**
Datalog	$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	?
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	**Expressive power**
Datalog	$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	?
Core-Chase Terminating Existential Rules	?
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	**Expressive power**
Datalog	$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	?
Core-Chase Terminating Existential Rules	?
Standard-Chase Terminating Disjunctive Exist. Rules	?
Core-Chase Terminating Disjunctive Exist. Rules	?
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	**Expressive power**
Datalog	$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	?
Core-Chase Terminating Existential Rules	?
Standard-Chase Terminating Disjunctive Exist. Rules	?
Core-Chase Terminating Disjunctive Exist. Rules	?
Existential Rules	$=$ hom-closed r.e. [R. & Thomazo, IJCAI'15]

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

Standard Chase? Disjunctive??

The Disjunctive Standard Chase in a Nutshell

Database $\mathcal{D}$ Rule $\rho = \forall \boldsymbol{x}.\big(\beta[\boldsymbol{x}] \to \bigvee_{i=1}^{k} \exists \boldsymbol{y}_i.\eta_i[\boldsymbol{x}_i, \boldsymbol{y}_i]\big)$

Definition: Rule ρ is applicable to $\mathcal{D}$ if:

1. there is a function $h : \boldsymbol{x} \to \mathrm{adom}(\mathcal{D})$ such that $h(\varphi) \subseteq \mathcal{D}$ (a match),
2. for every $i \in \{1, \ldots, k\}$, there is no function $h' : \boldsymbol{x} \cup \boldsymbol{y}_i \to \mathrm{adom}(\mathcal{D})$ with $h'(x) = h(x)$ for all $x \in \boldsymbol{x}$ and $h'(\eta_i) \subseteq \mathcal{D}$.

$\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ is the result of applying ρ to $\mathcal{D}$ under h if $\mathcal{D}_i = \mathcal{D} \cup \hat{h}_i(\eta_i)$ and:

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

- 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?

$\mathcal{D} = \{\text{Bicycle}(c)\}$

$$\begin{aligned}
\text{Bicycle}(x) &\rightarrow \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \\
\text{Wheel}(x) &\rightarrow \exists w.\text{properPartOf}(x, w) \wedge \text{Bicycle}(w) \\
\text{properPartOf}(x, y) &\rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) &\rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) &\rightarrow \text{hasPart}(y, x)
\end{aligned}$$

Will it terminate?

$\mathcal{D} = \{\text{Bicycle}(c)\}$

$$\begin{aligned}
\text{Bicycle}(x) &\rightarrow \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \\
\text{Wheel}(x) &\rightarrow \exists w.\text{properPartOf}(x, w) \wedge \text{Bicycle}(w) \\
\text{properPartOf}(x, y) &\rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) &\rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) &\rightarrow \text{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\rightarrow \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v)\\
\mathsf{Wheel}(x) &\rightarrow \exists w.\mathsf{properPartOf}(x, w) \wedge \mathsf{Bicycle}(w)\\
\mathsf{properPartOf}(x, y) &\rightarrow \mathsf{partOf}(x, y)\\
\mathsf{hasPart}(x, y) &\rightarrow \mathsf{partOf}(y, x)\\
\mathsf{partOf}(x, y) &\rightarrow \mathsf{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

$\mathcal{D}_1 = \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\}$

Will it terminate?

$\mathcal{D} = \{\text{Bicycle}(c)\}$

$$\begin{aligned}
\text{Bicycle}(x) &\rightarrow \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v)\\
\text{Wheel}(x) &\rightarrow \exists w.\text{properPartOf}(x, w) \wedge \text{Bicycle}(w)\\
\text{properPartOf}(x, y) &\rightarrow \text{partOf}(x, y)\\
\text{hasPart}(x, y) &\rightarrow \text{partOf}(y, x)\\
\text{partOf}(x, y) &\rightarrow \text{hasPart}(y, x)
\end{aligned}$$

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?

$$\mathcal{D} = \{\text{Bicycle}(c)\}$$

$$\begin{aligned}
\text{Bicycle}(x) &\to \exists v.\text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) &\to \exists w.\text{properPartOf}(x, w) \land \text{Bicycle}(w) \\
\text{properPartOf}(x, y) &\to \text{partOf}(x, y) \\
\text{hasPart}(x, y) &\to \text{partOf}(y, x) \\
\text{partOf}(x, y) &\to \text{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

$$\begin{aligned}
\mathcal{D}_1 &= \mathcal{D} \cup \{\text{hasPart}(c, n_1), \text{Wheel}(n_1)\} \\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{\text{partOf}(n_1, c)\} \\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{\text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2)\}
\end{aligned}$$

Will it terminate?

$$\mathcal{D} = \{\text{Bicycle}(c)\}$$

$$\begin{aligned}
\text{Bicycle}(x) &\to \exists v.\text{hasPart}(x, v) \land \text{Wheel}(v)\\
\text{Wheel}(x) &\to \exists w.\text{properPartOf}(x, w) \land \text{Bicycle}(w)\\
\text{properPartOf}(x, y) &\to \text{partOf}(x, y)\\
\text{hasPart}(x, y) &\to \text{partOf}(y, x)\\
\text{partOf}(x, y) &\to \text{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

$$\begin{aligned}
\mathcal{D}_1 &= \mathcal{D} \cup \{\text{hasPart}(c, n_1), \text{Wheel}(n_1)\}\\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{\text{partOf}(n_1, c)\}\\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{\text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2)\}\\
\mathcal{D}_4 &= \mathcal{D}_3 \cup \{\text{hasPart}(n_2, n_3), \text{Wheel}(n_3)\}
\end{aligned}$$

Will it terminate?

$$\mathcal{D} = \{\text{Bicycle}(c)\}$$

$$\begin{aligned}
\text{Bicycle}(x) &\to \exists v.\text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) &\to \exists w.\text{properPartOf}(x, w) \land \text{Bicycle}(w) \\
\text{properPartOf}(x, y) &\to \text{partOf}(x, y) \\
\text{hasPart}(x, y) &\to \text{partOf}(y, x) \\
\text{partOf}(x, y) &\to \text{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

$$\begin{aligned}
\mathcal{D}_1 &= \mathcal{D} \cup \{\text{hasPart}(c, n_1), \text{Wheel}(n_1)\} \\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{\text{partOf}(n_1, c)\} \\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{\text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2)\} \\
\mathcal{D}_4 &= \mathcal{D}_3 \cup \{\text{hasPart}(n_2, n_3), \text{Wheel}(n_3)\}
\end{aligned}$$

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

Will it terminate?

$$\mathcal{D} = \{\text{Bicycle}(c)\}$$

$$\begin{aligned}
\text{Bicycle}(x) &\to \exists v.\text{hasPart}(x, v) \land \text{Wheel}(v) \\
\text{Wheel}(x) &\to \exists w.\text{properPartOf}(x, w) \land \text{Bicycle}(w) \\
\text{properPartOf}(x, y) &\to \text{partOf}(x, y) \\
\text{hasPart}(x, y) &\to \text{partOf}(y, x) \\
\text{partOf}(x, y) &\to \text{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

$$\begin{aligned}
\mathcal{D}_1 &= \mathcal{D} \cup \{\text{hasPart}(c, n_1), \text{Wheel}(n_1)\} \\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{\text{partOf}(n_1, c)\} \\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{\text{properPartOf}(n_1, n_2), \text{Bicycle}(n_2)\} \\
\mathcal{D}_4 &= \mathcal{D}_3 \cup \{\text{hasPart}(n_2, n_3), \text{Wheel}(n_3)\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}_5 &= \mathcal{D}_4 \cup \{\text{partOf}(n_1, n_2)\} \\
\mathcal{D}_6 &= \mathcal{D}_5 \cup \{\text{hasPart}(n_2, n_1)\}
\end{aligned}$$

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x, w) \wedge \mathsf{Bicycle}(w) \\
\mathsf{properPartOf}(x, y) &\to \mathsf{partOf}(x, y) \\
\mathsf{hasPart}(x, y) &\to \mathsf{partOf}(y, x) \\
\mathsf{partOf}(x, y) &\to \mathsf{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

$\mathcal{D}_1 = \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\}$

$\mathcal{D}_2 = \mathcal{D}_1 \cup \{\mathsf{partOf}(n_1, c)\}$

$\mathcal{D}_3 = \mathcal{D}_2 \cup \{\mathsf{properPartOf}(n_1, n_2), \mathsf{Bicycle}(n_2)\}$

$\mathcal{D}_4 = \mathcal{D}_3 \cup \{\mathsf{hasPart}(n_2, n_3), \mathsf{Wheel}(n_3)\}$

$\mathcal{D}_5 = \mathcal{D}_4 \cup \{\mathsf{partOf}(n_1, n_2)\}$

$\mathcal{D}_6 = \mathcal{D}_5 \cup \{\mathsf{hasPart}(n_2, n_1)\}$

$\mathcal{D}_7 = \mathcal{D}_6 \cup \{\mathsf{partOf}(n_3, n_2)\}$

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x,v) \wedge \mathsf{Wheel}(v)\\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x,w) \wedge \mathsf{Bicycle}(w)\\
\mathsf{properPartOf}(x,y) &\to \mathsf{partOf}(x,y)\\
\mathsf{hasPart}(x,y) &\to \mathsf{partOf}(y,x)\\
\mathsf{partOf}(x,y) &\to \mathsf{hasPart}(y,x)
\end{aligned}$$

Applying the standard chase may yield:

$$\begin{aligned}
\mathcal{D}_1 &= \mathcal{D} \cup \{\mathsf{hasPart}(c,n_1), \mathsf{Wheel}(n_1)\} & \mathcal{D}_5 &= \mathcal{D}_4 \cup \{\mathsf{partOf}(n_1,n_2)\}\\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{\mathsf{partOf}(n_1,c)\} & \mathcal{D}_6 &= \mathcal{D}_5 \cup \{\mathsf{hasPart}(n_2,n_1)\}\\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{\mathsf{properPartOf}(n_1,n_2), \mathsf{Bicycle}(n_2)\} & \mathcal{D}_7 &= \mathcal{D}_6 \cup \{\mathsf{partOf}(n_3,n_2)\}\\
\mathcal{D}_4 &= \mathcal{D}_3 \cup \{\mathsf{hasPart}(n_2,n_3), \mathsf{Wheel}(n_3)\} & \mathcal{D}_8 &= \mathcal{D}_7 \cup \{\mathsf{partOf}(n_3,n_4), \mathsf{Bicycle}(n_4)\}
\end{aligned}$$

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x, w) \wedge \mathsf{Bicycle}(w) \\
\mathsf{properPartOf}(x, y) &\to \mathsf{partOf}(x, y) \\
\mathsf{hasPart}(x, y) &\to \mathsf{partOf}(y, x) \\
\mathsf{partOf}(x, y) &\to \mathsf{hasPart}(y, x)
\end{aligned}$$

Applying the standard chase may yield:

$$\begin{aligned}
\mathcal{D}_1 &= \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\} \\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{\mathsf{partOf}(n_1, c)\} \\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{\mathsf{properPartOf}(n_1, n_2), \mathsf{Bicycle}(n_2)\} \\
\mathcal{D}_4 &= \mathcal{D}_3 \cup \{\mathsf{hasPart}(n_2, n_3), \mathsf{Wheel}(n_3)\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}_5 &= \mathcal{D}_4 \cup \{\mathsf{partOf}(n_1, n_2)\} \\
\mathcal{D}_6 &= \mathcal{D}_5 \cup \{\mathsf{hasPart}(n_2, n_1)\} \\
\mathcal{D}_7 &= \mathcal{D}_6 \cup \{\mathsf{partOf}(n_3, n_2)\} \\
\mathcal{D}_8 &= \mathcal{D}_7 \cup \{\mathsf{partOf}(n_3, n_4), \mathsf{Bicycle}(n_4)\}
\end{aligned}$$

The chase can continue forever . . .

Will it terminate?

$\mathcal{D} = \{\text{Bicycle}(c)\}$

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

$$\text{Wheel}(x) \rightarrow \exists w.\text{properPartOf}(x, w) \wedge \text{Bicycle}(w)$$

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

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

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

Will it terminate?

$\mathcal{D} = \{\text{Bicycle}(c)\}$

$$\begin{aligned}
\text{Bicycle}(x) &\rightarrow \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \\
\text{Wheel}(x) &\rightarrow \exists w.\text{properPartOf}(x, w) \wedge \text{Bicycle}(w) \\
\text{properPartOf}(x, y) &\rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) &\rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) &\rightarrow \text{hasPart}(y, x)
\end{aligned}$$

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

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x, w) \wedge \mathsf{Bicycle}(w) \\
\mathsf{properPartOf}(x, y) &\to \mathsf{partOf}(x, y) \\
\mathsf{hasPart}(x, y) &\to \mathsf{partOf}(y, x) \\
\mathsf{partOf}(x, y) &\to \mathsf{hasPart}(y, x)
\end{aligned}$$

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

$\mathcal{D}_1 = \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\}$

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v)\\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x, w) \wedge \mathsf{Bicycle}(w)\\
\mathsf{properPartOf}(x, y) &\to \mathsf{partOf}(x, y)\\
\mathsf{hasPart}(x, y) &\to \mathsf{partOf}(y, x)\\
\mathsf{partOf}(x, y) &\to \mathsf{hasPart}(y, x)
\end{aligned}$$

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

$\mathcal{D}_1 = \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\}$

$\mathcal{D}_2 = \mathcal{D}_1 \cup \{\mathsf{partOf}(n_1, c)\}$

Will it terminate?

$$\mathcal{D} = \{\text{Bicycle}(c)\}$$

$$\begin{aligned}
\text{Bicycle}(x) &\rightarrow \exists v.\text{hasPart}(x, v) \wedge \text{Wheel}(v) \\
\text{Wheel}(x) &\rightarrow \exists w.\text{properPartOf}(x, w) \wedge \text{Bicycle}(w) \\
\text{properPartOf}(x, y) &\rightarrow \text{partOf}(x, y) \\
\text{hasPart}(x, y) &\rightarrow \text{partOf}(y, x) \\
\text{partOf}(x, y) &\rightarrow \text{hasPart}(y, x)
\end{aligned}$$

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

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

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

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

Will it terminate?

$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x, w) \wedge \mathsf{Bicycle}(w) \\
\mathsf{properPartOf}(x, y) &\to \mathsf{partOf}(x, y) \\
\mathsf{hasPart}(x, y) &\to \mathsf{partOf}(y, x) \\
\mathsf{partOf}(x, y) &\to \mathsf{hasPart}(y, x)
\end{aligned}$$

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

$\mathcal{D}_1 = \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\}$
$\mathcal{D}_2 = \mathcal{D}_1 \cup \{\mathsf{partOf}(n_1, c)\}$
$\mathcal{D}_3 = \mathcal{D}_2 \cup \{\mathsf{properPartOf}(n_1, n_2), \mathsf{Bicycle}(n_2)\}$
$\mathcal{D}_4 = \mathcal{D}_3 \cup \{\mathsf{partOf}(n_1, n_2)\}$

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x, v) \wedge \mathsf{Wheel}(v) \\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x, w) \wedge \mathsf{Bicycle}(w) \\
\mathsf{properPartOf}(x, y) &\to \mathsf{partOf}(x, y) \\
\mathsf{hasPart}(x, y) &\to \mathsf{partOf}(y, x) \\
\mathsf{partOf}(x, y) &\to \mathsf{hasPart}(y, x)
\end{aligned}$$

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

$\mathcal{D}_1 = \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\}$

$\mathcal{D}_2 = \mathcal{D}_1 \cup \{\mathsf{partOf}(n_1, c)\}$

$\mathcal{D}_3 = \mathcal{D}_2 \cup \{\mathsf{properPartOf}(n_1, n_2), \mathsf{Bicycle}(n_2)\}$

$\mathcal{D}_4 = \mathcal{D}_3 \cup \{\mathsf{partOf}(n_1, n_2)\}$

$\mathcal{D}_5 = \mathcal{D}_4 \cup \{\mathsf{hasPart}(n_2, n_1)\}$

Will it terminate?

$$\mathcal{D} = \{\mathsf{Bicycle}(c)\}$$

$$\begin{aligned}
\mathsf{Bicycle}(x) &\to \exists v.\mathsf{hasPart}(x, v) \land \mathsf{Wheel}(v)\\
\mathsf{Wheel}(x) &\to \exists w.\mathsf{properPartOf}(x, w) \land \mathsf{Bicycle}(w)\\
\mathsf{properPartOf}(x, y) &\to \mathsf{partOf}(x, y)\\
\mathsf{hasPart}(x, y) &\to \mathsf{partOf}(y, x)\\
\mathsf{partOf}(x, y) &\to \mathsf{hasPart}(y, x)
\end{aligned}$$

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

$$\begin{aligned}
\mathcal{D}_1 &= \mathcal{D} \cup \{\mathsf{hasPart}(c, n_1), \mathsf{Wheel}(n_1)\}\\
\mathcal{D}_2 &= \mathcal{D}_1 \cup \{\mathsf{partOf}(n_1, c)\}\\
\mathcal{D}_3 &= \mathcal{D}_2 \cup \{\mathsf{properPartOf}(n_1, n_2), \mathsf{Bicycle}(n_2)\}\\
\mathcal{D}_4 &= \mathcal{D}_3 \cup \{\mathsf{partOf}(n_1, n_2)\}\\
\mathcal{D}_5 &= \mathcal{D}_4 \cup \{\mathsf{hasPart}(n_2, n_1)\}
\end{aligned}$$

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:

Termination on ...	instance $\mathcal{D}$	all instances
Skolem chase	$\mathsf{CT}^{\mathsf{sk}}_{\mathcal{D}}$	$\mathsf{CT}^{\mathsf{sk}}_{\forall}$
Standard chase (all/some strategies)	$\mathsf{CT}^{\mathsf{std}}_{\mathcal{D}\forall}$ / $\mathsf{CT}^{\mathsf{std}}_{\mathcal{D}\exists}$	$\mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ / $\mathsf{CT}^{\mathsf{std}}_{\forall\exists}$
Datalog-first chase (all/some strategies)	$\mathsf{CT}^{\mathsf{dlf}}_{\mathcal{D}\forall}$ / $\mathsf{CT}^{\mathsf{dlf}}_{\mathcal{D}\exists}$	$\mathsf{CT}^{\mathsf{dlf}}_{\forall\forall}$ / $\mathsf{CT}^{\mathsf{dlf}}_{\forall\exists}$
Core chase	$\mathsf{CT}^{\mathsf{cor}}_{\mathcal{D}}$	$\mathsf{CT}^{\mathsf{cor}}_{\forall}$

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

$$\mathsf{CT}^{\mathsf{sk}}_{\forall} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall} \subset \mathsf{CT}^{\mathsf{dlf}}_{\forall\forall} \subset \mathsf{CT}^{\mathsf{dlf}}_{\forall\exists} = \mathsf{CT}^{\mathsf{std}}_{\forall\exists} \subset \mathsf{CT}^{\mathsf{cor}}_{\forall}$$

Main Result

Logic		Expressive power
Datalog		$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$CT^{sk}_{\forall}$	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	$CT^{std}_{\forall\forall}$	?
Core-Chase Terminating Existential Rules	$CT^{cor}_{\forall}$	?
Standard-Chase Terminating Disjunctive Exist. Rules	$CT^{std,\vee}_{\forall\forall}$	?
Core-Chase Terminating Disjunctive Exist. Rules	$CT^{cor,\vee}_{\forall}$	?

Main Result

Logic		Expressive power
Datalog		$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\mathrm{CT}^{\mathrm{sk}}_{\forall}$	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	$\mathrm{CT}^{\mathrm{std}}_{\forall\forall}$	$=$ hom-closed decidable
Core-Chase Terminating Existential Rules	$\mathrm{CT}^{\mathrm{cor}}_{\forall}$	?
Standard-Chase Terminating Disjunctive Exist. Rules	$\mathrm{CT}^{\mathrm{std},\vee}_{\forall\forall}$	?
Core-Chase Terminating Disjunctive Exist. Rules	$\mathrm{CT}^{\mathrm{cor},\vee}_{\forall}$	?

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

Logic		**Expressive power**
Datalog		$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\mathrm{CT}^{\mathrm{sk}}_{\forall}$	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	$\mathrm{CT}^{\mathrm{std}}_{\forall\forall}$	$=$ hom-closed decidable
Core-Chase Terminating Existential Rules	$\mathrm{CT}^{\mathrm{cor}}_{\forall}$	$=$ hom-closed decidable
Standard-Chase Terminating Disjunctive Exist. Rules	$\mathrm{CT}^{\mathrm{std},\vee}_{\forall\forall}$	$=$ hom-closed decidable
Core-Chase Terminating Disjunctive Exist. Rules	$\mathrm{CT}^{\mathrm{cor},\vee}_{\forall}$	$=$ hom-closed decidable

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.

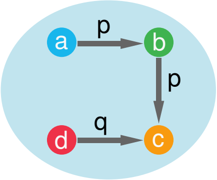
Proof Plan

(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

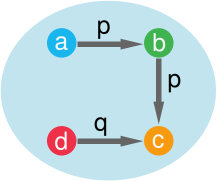
Order!



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

$$\mathsf{elem}(x) \wedge \mathsf{elem}(y) \rightarrow (x \approx y) \vee (x \not\approx y)$$
$$(x \not\approx y) \rightarrow (x < y) \vee (y < x)$$
$$\ldots$$

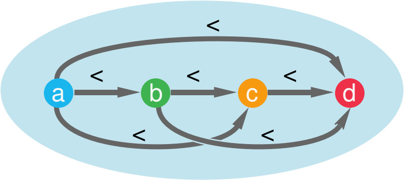
Order!



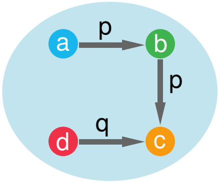
We can use disjunctions to guess how database elements are ordered:

$$\mathsf{elem}(x) \wedge \mathsf{elem}(y) \rightarrow (x \approx y) \vee (x \not\approx y)$$
$$(x \not\approx y) \rightarrow (x < y) \vee (y < x)$$
$$\ldots$$

Many possible models arise:



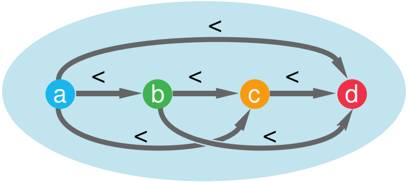
Order!

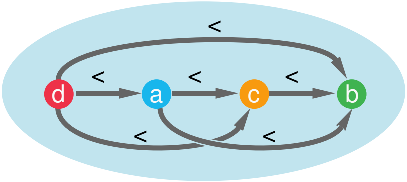


We can use disjunctions to guess how database elements are ordered:

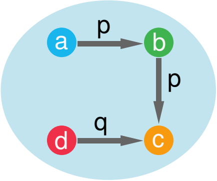
$$\mathsf{elem}(x) \land \mathsf{elem}(y) \to (x \approx y) \lor (x \not\approx y)$$
$$(x \not\approx y) \to (x < y) \lor (y < x)$$
$$\ldots$$

Many possible models arise:





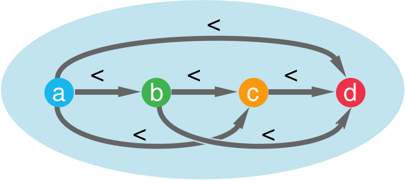
Order!

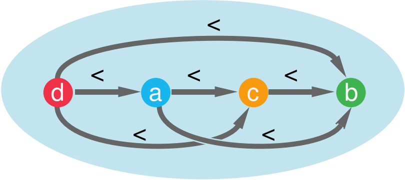


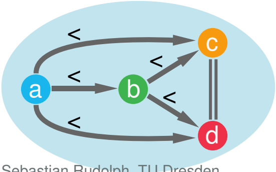
We can use disjunctions to guess how database elements are ordered:

$$\text{elem}(x) \wedge \text{elem}(y) \rightarrow (x \approx y) \vee (x \not\approx y)$$
$$(x \not\approx y) \rightarrow (x < y) \vee (y < x)$$
$$\ldots$$

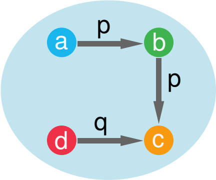
Many possible models arise:







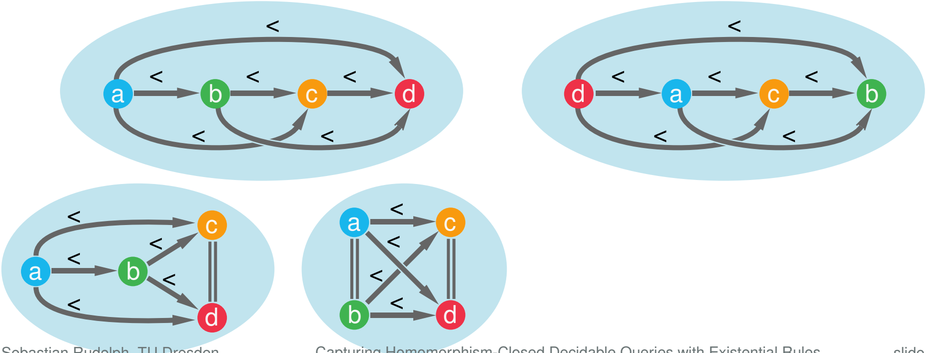
Order!



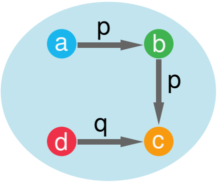
We can use disjunctions to guess how database elements are ordered:

$$\mathsf{elem}(x) \wedge \mathsf{elem}(y) \rightarrow (x \approx y) \vee (x \not\approx y)$$
$$(x \not\approx y) \rightarrow (x < y) \vee (y < x)$$
$$\ldots$$

Many possible models arise:



Order!



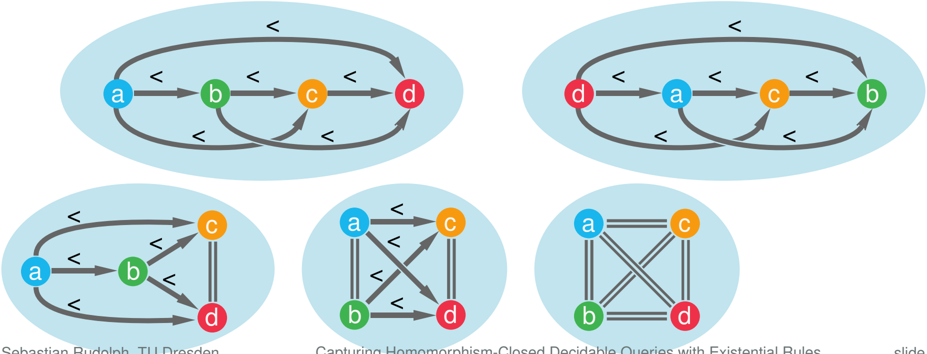
We can use disjunctions to guess how database elements are ordered:

$$\text{elem}(x) \wedge \text{elem}(y) \rightarrow (x \approx y) \vee (x \not\approx y)$$

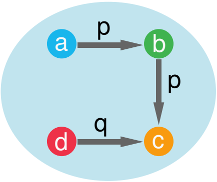
$$(x \not\approx y) \rightarrow (x < y) \vee (y < x)$$

$$\ldots$$

Many possible models arise:



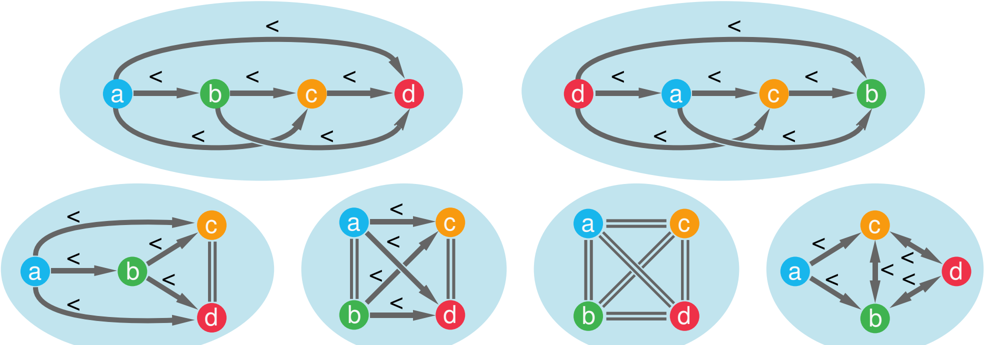
Order!



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

$$\mathsf{elem}(x) \wedge \mathsf{elem}(y) \rightarrow (x \approx y) \vee (x \not\approx y)$$
$$(x \not\approx y) \rightarrow (x < y) \vee (y < x)$$
$$\ldots$$

Many possible models arise:



Completion

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

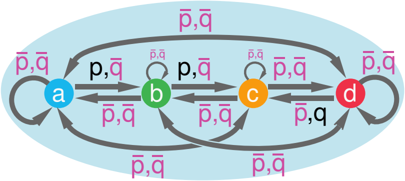
$$\mathsf{elem}(x_1) \wedge \mathsf{elem}(x_2) \rightarrow p(x_1, x_2) \vee \bar{p}(x_1, x_2)$$
$$\mathsf{elem}(x_1) \wedge \mathsf{elem}(x_2) \rightarrow q(x_1, x_2) \vee \bar{q}(x_1, x_2)$$
$$\ldots$$

Completion

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

$$\text{elem}(x_1) \wedge \text{elem}(x_2) \rightarrow p(x_1, x_2) \vee \bar{p}(x_1, x_2)$$
$$\text{elem}(x_1) \wedge \text{elem}(x_2) \rightarrow q(x_1, x_2) \vee \bar{q}(x_1, x_2)$$
$$\ldots$$

For each possible order, we obtain many possible completions:

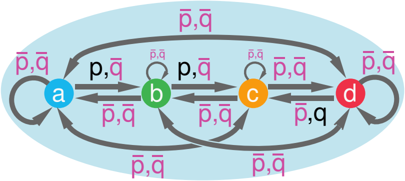


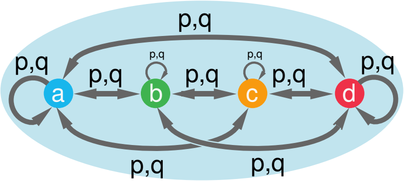
Completion

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

$$\mathsf{elem}(x_1) \wedge \mathsf{elem}(x_2) \to p(x_1, x_2) \vee \bar{p}(x_1, x_2)$$
$$\mathsf{elem}(x_1) \wedge \mathsf{elem}(x_2) \to q(x_1, x_2) \vee \bar{q}(x_1, x_2)$$
$$\ldots$$

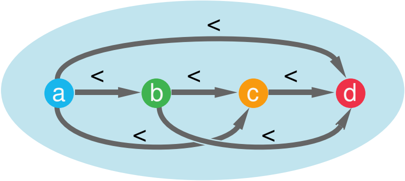
For each possible order, we obtain many possible completions:





. . .

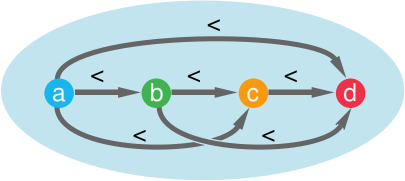
Order! ORDER!



This is not a suitable successor relation.

(since < is transitive)

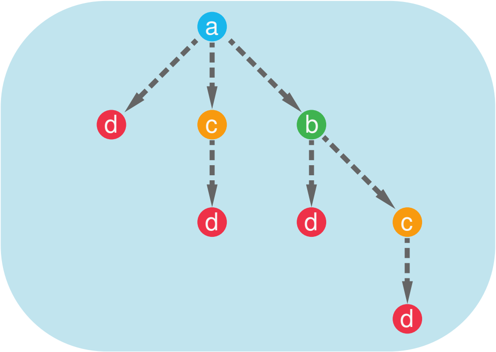
Order! ORDER!



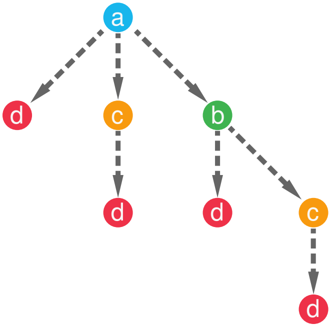
This is not a suitable successor relation.

(since < is transitive)

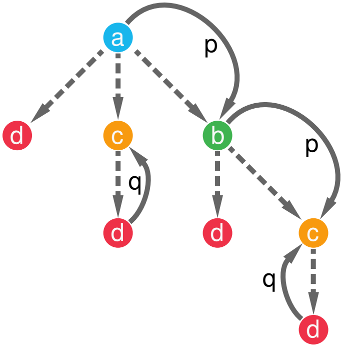
Solution: Construct a tree by tracing directed paths:

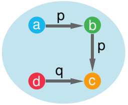


Tree to tape(s)

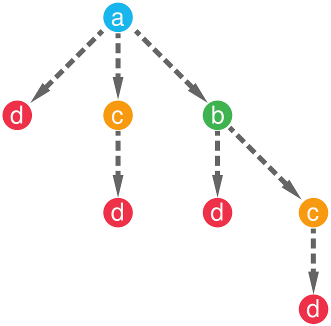


Tree to tape(s)

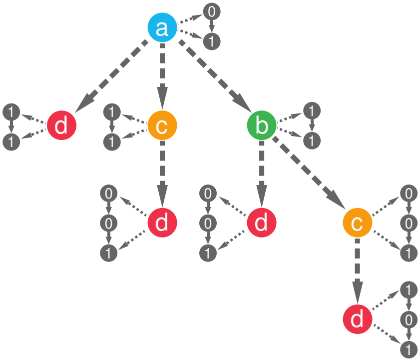




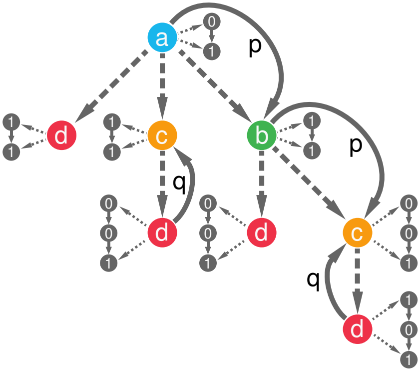
Tree to tape(s)



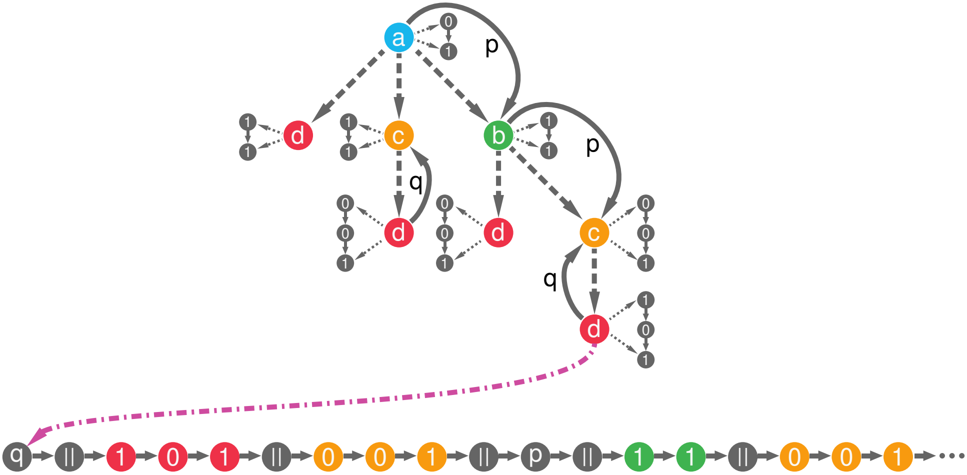
Tree to tape(s)



Tree to tape(s)

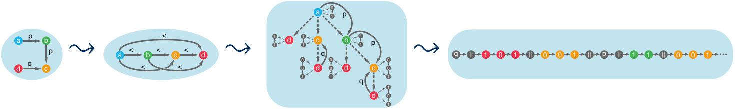


Tree to tape(s)



Proof Plan

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

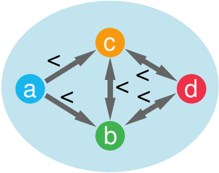


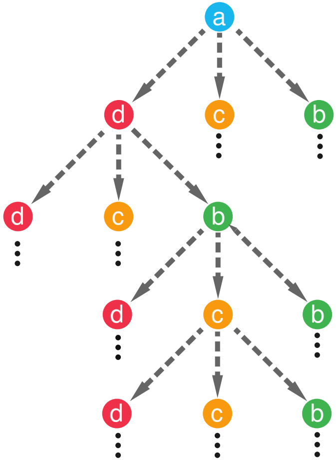
(2) Standard-chase terminating disjunctive existential rules can express all decidable hom-closed queries

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

Enforcing Termination

Some models are infinite:

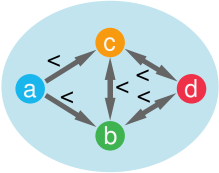


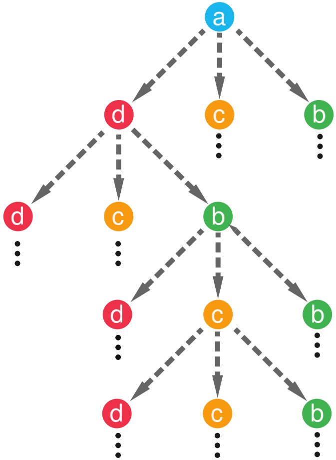


But: We can write a Datalog query to detect when this occurs.

Enforcing Termination

Some models are infinite:



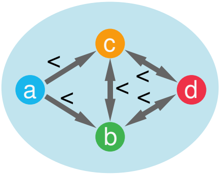


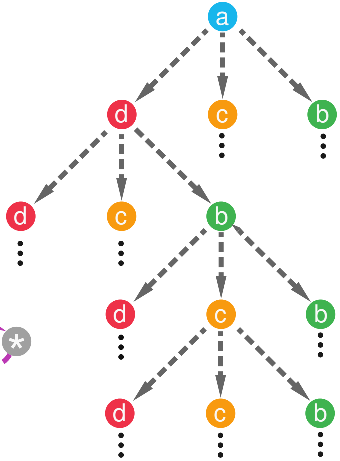
But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:

Enforcing Termination

Some models are infinite:





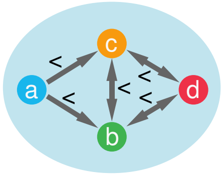
But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:

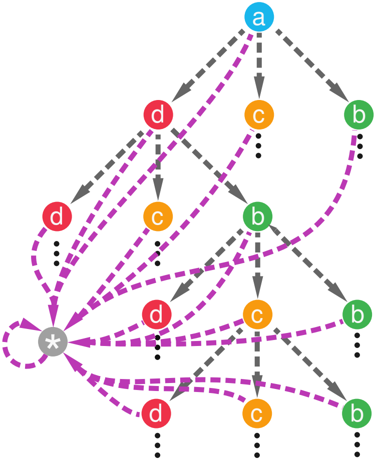
1. Prepare an “inactive” structure for which any further model expansion is redundant

Enforcing Termination

Some models are infinite:



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

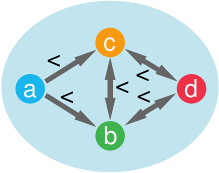


Emergency Brake construction:

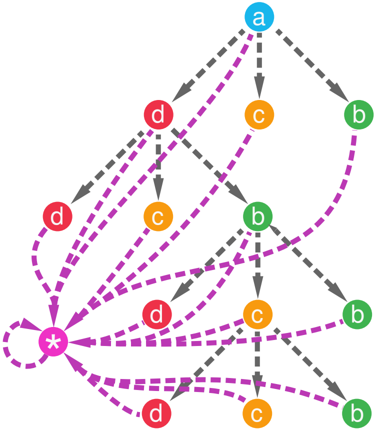
1. Prepare an “inactive” structure for which any further model expansion is redundant
2. Connect all elements to this structure

Enforcing Termination

Some models are infinite:



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

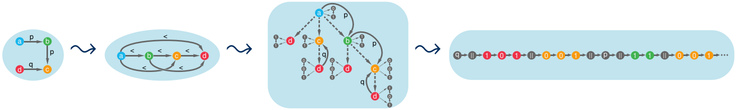


Emergency Brake construction:

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



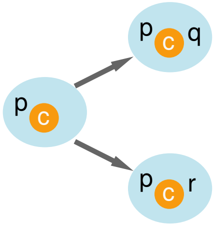
(2) Standard-chase terminating disjunctive existential rules can express all decidable hom-closed queries

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

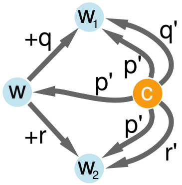
Simulating Disjunctive Datalog in Existential Rules

Idea: Represent possible worlds with existentially introduced elements

$$p(x) \rightarrow q(x) \vee r(x)$$



$\rightsquigarrow$



Challenge: Creation of fresh "worlds" must terminate
$\rightsquigarrow$ adapt chase-terminating set-modelling technique of [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 [Grahne & Onet, Fund. Inf. 2018].

Open questions:

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