Capturing Homomorphism-Closed Decidable Queries with Existential Rules

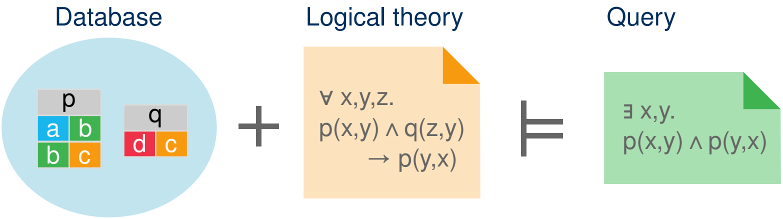
As presented at the 18th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'21)

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

KR for Data Access

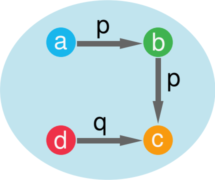
Ontology-Based Query Answering: query results = logical entailments over databases



KR for Data Access

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

Database



+

Logical theory

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

$\models$

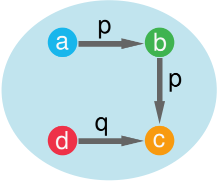
Query

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

KR for Data Access

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

Database + Logical theory ⊨ Query



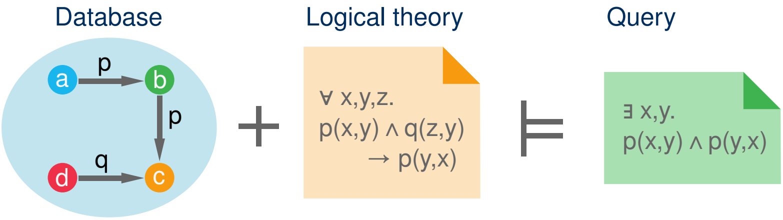
Logical theory: $\forall x,y,z.\ p(x,y) \wedge q(z,y) \rightarrow p(y,x)$

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

Logic	Data Complexity
DL-Lite	AC_0
Datalog	P
Disjunctive Datalog	coNP
Existential Rules	r.e.
Disjunctive Exist. Rules	r.e.

KR for Data Access

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



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

Decidable Reasoning for Existential Rules

The Chase: Iteratively apply rules bottom-up

- If chase terminates: decide queries on resulting structure
- Termination depends on the details of the chase definition:
  - Skolem chase: avoid duplicates of skolem terms
  - Standard chase: avoid semantic redundancy locally
  - Core chase: avoid semantic redundancy globally
- Chase termination is undecidable in all cases

Decidable Reasoning for Existential Rules

The Chase: Iteratively apply rules bottom-up

- If chase terminates: decide queries on resulting structure
- Termination depends on the details of the chase definition:
  - Skolem chase: avoid duplicates of skolem terms
  - Standard chase: avoid semantic redundancy locally
  - Core chase: avoid semantic redundancy globally
- Chase termination is undecidable in all cases

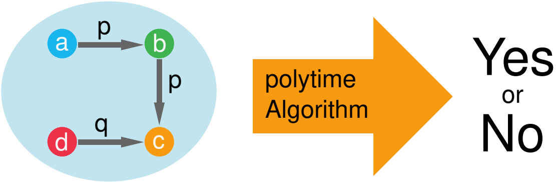
Logic	Data Complexity
Datalog	P
Skolem-Chase Terminating Existential Rules	P [Marnette, PODS 2009]
Standard-Chase Terminating Existential Rules	?
Core-Chase Terminating Existential Rules	?

Expressing Queries

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

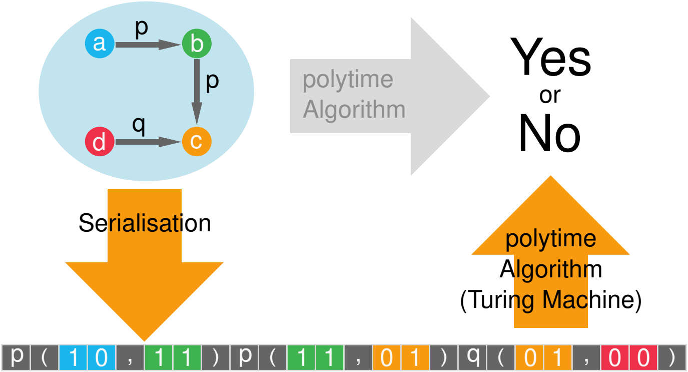
Expressing Queries

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



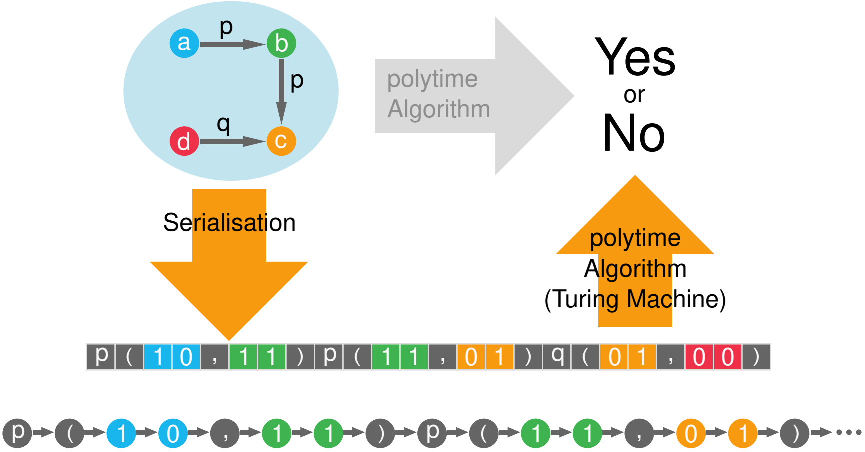
Expressing Queries

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



Expressing Queries

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



Expressing Queries

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

So can it?

Expressing Queries

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

So can it?

No!

Expressing Queries

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

So can it?

No!

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

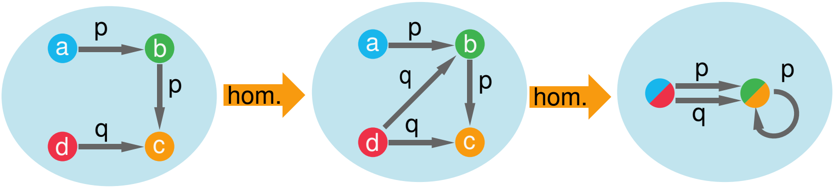
Expressing Queries

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

So can it?

No!

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



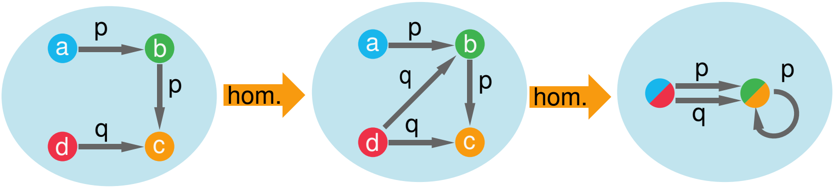
Expressing Queries

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

So can it?

No!

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



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

Main Result

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	?

Main Result

Logic	**Expressive power**
Datalog	$\subset$ hom-closed P
Skolem-Chase Terminating Existential Rules	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	$=$ hom-closed decidable
Core-Chase Terminating Existential Rules	?
Standard-Chase Terminating Disjunctive Exist. Rules	?
Core-Chase Terminating Disjunctive Exist. Rules	?

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	$\subset$ hom-closed P
Standard-Chase Terminating Existential Rules	= hom-closed decidable
Core-Chase Terminating Existential Rules	= hom-closed decidable
Standard-Chase Terminating Disjunctive Exist. Rules	= hom-closed decidable
Core-Chase Terminating Disjunctive Exist. Rules	= 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.

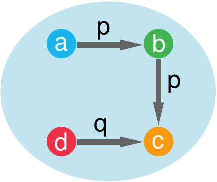
Proof Plan

(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!



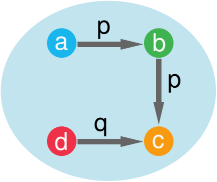
We can use disjunctions to guess how database elements are ordered:

$$\mathsf{elem}(x) \wedge \mathsf{elem}(y) \to (x \approx y) \vee (x \not\approx y)$$

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

$$\ldots$$

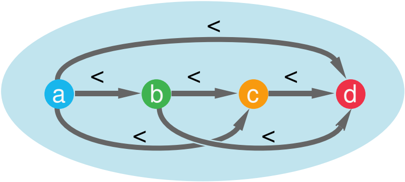
Order!



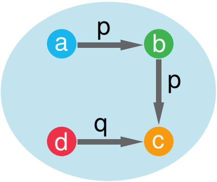
We can use disjunctions to guess how database elements are ordered:

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

Many possible models arise:



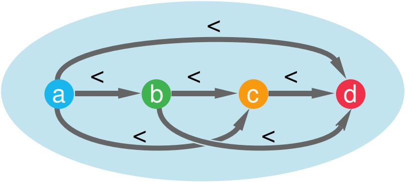
Order!

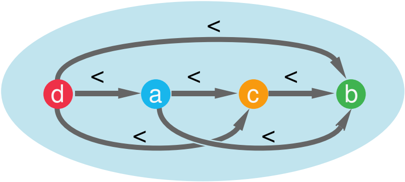


We can use disjunctions to guess how database elements are ordered:

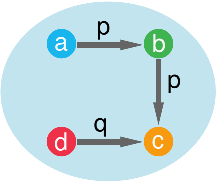
$$\mathsf{elem}(x) \wedge \mathsf{elem}(y) \to (x \approx y) \vee (x \not\approx y)$$
$$(x \not\approx y) \to (x < y) \vee (y < x)$$
$$\ldots$$

Many possible models arise:





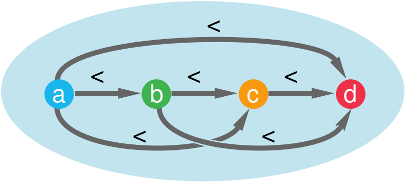
Order!

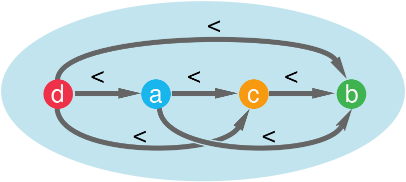


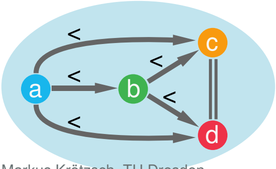
We can use disjunctions to guess how database elements are ordered:

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

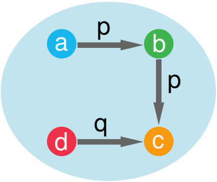
Many possible models arise:







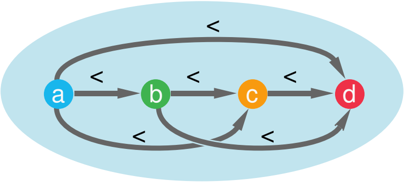
Order!

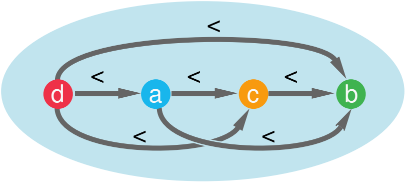


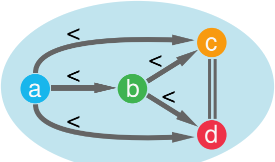
We can use disjunctions to guess how database elements are ordered:

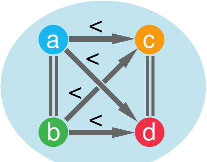
$$\mathsf{elem}(x) \wedge \mathsf{elem}(y) \to (x \approx y) \vee (x \not\approx y)$$
$$(x \not\approx y) \to (x < y) \vee (y < x)$$
$$\ldots$$

Many possible models arise:

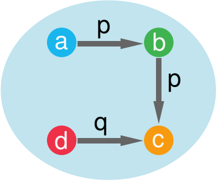








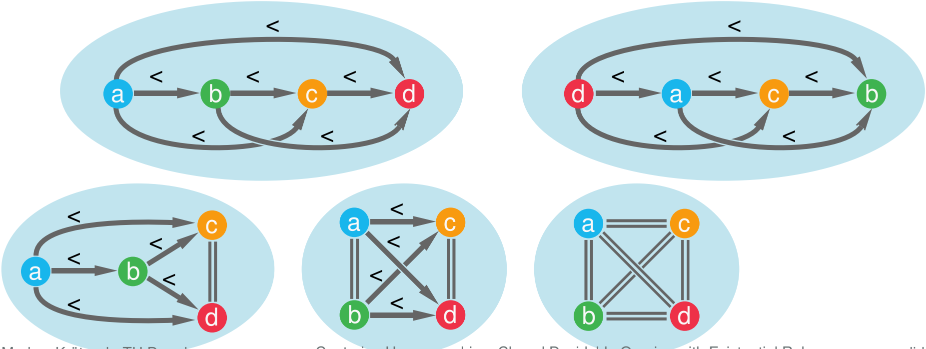
Order!



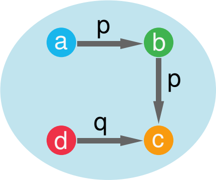
We can use disjunctions to guess how database elements are ordered:

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

Many possible models arise:



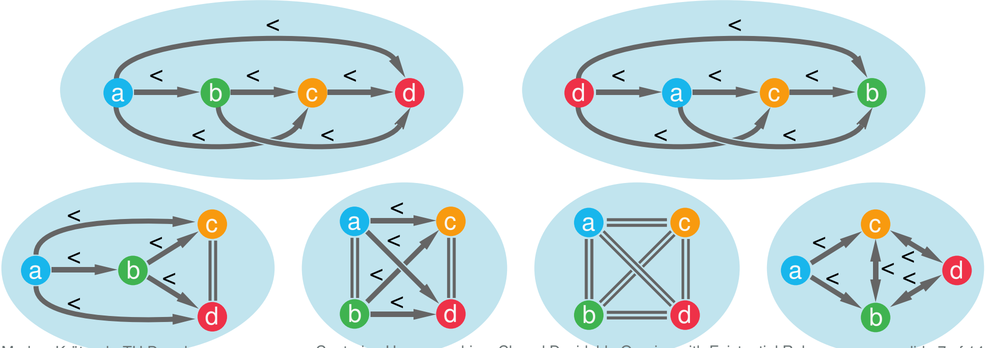
Order!



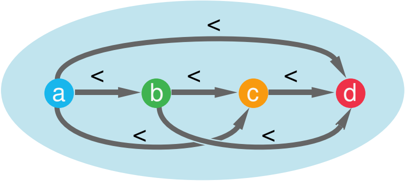
We can use disjunctions to guess how database elements are ordered:

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

Many possible models arise:



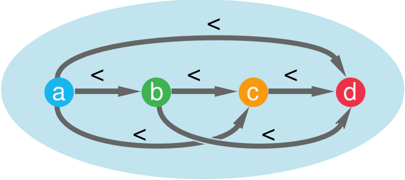
Order! ORDER!



This is not a suitable successor relation.

(since < is transitive)

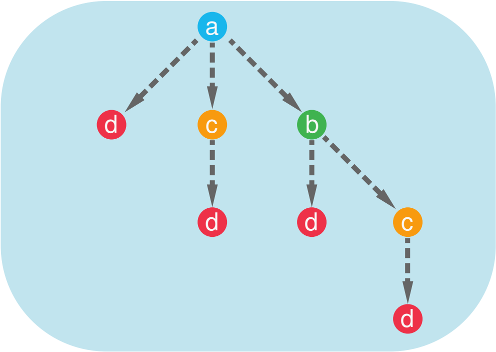
Order! ORDER!



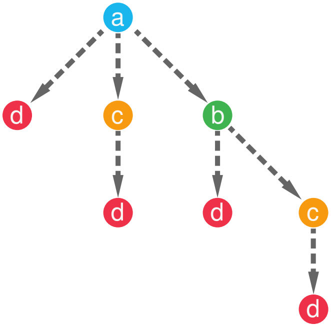
This is not a suitable successor relation.

(since < is transitive)

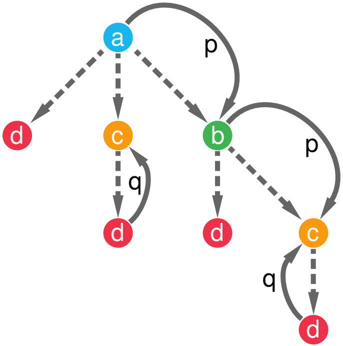
Solution: Construct a tree by tracing directed paths:

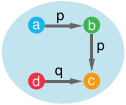


Tree to tape(s)

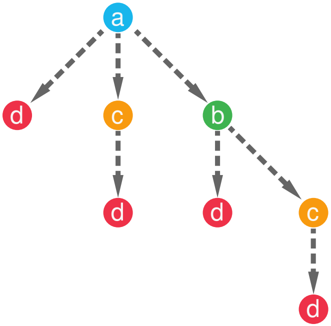


Tree to tape(s)

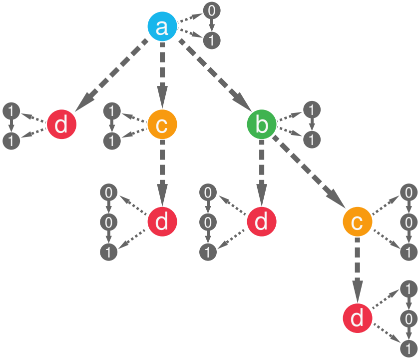




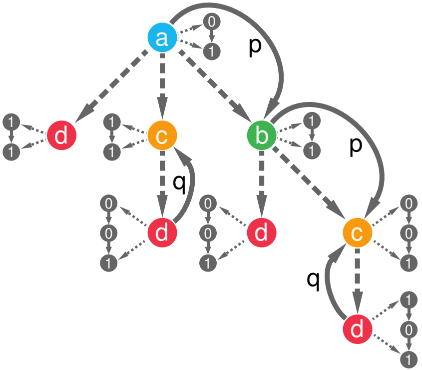
Tree to tape(s)



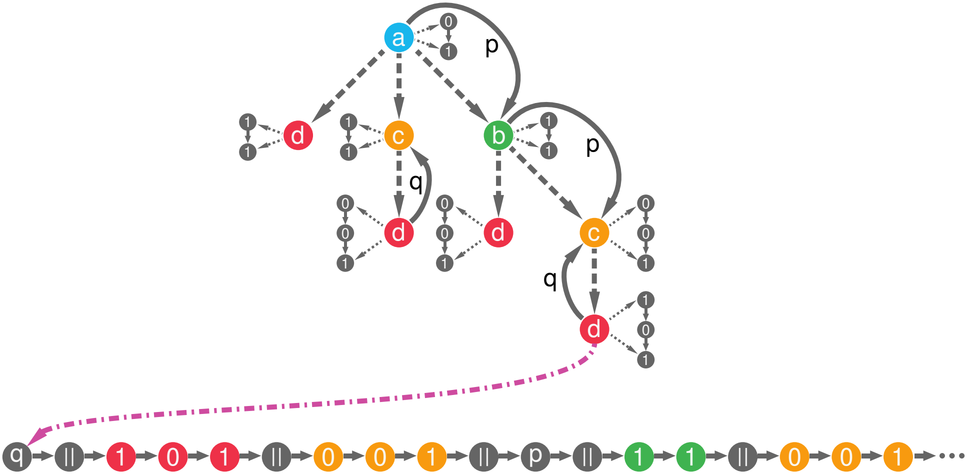
Tree to tape(s)



Tree to tape(s)

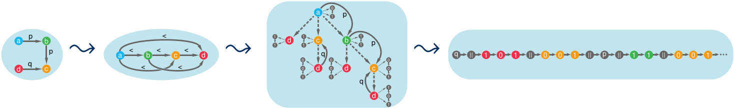


Tree to tape(s)



Proof Plan

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

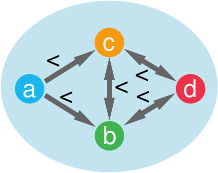


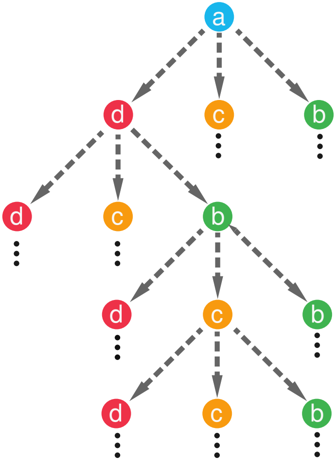
(2) Standard-chase terminating disjunctive existential rules can express all decidable hom-closed queries

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

Enforcing Termination

Some models are infinite:

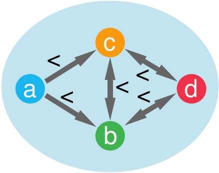


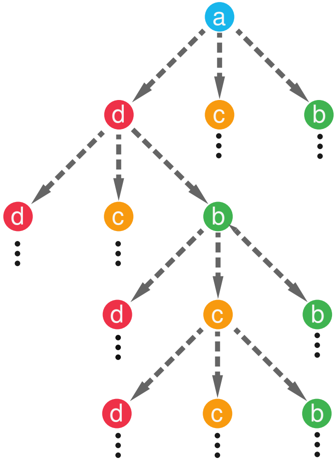


But: We can write a Datalog query to detect when this occurs.

Enforcing Termination

Some models are infinite:



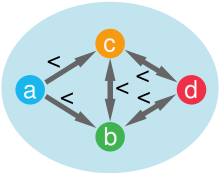


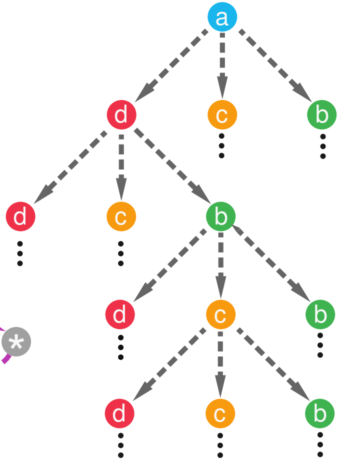
But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:

Enforcing Termination

Some models are infinite:





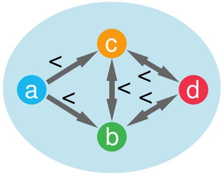
But: We can write a Datalog query to detect when this occurs.

Emergency Brake construction:

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

Enforcing Termination

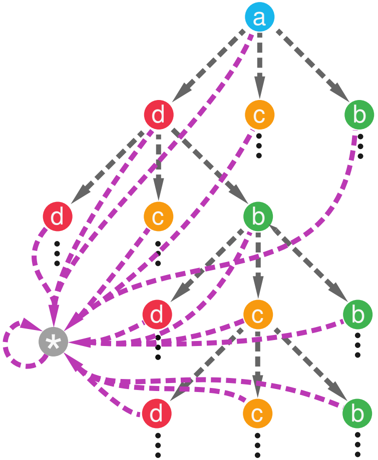
Some models are infinite:



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

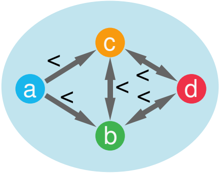
Emergency Brake construction:

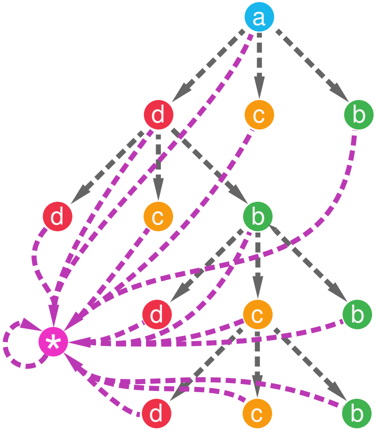
1. Prepare an “inactive” structure for which any further model expansion is redundant
2. Connect all elements to this structure



Enforcing Termination

Some models are infinite:





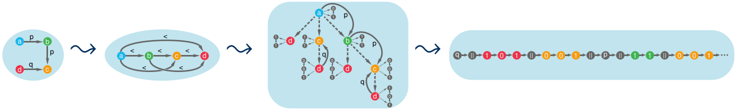
But: We can write a Datalog query to detect when this occurs.

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



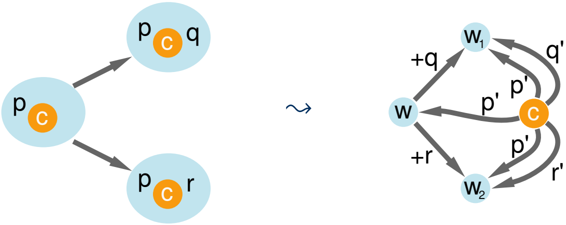
(2) Standard-chase terminating disjunctive existential rules can express all decidable hom-closed queries

(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

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



Challenge: Creation of fresh "worlds" must terminate
⇝ adapt chase-terminating set-modelling technique of [Krötzsch et al. 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?