Parallelisable Existential Rules: a Story of Pieces

Maxime Buron, Marie-Laure Mugnier, Michaël Thomazo

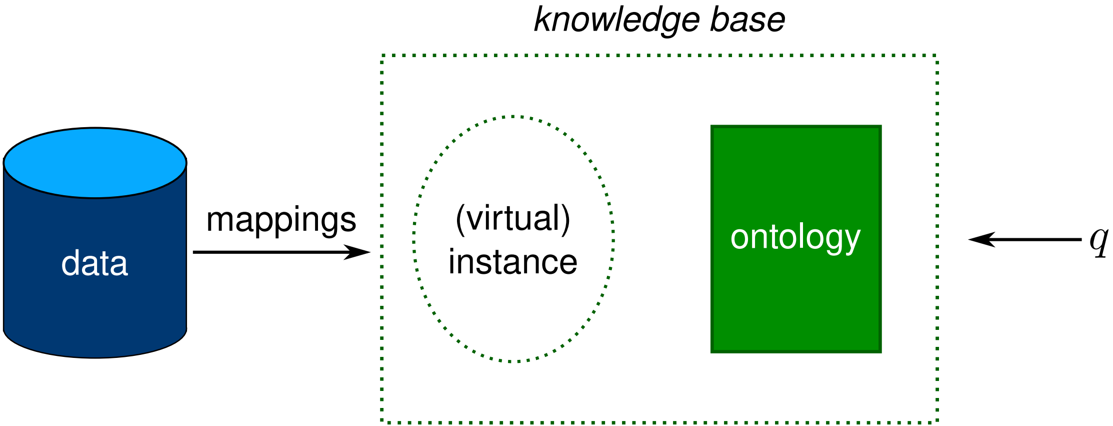
BDA 2022,
25/10/2022

Table of Contents

Motivation: how to answer a query in OBDA using only mappings ?

Context

Ontology-Based Data Access



Mappings as existential rules

Existential rules

$$\forall \vec{x}\ \forall \vec{y}\ (\ \text{Body}[\vec{x}, \vec{y}] \rightarrow \exists \vec{z}\ \text{Head}[\vec{x}, \vec{z}]\ )$$

Mappings (aka source-to-target Tuple Generating Dependencies)

$$\forall \vec{x}\ (\ \exists \vec{y}\ \text{Body}[\vec{x}, \vec{y}] \rightarrow \exists \vec{z}\ \text{Head}[\vec{x}, \vec{z}]\ )$$

- Body is a conjunctive query on the data with answer variables $\vec{x}$
- Head is a conjunctive query on the vocabulary of the ontology with answer variables $\vec{x}$

In the following:

- Rules and mappings have no constants

Chasing with existential rules

Example

$$\mathcal{M}: \quad \begin{array}{l} M_1 = s_1(x,y) \rightarrow t_1(x,y) \\ M_2 = s_2(x,y) \rightarrow t_2(x) \end{array} \quad \Bigg| \quad \mathcal{R}: \quad \begin{array}{l} R_1 = t_2(x) \rightarrow \exists z\ t_3(x,z) \\ R_2 = t_1(x,y) \wedge t_3(x,z) \rightarrow t_4(y) \end{array}$$

Chasing steps

- $\text{chase}_0(D, \mathcal{M} \cup \mathcal{R}) = D = \{s_1(a,b), s_2(a,c)\}$

Chasing with existential rules

Example

$\mathcal{M}$: $M_1 = s_1(x, y) \rightarrow t_1(x, y)$
$M_2 = s_2(x, y) \rightarrow t_2(x)$

$\mathcal{R}$: $R_1 = t_2(x) \rightarrow \exists z\ t_3(x, z)$
$R_2 = t_1(x, y) \wedge t_3(x, z) \rightarrow t_4(y)$

Chasing steps

- $\text{chase}_0(D, \mathcal{M} \cup \mathcal{R}) = D = \{s_1(a, b), s_2(a, c)\}$
- $\text{chase}_1(D, \mathcal{M} \cup \mathcal{R}) = \text{chase}_0(D, \mathcal{M} \cup \mathcal{R}) \cup \{t_1(a, b), t_2(a)\}$

Chasing with existential rules

Example

$$\mathcal{M}: \quad \begin{aligned} M_1 &= s_1(x,y) \rightarrow t_1(x,y) \\ M_2 &= s_2(x,y) \rightarrow t_2(x) \end{aligned} \qquad \Bigg| \qquad \mathcal{R}: \quad \begin{aligned} R_1 &= t_2(x) \rightarrow \exists z\ t_3(x,z) \\ R_2 &= t_1(x,y) \wedge t_3(x,z) \rightarrow t_4(y) \end{aligned}$$

Chasing steps

- $\text{chase}_0(D, \mathcal{M} \cup \mathcal{R}) = D = \{s_1(a,b), s_2(a,c)\}$
- $\text{chase}_1(D, \mathcal{M} \cup \mathcal{R}) = \text{chase}_0(D, \mathcal{M} \cup \mathcal{R}) \cup \{t_1(a,b), t_2(a)\}$
- $\text{chase}_2(D, \mathcal{M} \cup \mathcal{R}) = \text{chase}_1(D, \mathcal{M} \cup \mathcal{R}) \cup \{t_3(a,z_0)\}$

Chasing with existential rules

Example

$\mathcal{M}$: $M_1 = s_1(x,y) \rightarrow t_1(x,y)$
$M_2 = s_2(x,y) \rightarrow t_2(x)$

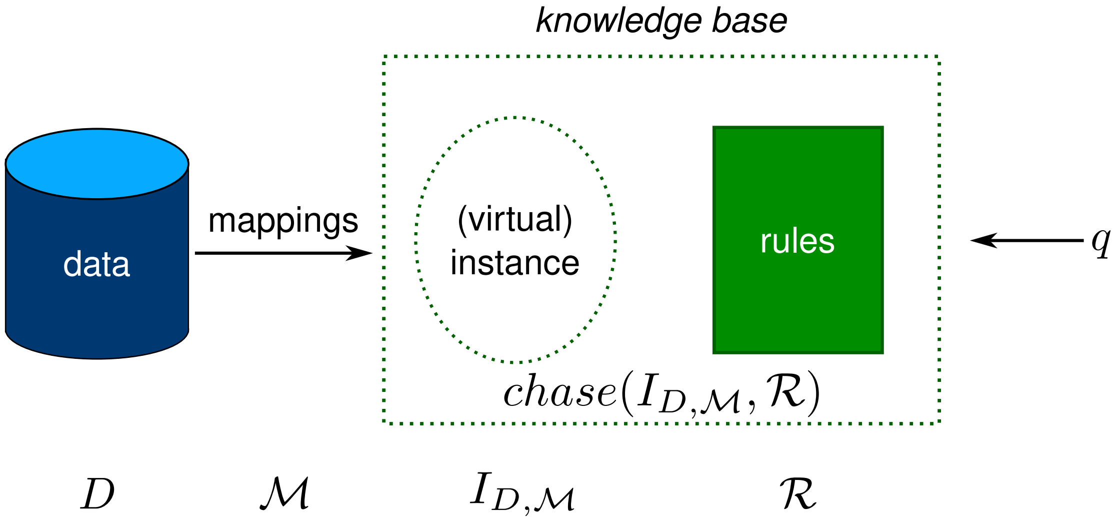
$\mathcal{R}$: $R_1 = t_2(x) \rightarrow \exists z\ t_3(x,z)$
$R_2 = t_1(x,y) \wedge t_3(x,z) \rightarrow t_4(y)$

Chasing steps

- $\text{chase}_0(D, \mathcal{M} \cup \mathcal{R}) = D = \{s_1(a,b), s_2(a,c)\}$
- $\text{chase}_1(D, \mathcal{M} \cup \mathcal{R}) = \text{chase}_0(D, \mathcal{M} \cup \mathcal{R}) \cup \{t_1(a,b), t_2(a)\}$
- $\text{chase}_2(D, \mathcal{M} \cup \mathcal{R}) = \text{chase}_1(D, \mathcal{M} \cup \mathcal{R}) \cup \{t_3(a,z_0)\}$
- $\text{chase}_3(D, \mathcal{M} \cup \mathcal{R}) = \text{chase}_2(D, \mathcal{M} \cup \mathcal{R}) \cup \{t_4(b)\}$

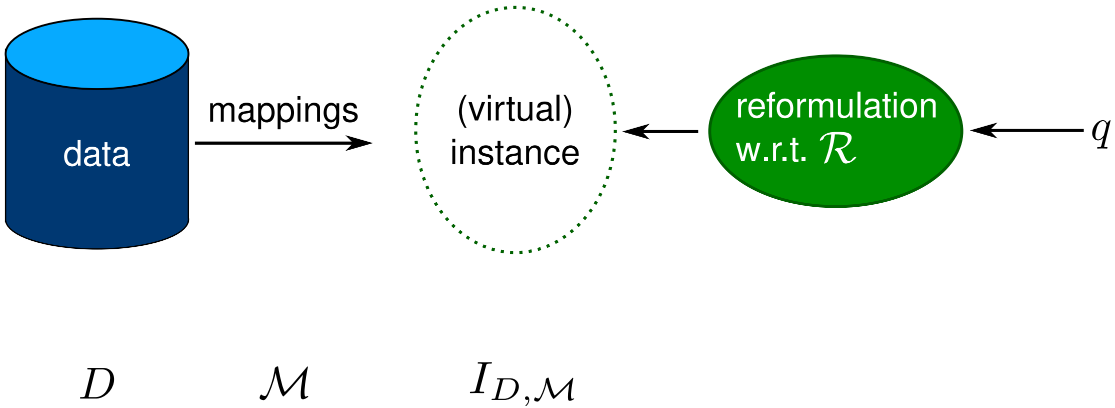
Context

Ontology-Based Data Access with existential rules



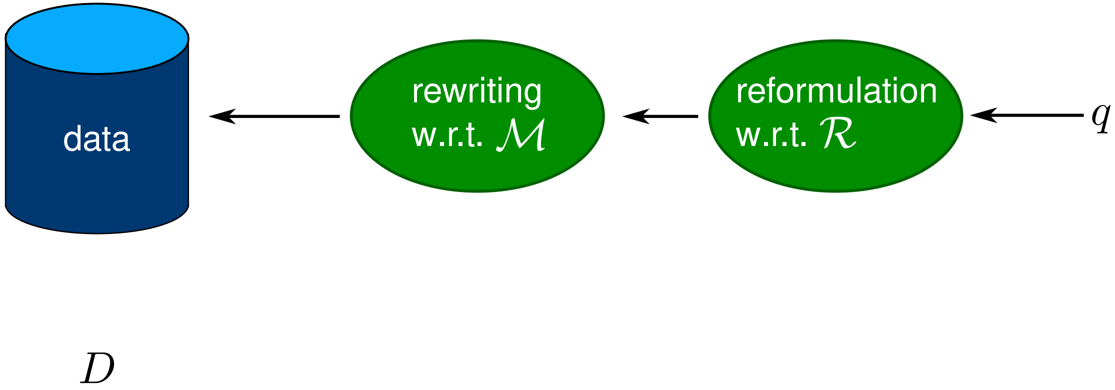
Context

OBDA classical mediation-based query answering method



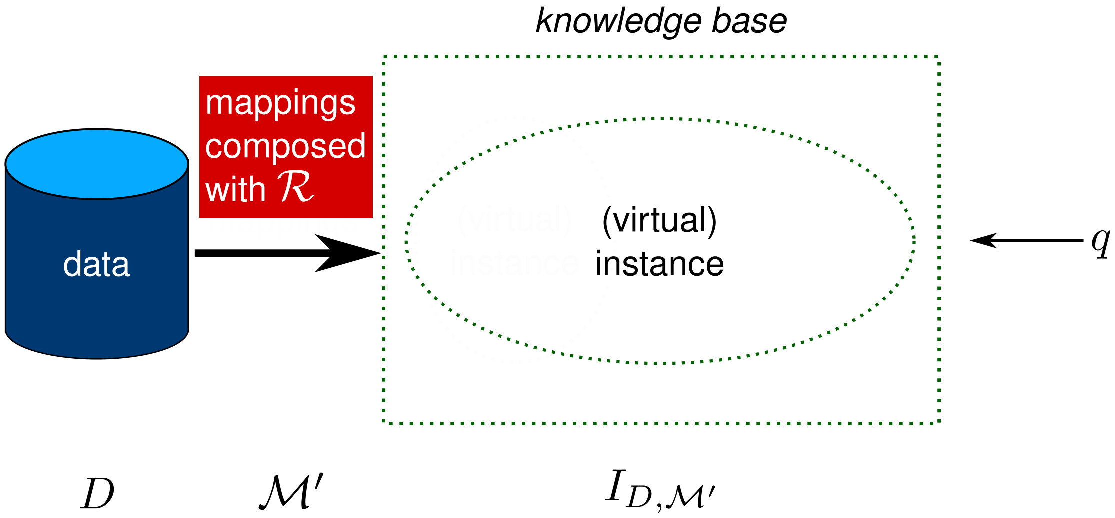
Context

OBDA classical mediation-based query answering method



Context

OBDA query answering by compiling the rules into the mappings



Example

Composing $\mathcal{M}$ with $\mathcal{R}$

$$\mathcal{M}: \quad \begin{array}{l} M_1 = s_1(x,y) \rightarrow t_1(x,y) \\ M_2 = s_2(x,y) \rightarrow t_2(x) \end{array} \quad \Bigg| \quad \mathcal{R}: \quad \begin{array}{l} R_1 = t_2(x) \rightarrow \exists z\ t_3(x,z) \\ R_2 = t_1(x,y) \wedge t_3(x,z) \rightarrow t_4(y) \end{array}$$

$$\mathcal{M}': \begin{array}{l} M_1 = s_1(x,y) \rightarrow t_1(x,y) \\ M_2 = s_2(x,y) \rightarrow t_2(x) \end{array}$$

Example

Composing $\mathcal{M}$ with $\mathcal{R}$

$\mathcal{M}$: $M_1 = s_1(x,y) \rightarrow t_1(x,y)$
$M_2 = s_2(x,y) \rightarrow t_2(x)$

$\mathcal{R}$: $R_1 = t_2(x) \rightarrow \exists z\ t_3(x,z)$
$R_2 = t_1(x,y) \wedge t_3(x,z) \rightarrow t_4(y)$

$\mathcal{M}'$: $M_1 = s_1(x,y) \rightarrow t_1(x,y)$
$M_2 = s_2(x,y) \rightarrow t_2(x)$
$M_3 = R_1 \circ M_2 = s_2(x,y) \rightarrow \exists z\ t_3(x,z)$

Example

Composing $\mathcal{M}$ with $\mathcal{R}$

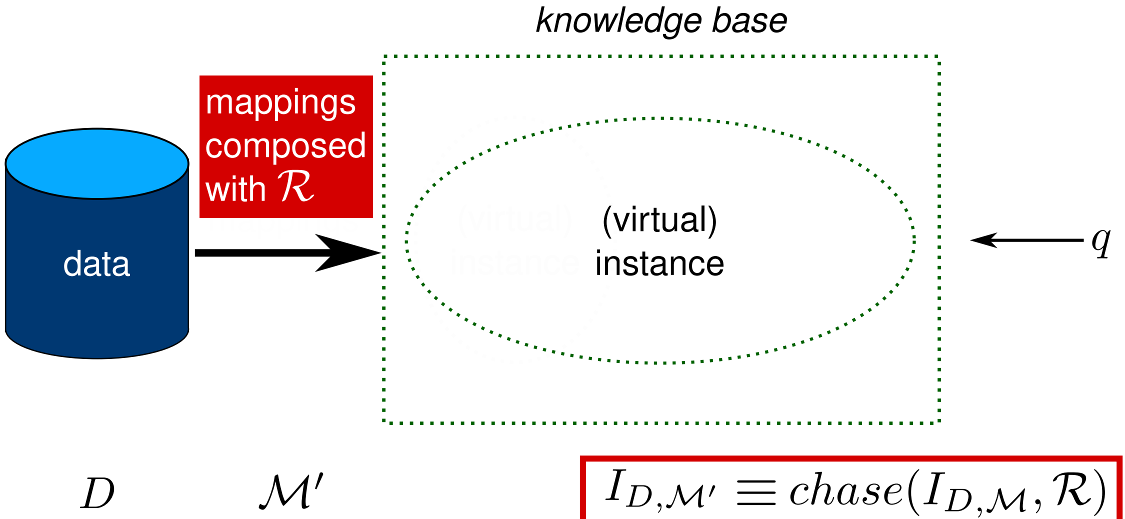
$\mathcal{M}$: $M_1 = s_1(x, y) \rightarrow t_1(x, y)$
$M_2 = s_2(x, y) \rightarrow t_2(x)$

$\mathcal{R}$: $R_1 = t_2(x) \rightarrow \exists z\ t_3(x, z)$
$R_2 = t_1(x, y) \wedge t_3(x, z) \rightarrow t_4(y)$

$\mathcal{M}'$: $M_1 = s_1(x, y) \rightarrow t_1(x, y)$
$M_2 = s_2(x, y) \rightarrow t_2(x)$
$M_3 = R_1 \circ M_2 = s_2(x, y) \rightarrow \exists z\ t_3(x, z)$
$M_4 = (R_2 \circ M_1) \circ M_3 = s_1(x, y) \wedge s_2(x, z) \rightarrow t_4(y)$

Context

OBDA query answering by compiling the rules into the mappings



$$I_{D,\mathcal{M}'} \equiv chase(I_{D,\mathcal{M}}, \mathcal{R})$$

Characterization of the parallelisable rule sets

Research question and contributions

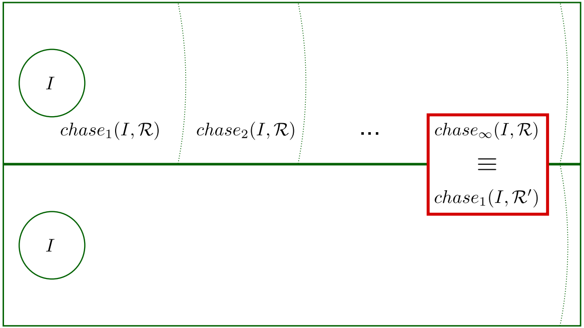
Research question: When can the chase be simulated in a single breadth-first step?

$\mathcal{R}$ is parallelisable if there exists a *finite* rule set *independent from any instance* able to produce an equivalent chase of $\mathcal{R}$ in a single step.

$\Rightarrow$ How to characterize parallelisable sets of rules?

Contributions

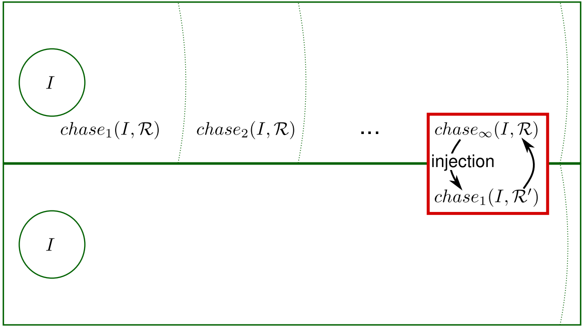
- Parallelisable = Bounded + Pieceful
- Links between parallelisability and rule composition



Parallelisability

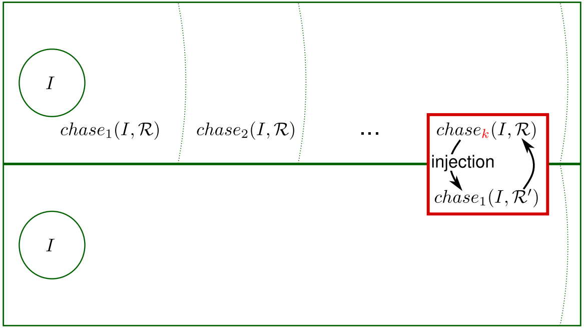
$\mathcal{R}$ is parallelisable if there exists a **finite** rule set $\mathcal{R}'$ such that for any instance I:

1. there is an **injective homomorphism** from $chase_\infty(I, \mathcal{R})$ to $chase_1(I, \mathcal{R}')$
2. there is a homomorphism from $chase_1(I, \mathcal{R}')$ to $chase_\infty(I, \mathcal{R})$



Parallelisability ensures boundedness

$\mathcal{R}$ is bounded if there is k s.t. for any instance I, $chase_k(I, \mathcal{R}) = chase_\infty(I, \mathcal{R})$



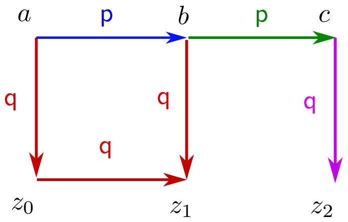
If $\mathcal{R}$ is *parallelisable* then it is bounded, but the converse does not hold

Key notion: Piece

Piece

Minimal set of atoms 'glued' by nulls in the chase or by existential variables in rule heads.

$$p(a, b),$$
$$p(b, c),$$
$$q(a, z_0), q(z_0, z_1), q(b, z_1),$$
$$q(c, z_2)$$



In the following:

We consider that the rules are decomposed in rules having a single-piece head.

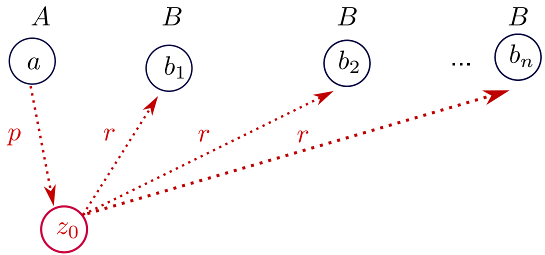
Boundedness does not ensure parallelisability

Prime example (bounded)

$R_1 : A(x) \rightarrow \exists z\ p(x, z)$
$R_2 : p(x, z) \wedge B(y) \rightarrow r(z, y)$

$I_n = \{A(a), B(b_1), \ldots, B(b_n)\}$

$chase_\infty(I_n, \mathcal{R}) =$



For any n, $chase_\infty(I_n, \mathcal{R})$ contains a piece of $n + 1$ atoms, hence this rule set is not parallelisable.

A new class: Pieceful

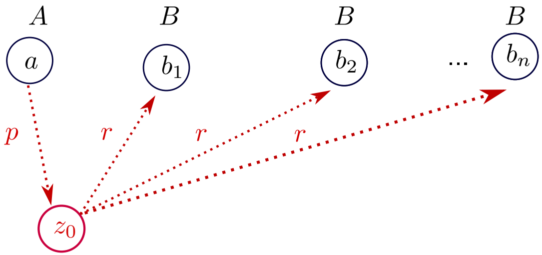
The *frontier* variables of a rule are the shared variables between its body and head.

$\mathcal{R}$ is pieceful if for any trigger (R, π) in any derivation with $\mathcal{R}$,

- either $\pi(frontier(R))$ belongs to the terms of the initial instance
- or $\pi(frontier(R))$ belongs to the terms of atoms brought by a *single* previous rule application.

Prime example is not pieceful

$chase_\infty(I_n, \mathcal{R}) =$



Prime example (bounded)

$R_1 : A(x) \to \exists z\ p(x,z)$
$R_2 : p(x,z) \land B(y) \to r(z,y)$

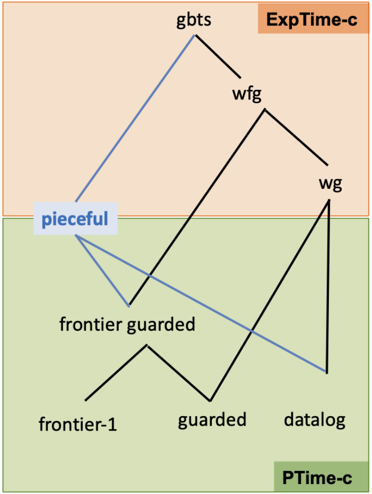
$I_n = \{A(a), B(b_1), \dots, B(b_n)\}$

First trigger: $(R_1, \{x \mapsto a\}$; creates $p(a, z_0)$
Then: $(R_2, \{x \mapsto a, \mathbf{z} \mapsto \mathbf{z_0}, \mathbf{y} \mapsto \mathbf{b_1}\})$

Parallelisability $\Rightarrow$ Piecefulness

Why? If a rule set $\mathcal{R}$ is not pieceful, one can create an instance I_n s.t. $chase(I_n, \mathcal{R})$ has a null that occurs in at least n atoms.

New landscape



(with data complexity of conjunctive query entailment)

Parallelisability = Boundedness + Piecefulness

What we have so far:

- Parallelisability $\Rightarrow$ Boundedness (but the converse is false: see prime example)
- Parallelisability $\Rightarrow$ Piecefulness (but the converse is false: see transitivity)

Parallelisability = Boundedness + Piecefulness

What we have so far:

- Parallelisability $\Rightarrow$ Boundedness (but the converse is false: see prime example)
- Parallelisability $\Rightarrow$ Piecefulness (but the converse is false: see transitivity)

Boundedness + Piecefulness $\Rightarrow$ Parallelisability

- If $\mathcal{R}$ is pieceful, the size of a piece in $chase_k(I, \mathcal{R})$ is bounded independently from I
- If $\mathcal{R}$ is pieceful *and bounded*, the size of a piece in the chase is bounded independently from I. Hence, there is a finite number of 'non-isomorphic' pieces associated with $\mathcal{R}$
- If $\mathcal{R}$ is bounded, each piece (seen as a query) has a finite set of rewritings (reformulations) with $\mathcal{R}$ $\Rightarrow$ roughly, $\mathcal{R}'$ is the set of all rules of the form $rewriting(P) \rightarrow P$

Parallelisability = Boundedness + Piecefulness

What we have so far:

- Parallelisability ⇒ Boundedness (but the converse is false: see prime example)
- Parallelisability ⇒ Piecefulness (but the converse is false: see transitivity)

Boundedness + Piecefulness ⇒ Parallelisability

- If $\mathcal{R}$ is pieceful, the size of a piece in $chase_k(I, \mathcal{R})$ is bounded independently from I
- If $\mathcal{R}$ is pieceful *and bounded*, the size of a piece in the chase is bounded independently from I. Hence, there is a finite number of 'non-isomorphic' pieces associated with $\mathcal{R}$
- If $\mathcal{R}$ is bounded, each piece (seen as a query) has a finite set of rewritings (reformulations) with $\mathcal{R}$ ⇒ roughly, $\mathcal{R}'$ is the set of all rules of the form $rewriting(P) \rightarrow P$

Parallelisabillity is undecidable

Since Pieceful includes Datalog and the boundedness in Datalog is undecidable.

Rule composition

Existential rule composition

An extension of Datalog unfolding

Composition definition

- Keeps rules with single-piece head
- Based on piece-unifiers instead of classical unifiers
- Generates rules inducing every pieces of the chase (growing heads)

Definition of $\mathcal{R}^\star$ the composed rules from $\mathcal{R}$:

Starting from $\mathcal{R}$, we repeatedly compose the rules in $\mathcal{R}^\star$ pairwise

Existential rule composition

An extension of Datalog unfolding

Composition definition

- Keeps rules with single-piece head
- Based on piece-unifiers instead of classical unifiers
- Generates rules inducing every pieces of the chase (growing heads)

Definition of $\mathcal{R}^\star$ the composed rules from $\mathcal{R}$:

Starting from $\mathcal{R}$, we repeatedly compose the rules in $\mathcal{R}^\star$ pairwise

Soundness and completeness of $\mathcal{R}^\star$: $I, \mathcal{R} \models q$ iff $chase_1(I, \mathcal{R}^\star) \models q$

Rule composition on the prime example

$R_1 : A(x) \rightarrow \exists z\ p(x, z)$
$R_2 : p(x, z) \wedge B(y) \rightarrow r(z, y)$

Let us build $\mathcal{R}^{\star}$:
$R_2 \circ R_1 : A(x) \wedge B(y) \rightarrow \exists z\ p(x, z) \wedge r(z, y)$

Rule composition on the prime example

$R_1 : A(x) \to \exists z\ p(x, z)$
$R_2 : p(x, z) \wedge B(y) \to r(z, y)$

Let us build $\mathcal{R}^\star$:
$R_2 \circ R_1 : A(x) \wedge B(y) \to \exists z\ p(x, z) \wedge r(z, y)$
$R_2 \circ (R_2 \circ R_1) : A(x) \wedge B(y) \wedge B(y_1) \to \exists z\ p(x, z) \wedge r(z, y) \wedge r(z, y_1)$

Rule composition on the prime example

$R_1 : A(x) \to \exists z\ p(x,z)$
$R_2 : p(x,z) \wedge B(y) \to r(z,y)$

Let us build $\mathcal{R}^\star$:
$R_2 \circ R_1 : A(x) \wedge B(y) \to \exists z\ p(x,z) \wedge r(z,y)$
$R_2 \circ (R_2 \circ R_1) : A(x) \wedge B(y) \wedge B(y_1) \to \exists z\ p(x,z) \wedge r(z,y) \wedge r(z,y_1)$
etc.
At each step, a new rule $R_2 \circ R^*$, where R^* is the rule created at the preceding step:
$A(x) \wedge B(y) \wedge B(y_1) \ldots B(y_i) \to \exists z\ p(x,z) \wedge r(z,y) \wedge r(z,y_1) \ldots \wedge r(z,y_i)$

What this example shows:

- Completeness requires composition of the form $R \circ R^*$ (and not only $R^* \circ R$ as in datalog)
- $\mathcal{R}^\star$ may be infinite even if $\mathcal{R}$ is bounded, with no finite subset of $\mathcal{R}^\star$ being complete.

Parallelisation by rule composition

Completeness of $\mathcal{R}^\star$

If $\mathcal{R}$ is pieceful, then for any instance I, each piece of $chase_\infty(I, \mathcal{R})$ can be obtained by applying a rule from $\mathcal{R}^\star$ to I

Conjecture

This is true even if $\mathcal{R}$ is not pieceful

Corollary

If $\mathcal{R}$ is parallelisable (ie pieceful and bounded) then it is parallelisable by a finite subset of $\mathcal{R}^\star$

Open issues

Many perspectives

- Better understand rule composition to *compute parallelisation in practice*
- Better understand the properties of the *pieceful* class
- *More succint rule composition* based on rule skolemization?
  It would lead beyond (skolemized) existential rules when rules are not pieceful