Graal

A Toolkit for Query Answering with Existential Rules

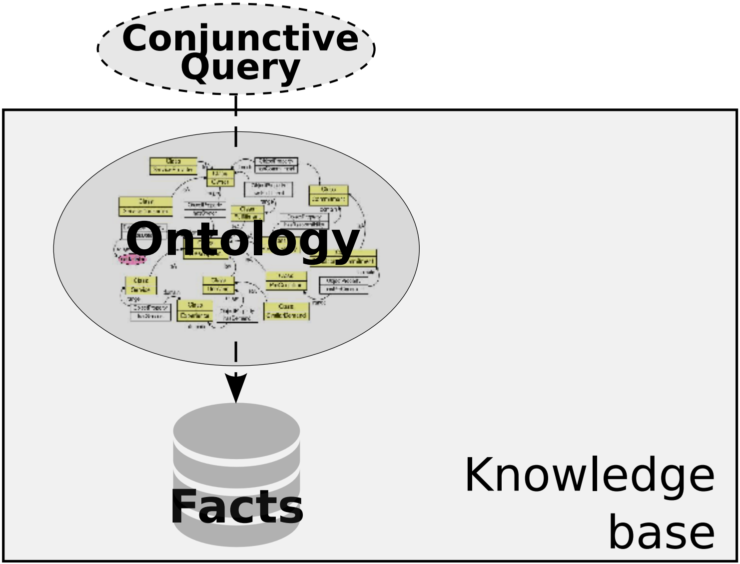
Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, Swan Rocher, and **Clément Sipieter**

-

GraphIK team

RuleML 2015

Ontology-mediated Query Answering



Existential rule framework

=

Datalog^{+} [Calì et al., 2009]

What is Datalog$^+$?

An extension of positive Datalog:

What is Datalog$^+$?

An extension of positive Datalog:

- Existentially quantified variables in rule heads
  $\forall x\ (human(x) \rightarrow \exists y\ parentOf(y, x))$

What is Datalog$^+$?

An extension of positive Datalog:

- Existentially quantified variables in rule heads
  $\forall x\ (human(x) \rightarrow \exists y\ parentOf(y, x))$
- Negative constraints
  $\forall x\ (man(x) \wedge woman(x) \rightarrow \bot)$

What is Datalog$^+$?

An extension of positive Datalog:

- Existentially quantified variables in rule heads
  $\forall x\ (human(x) \rightarrow \exists y\ parentOf(y,x))$
- Negative constraints
  $\forall x\ (man(x) \wedge woman(x) \rightarrow \bot)$
- Equality rules
  $\forall x \forall y \forall z\ (motherOf(y,x) \wedge motherOf(z,x) \rightarrow y = z)$

What is Datalog$^+$?

An extension of positive Datalog:

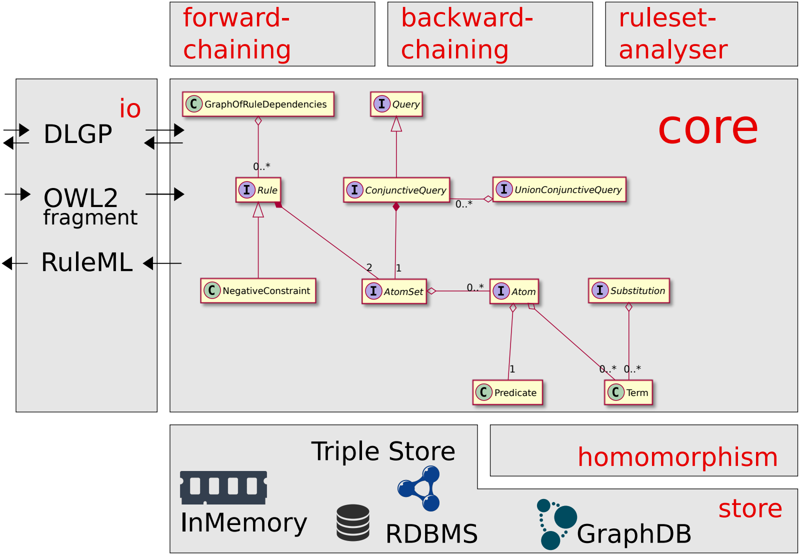
- Existentially quantified variables in rule heads
  $\forall x\ (human(x) \rightarrow \exists y\ parentOf(y,x))$
- Negative constraints
  $\forall x\ (man(x) \wedge woman(x) \rightarrow \bot)$
- Equality rules
  $\forall x \forall y \forall z\ (motherOf(y,x) \wedge motherOf(z,x) \rightarrow y = z)$

Generalizes:

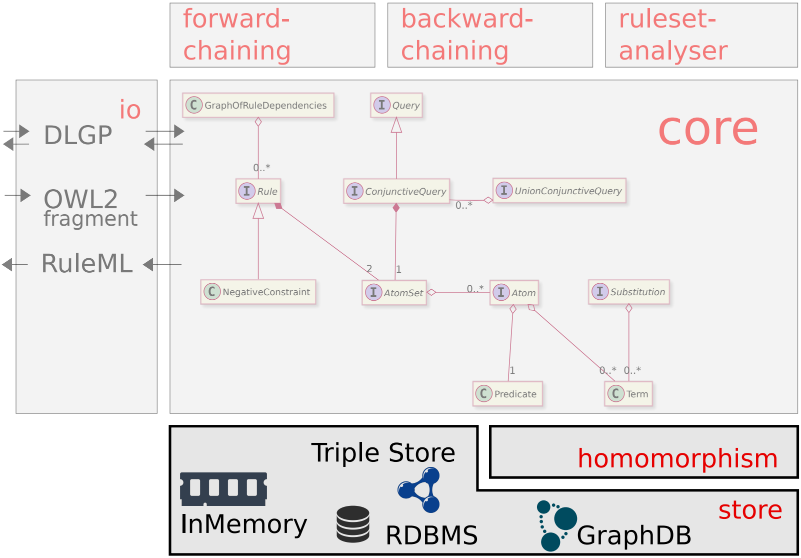
- Horn description logics
  (e.g. DL-Lite, the 3 tractable profiles of OWL2),
- database dependencies (TGDs and EGDs).

Graal, an architecture overview

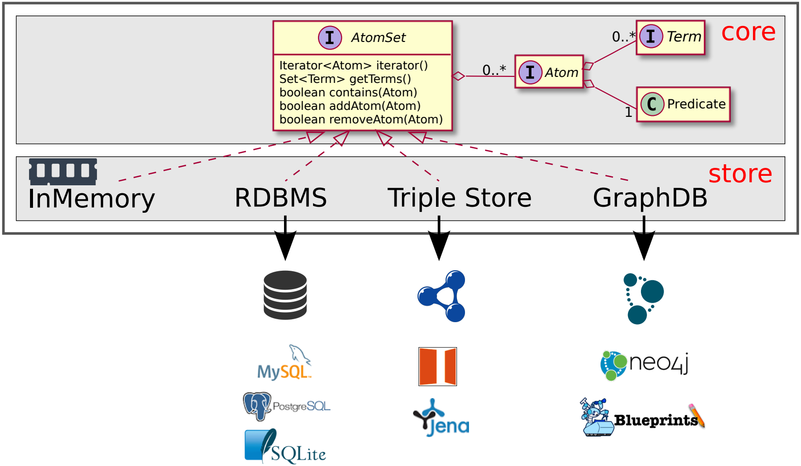
Graal - General architecture



Graal - General architecture



Store: Storing data

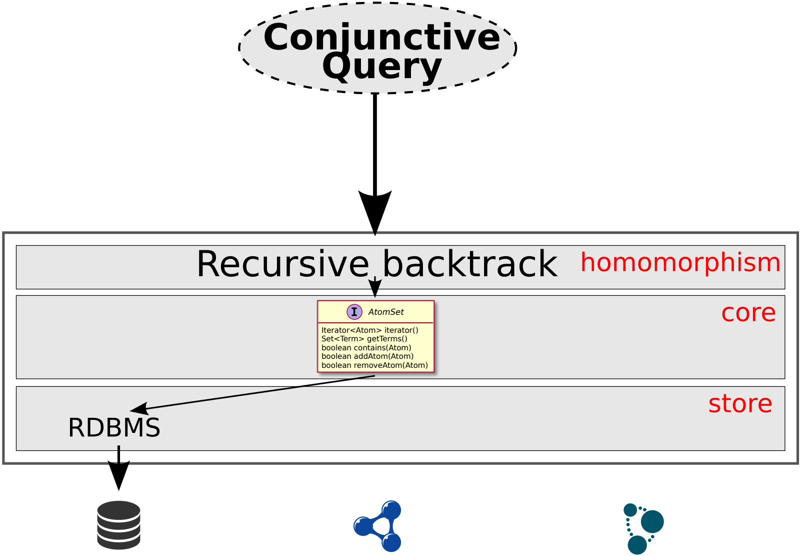


Homomorphism: Querying data

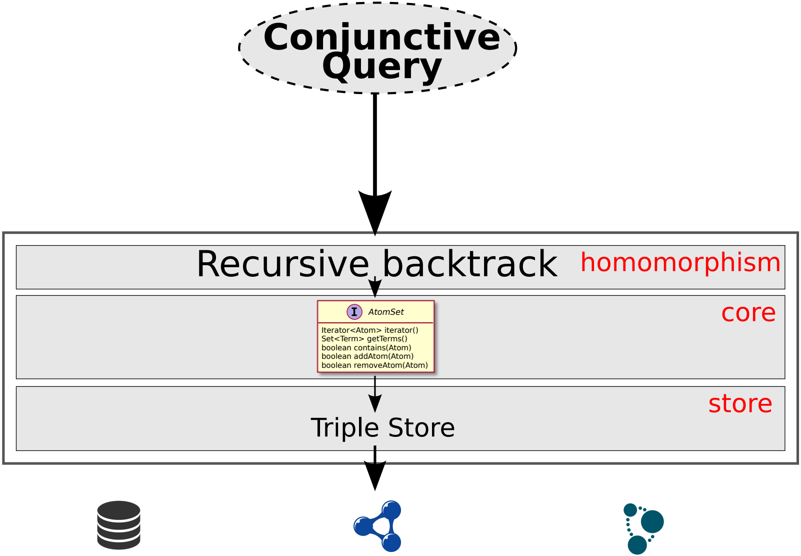
Conjunctive Query

Graal

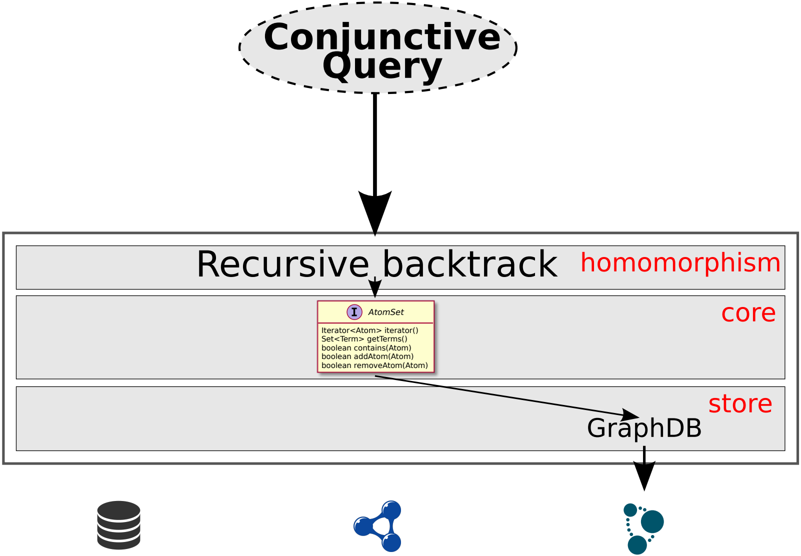
Homomorphism: Querying data



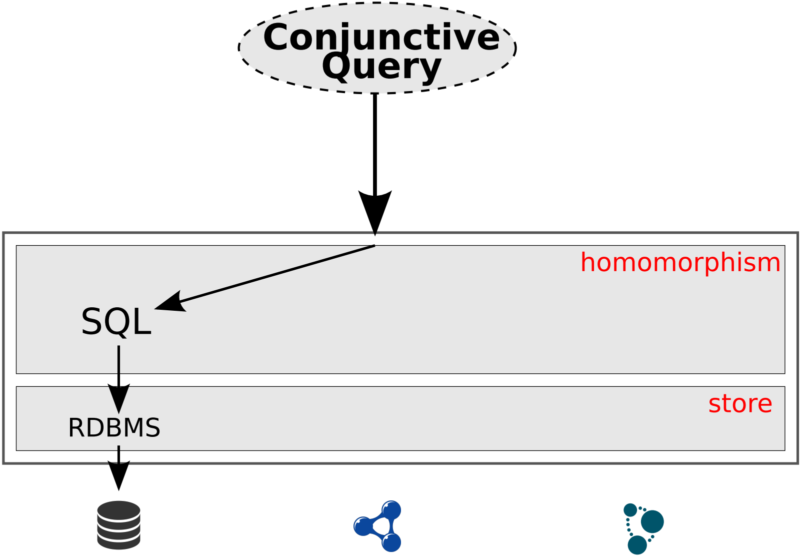
Homomorphism: Querying data



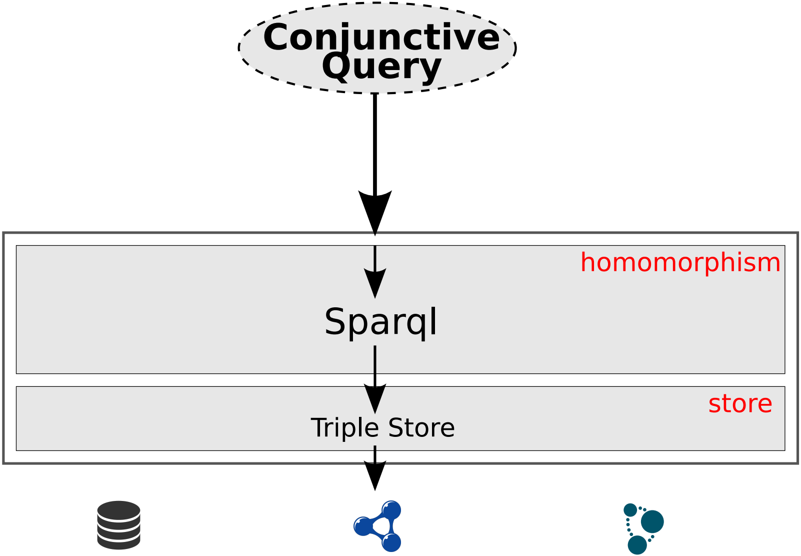
Homomorphism: Querying data



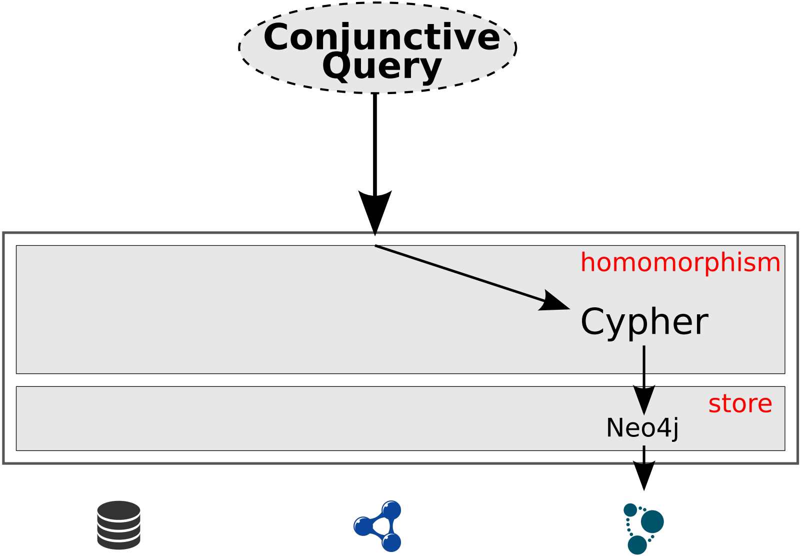
Homomorphism: Querying data



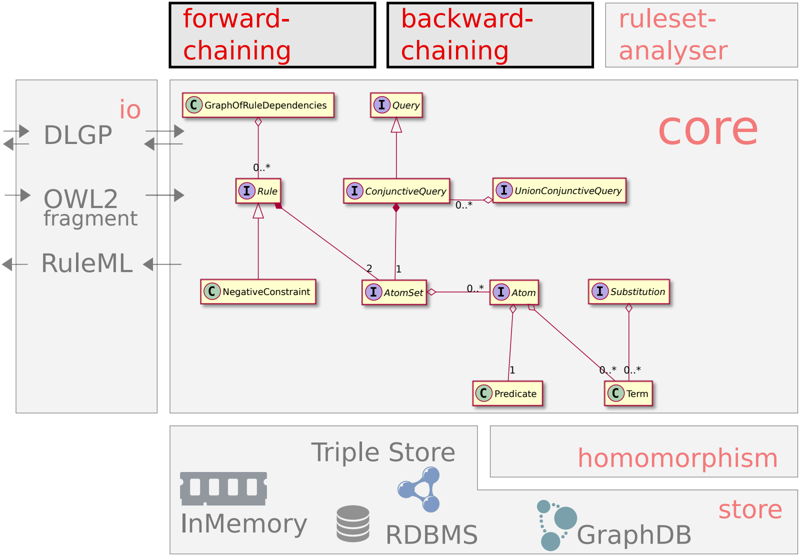
Homomorphism: Querying data



Homomorphism: Querying data

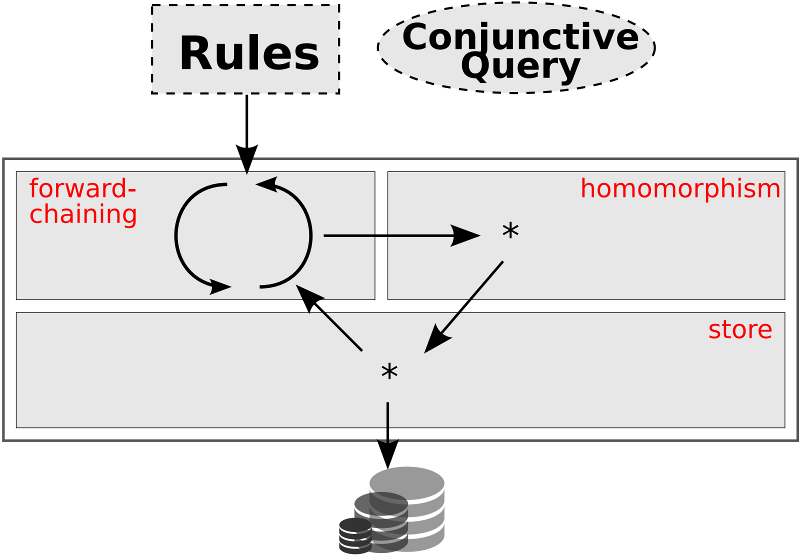


Taking ontology into account

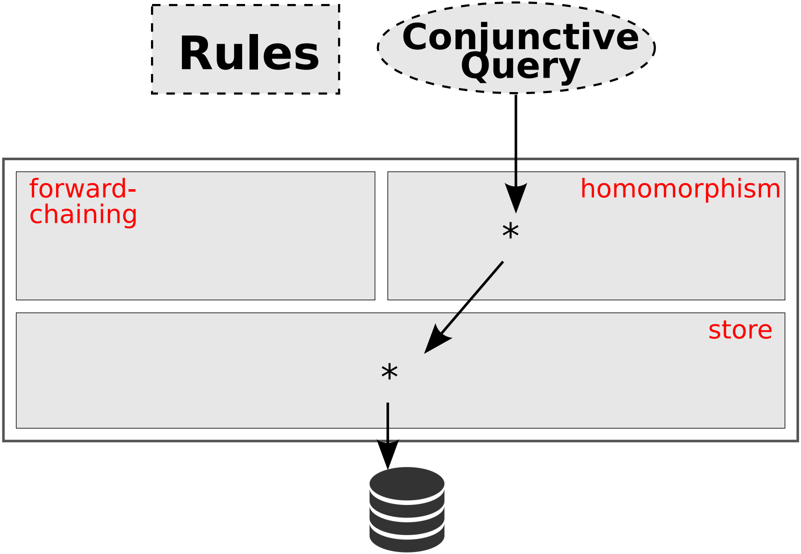


Forward Chaining / Chase

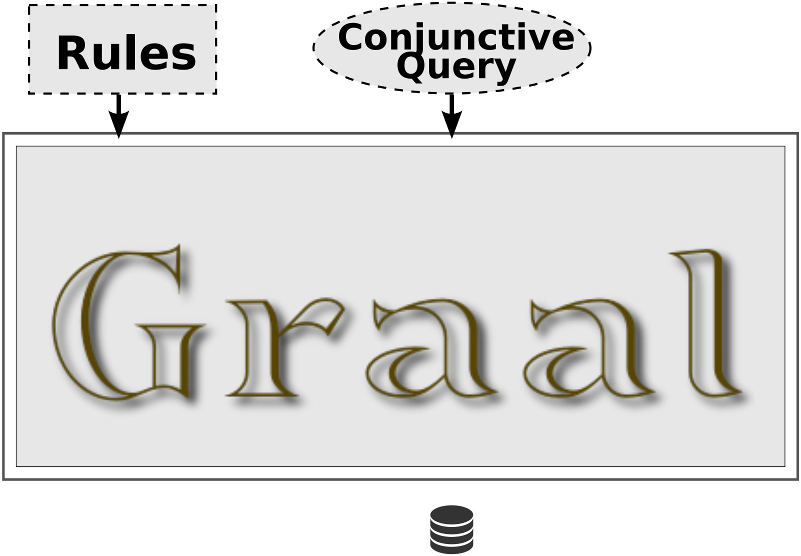
Forward Chaining / Chase



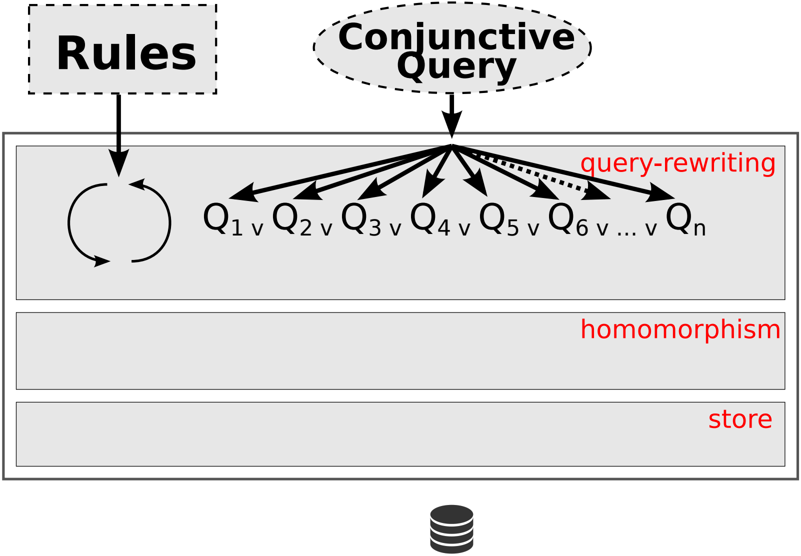
Forward Chaining / Chase



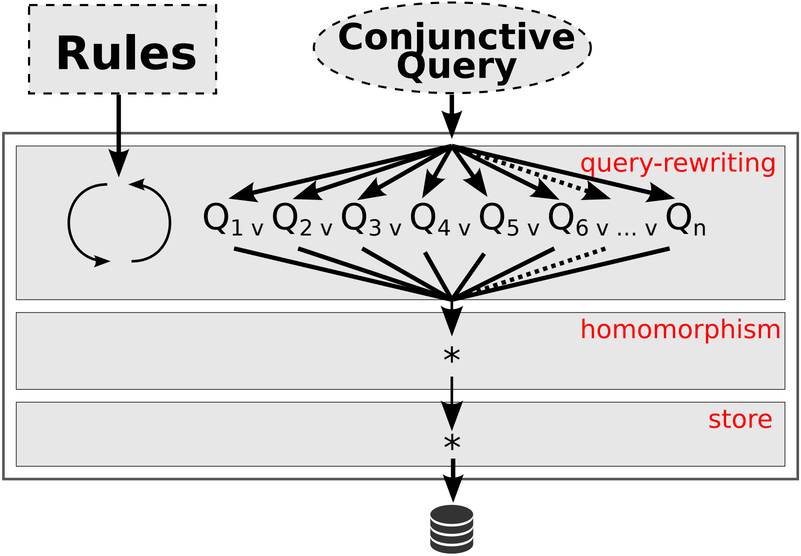
Backward Chaining / Query rewriting



Backward Chaining / Query rewriting



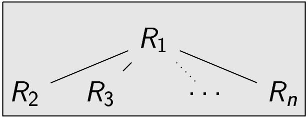
Backward Chaining / Query rewriting



Pure - A compilation-based query rewriter

Efficiency of the Query Rewriting approach in practice?

+ data do not grow
– the size of the rewriting set can be prohibitive in practice



$$q = R_1(x_1, x_2) \wedge \dots \wedge R_1(x_{k-1}, x_k)$$

size of the rewriting set: n^{k-1} **CQs**

$$R_2(x, y) \rightarrow R_1(x, y)$$
$$R_3(x, y) \rightarrow R_1(x, y)$$
$$\dots$$
$$R_n(x, y) \rightarrow R_1(x, y)$$

It is not a theoretical worst-case: it happens often in practice because **hierarchies** are at the heart of ontologies.

Compilation-based Query Rewriting

Preprocess some simple rules known as sources of combinatorial explosion:

$$atom_1 \rightarrow atom_2$$

without existential variable

E.g., subclass, subproperty, domain, range, inverse properties. . .

$\mathcal{R} = \mathcal{R}_C \cup \mathcal{R}_E$

1. Compile $\mathcal{R}_C$ into a **preorder over atoms**
2. Embed this preorder into the rewriting process

$$\mathcal{R}_C \xrightarrow{\text{Offline compilation}} \left.\begin{array}{c}\text{Preorder}\\ \mathcal{R}_E\\ q\end{array}\right] \xrightarrow{\text{Query rewriting}} \begin{array}{c}\text{Union of}\\ \text{“pivotal”}\\ \text{queries}\\ \\ \text{preorder}\end{array}$$

Example

R_0 : $project(x, y, z, w) \rightarrow hasArea(x, y)$
R_1 : $project(x, y, z, w) \rightarrow hasScManager(x, z)$
R_2 : $project(x, y, z, w) \rightarrow hasAdmManager(x, w)$
R_3 : $sensitiveArea(x) \rightarrow area(x)$
R_4 : $security(x) \rightarrow sensitiveArea(x)$
R_5 : $innovation(x) \rightarrow sensitiveArea(x)$
R_6 : $hasScManager(x, y) \rightarrow hasManager(x, y)$
R_7 : $hasAdmManager(x, y) \rightarrow hasManager(x, y)$
R_8 : $isManagerOf(y, x) \rightarrow hasManager(x, y)$
R_9 : $hasManager(y, x) \rightarrow isManagerOf(x, y)$
R_{10} : $manager(x) \rightarrow isManager(x, y)$
R_{11} : $isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z) \rightarrow criticalManager(x)$
R_{12} : $criticalManager(x) \rightarrow isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z)$
R_{13} : $accreditatedManager(x) \rightarrow isManagerOf(x, y) \wedge project(y, z, v, w) \wedge security(z)$

Closure of the compilable rules

R_0 : $project(x, y, z, w) \rightarrow hasArea(x, y)$
R_1 : $project(x, y, z, w) \rightarrow hasScManager(x, z)$
R_2 : $project(x, y, z, w) \rightarrow hasAdmManager(x, w)$
R_3 : $sensitiveArea(x) \rightarrow area(x)$
R_4 : $security(x) \rightarrow sensitiveArea(x)$
R_5 : $innovation(x) \rightarrow sensitiveArea(x)$
R_6 : $hasScManager(x, y) \rightarrow hasManager(x, y)$
R_7 : $hasAdmManager(x, y) \rightarrow hasManager(x, y)$
R_8 : $isManagerOf(y, x) \rightarrow hasManager(x, y)$
R_9 : $hasManager(y, x) \rightarrow isManagerOf(x, y)$

We compute all **inferred rules obtained by composition**:

R_a : $R_1 \cdot R_6 = project(x, y, z, w) \rightarrow hasManager(x, z)$
R_b : $R_2 \cdot R_7 = project(x, y, z, w) \rightarrow hasManager(x, w)$
R_c : $R_4 \cdot R_3 = security(x) \rightarrow area(x)$
R_d : $R_5 \cdot R_3 = innovation(x) \rightarrow area(x)$
R_e : $R_6 \cdot R_9 = hasScManager(x, y) \rightarrow isManagerOf(y, x)$
R_f : $R_7 \cdot R_9 = hasAdmManager(x, y) \rightarrow isManagerOf(y, x)$
R_g : $R_a \cdot R_9 = project(x, y, z, w) \rightarrow isManagerOf(z, x)$
R_h : $R_b \cdot R_9 = project(x, y, z, w) \rightarrow isManagerOf(w, x)$

Preorder

$$\frac{isManagerOf(x,y)}{\begin{array}{c}hasScManager(y,x)\\hasAdmManager(y,x)\\hasManager(y,x)\\project(y,z,x,w)\\project(y,z,w,x)\end{array}}$$

$$\frac{area(x)}{\begin{array}{c}security(x)\\innovation(x)\\sensitiveArea(x)\end{array}}$$

$$\frac{hasManager(x,y)}{\begin{array}{c}hasScManager(x,y)\\hasAdmManager(x,y)\\isManagerOf(y,x)\\project(x,z,y,w)\\project(x,z,w,y)\end{array}}$$

$$\frac{hasAdmManager(x,y)}{project(x,z,w,x)}$$

$$\frac{sensitiveArea(x)}{\begin{array}{c}security(x)\\innovation(x)\end{array}}$$

$$\frac{hasArea(x,y)}{project(x,y,z,w)}$$

$$\frac{hasScManager(x,y)}{project(x,z,x,w)}$$

$\preccurlyeq$-Homomorphism

$$Q = isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z)$$

$$\begin{aligned} F = {} & isManagerOf(Alice, Project1) \wedge \\ & project(Project1, Spying, Alice, Bob) \wedge \\ & security(Spying) \end{aligned}$$

$$\frac{hasArea(x, y)}{project(x, y, z, w)} \qquad \frac{sensitiveArea(x)}{\begin{array}{c} security(x) \\ innovation(x) \end{array}} \qquad \ldots$$

COMPILED

$\preccurlyeq$-Homomorphism

$$
\begin{aligned}
Q &= isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z) \\
F &= isManagerOf(Alice, Project1) \wedge \\
&\quad project(Project1, Spying, Alice, Bob) \wedge \\
&\quad security(Spying)
\end{aligned}
$$

$$h = \{\{x, Alice\}, \{y, Project1\} \dots$$

$$\frac{hasArea(x, y)}{project(x, y, z, w)} \qquad \frac{sensitiveArea(x)}{\begin{array}{c} security(x) \\ innovation(x) \end{array}} \qquad \dots$$

COMPILED

$\preccurlyeq$-Homomorphism

$$Q = isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z)$$
$$F = isManagerOf(Alice, Project1) \wedge$$
$$project(Project1, Spying, Alice, Bob) \wedge$$
$$security(Spying)$$

$$h = \{\{x, Alice\}, \{y, Project1\}, \{z, Spying\}\}$$

$$\frac{hasArea(x, y)}{project(x, y, z, w)} \qquad \frac{sensitiveArea(x)}{\begin{array}{c} security(x) \\ innovation(x) \end{array}} \qquad \ldots$$

COMPILED

Query rewriting using $\preccurlyeq$-Homomorphism

$Q = criticalManager(x)$

[Optimized rewriting: **3** PCQs] [Classical rewriting: **38** CQs]

R_{11} : $isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z) \rightarrow criticalManager(x)$
R_{13} : $accreditatedManager(x) \rightarrow isManagerOf(x, y) \wedge project(y, z, v, w) \wedge security(z)$

$\vdots$

$$\frac{hasArea(x, y)}{project(x, y, z, w)} \qquad \frac{sensitiveArea(x)}{\begin{array}{c} security(x) \\ innovation(x) \end{array}} \qquad \ldots$$

COMPILED

Query rewriting using $\preccurlyeq$-Homomorphism

$$Q = criticalManager(x)$$
$$Q' = isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z)$$

[Optimized rewriting: **3** PCQs] [Classical rewriting: **38** CQs]

R_{11} : $isManagerOf(x, y) \wedge hasArea(y, z) \wedge sensitiveArea(z) \rightarrow criticalManager(x)$
R_{13} : $accreditatedManager(x) \rightarrow isManagerOf(x, y) \wedge project(y, z, v, w) \wedge security(z)$

$\vdots$

$$\frac{hasArea(x, y)}{project(x, y, z, w)} \qquad \frac{sensitiveArea(x)}{\begin{array}{c} security(x) \\ innovation(x) \end{array}} \qquad \dots$$

COMPILED

Query rewriting using $\preccurlyeq$-Homomorphism

$$Q = criticalManager(x)$$
$$Q' = isManagerOf(x,y) \wedge hasArea(y,z) \wedge sensitiveArea(z)$$
$$Q'' = accreditatedManager(x)$$

[Optimized rewriting: **3** PCQs] [Classical rewriting: **38** CQs]

R_{11} : $isManagerOf(x,y) \wedge hasArea(y,z) \wedge sensitiveArea(z) \rightarrow criticalManager(x)$
R_{13} : $accreditatedManager(x) \rightarrow isManagerOf(x,y) \wedge project(y,z,v,w) \wedge security(z)$

$$\vdots$$

$$\frac{hasArea(x,y)}{project(x,y,z,w)} \qquad \frac{sensitiveArea(x)}{\begin{array}{c} security(x) \\ innovation(x) \end{array}} \qquad \ldots$$

COMPILED

Results - Impact of compilation on rewriting sizes

Benchmark:
translation of DL-LiteR ontologies:

- Adolena (102 rules, 75% compilable)
- Vicodi (222 rules, 100% compilable)
- OpenGalen2-Lite (51k rules, 55% compilable)
- OBOprotein (43k rules, 82% compilable)

Each ontology is provided with 5 queries.

		UCQ	p-UCQ
A	Q1	27	2
	Q2	50	2
	Q3	104	1
	Q4	224	2
	Q5	624	1
V	Q1	15	1
	Q2	10	1
	Q3	72	1
	Q4	185	1
	Q5	30	1
G	Q1	2	1
	Q2	1152	1
	Q3	488	5
	Q4	147	1
	Q5	324	19
O	Q1	27	20
	Q2	1356	1264
	Q3	33887	1
	Q4	34733	682
	Q5	36612	-

Query evaluation (Future work)

Union of Pivotal Queries (p-UCQ)
+
Preorder (P)

Query evaluation (Future work)

Union of Pivotal Queries (p-UCQ)
+
Preorder (P)

- Unfold p-UCQ into a classical UCQ

Query evaluation (Future work)

Union of Pivotal Queries (p-UCQ)
+
Preorder (P)

- Unfold p-UCQ into a classical UCQ
- Use P in a backtrack algorithm

Query evaluation (Future work)

Union of Pivotal Queries (p-UCQ)
+
Preorder (P)

- Unfold p-UCQ into a classical UCQ
- Use P in a backtrack algorithm
- Saturate data with P

Query evaluation (Future work)

Union of Pivotal Queries (p-UCQ)
+
Preorder (P)

- Unfold p-UCQ into a classical UCQ
- Use P in a backtrack algorithm
- Saturate data with P
- Use Semi-Conjunctive Queries

https://graphik-team.github.io/graal/

Graal Homepage

Home Documentation Publications Experiments Downloads Sources

Graal is a Java toolkit dedicated to querying knowledge bases within the framework of existential rules, aka Datalog+/-.

The main features of Graal are the following:

- a basic layer that provides generic interfaces to store and query various kinds of data without considering the rules;
- 'saturation' algorithms, which apply rules on the data in a forward chaining manner;
- 'query rewriting' algorithms, which reformulate a conjunctive query into a set (or 'union') of conjunctive queries;
- utility tools:
  - a format called DLGP (for 'datalog+') and its parser;
  - a tool called Kiabora, which performs a structural analysis of an existential rule set to determine its decidability properties; it also allows to decompose rules;
  - a translator from OWL 2 files to *dlgp*;
  - a translator from *dlgp* to RuleML.

Related standalone tools

- Query rewriter (Pure: basic and compilation-based versions)
- Translator from OWL 2 to DLGP v2
- Translator from DLGP v2 to RuleML
- Online Kiabora (works with DLGP v1)

Results - Impact of compilation on rewriting times

		Pure	Pure_C	Pure_C to UCQ
A	Q1	190	20	140
	Q2	100	10	50
	Q3	180	20	40
	Q4	290	10	140
	Q5	1510	10	450
V	Q1	20	10	10
	Q2	20	10	20
	Q3	120	10	80
	Q4	130	10	70
	Q5	20	10	20
G	Q1	10	10	20
	Q2	1070	60	630
	Q3	1030	80	270
	Q4	30	20	20
	Q5	900	40	100
O	Q1	450	140	150
	Q2	1170	1120	1880
	Q3	TO	100	558000
	Q4	TO	440	TO
	Q5	TO	TO	TO

TO = 10min.

Results - Impact of compilation on rewriting times

		Pure	Pure_C	Pure_C to UCQ	Nyaya	Requiem	Iqaros	tw	Rapid
A	Q1	190	20	140	1130	270	60	20	20
	Q2	100	10	50	870	110	60	20	30
	Q3	180	20	40	2370	140	200	10	40
	Q4	290	10	140	5560	260	140	20	50
	Q5	1510	10	450	33210	470	580	20	100
V	Q1	20	10	10	20	20	20	10	10
	Q2	20	10	20	60	20	20	10	10
	Q3	120	10	80	30	70	30	10	30
	Q4	130	10	70	30	140	40	10	40
	Q5	20	10	20	30	80	50	20	30
G	Q1	10	10	20	-	50	50	10	10
	Q2	1070	60	630	-	209050	5870	20	80
	Q3	1030	80	270	-	259110	9190	30	60
	Q4	30	20	20	-	190260	780	10	20
	Q5	900	40	100	-	238460	7410	30	50
O	Q1	450	140	150	-	3450	6680	20	30
	Q2	1170	1120	1880	-	21790	27820	580	960
	Q3	TO	100	558000	-	TO	TO	80	620
	Q4	TO	440	TO	-	TO	139990	1240	14700
	Q5	TO	TO	TO	-	TO	TO	TO	562230

TO = 10min.