Existential Rules: A Graph-Based View
Marie-Laure Mugnier

▶ To cite this version:


HAL Id: lirmm-00763474
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00763474
Submitted on 16 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Existential Rules:
A Graph-based View

Marie-Laure Mugnier
University of Montpellier
Ontology-based Data Access (OBDA)

Adding an ontological layer:

- to abstract from a specific database schema
- to provide a unified view of multiple sources
- to infer new facts, thus allowing for data incompleteness
Outline

- Existential rules: a logic- and graph-based framework
- Decidability and algorithmic issues
  - Focus on:
    - tree-shaped saturation in forward chaining
    - piece-based unification in backward chaining
- A (graph) tool for combining decidable classes of rules
Data / Facts

Relational Database

<table>
<thead>
<tr>
<th>parentOf</th>
<th>Male</th>
<th>Fem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

RDF (Semantic Web)

∃x( parentOf(a,b) ∧ parentOf(a,c) ∧ parentOf(c,x) ∧ F(a) ∧ M(b) )

Abstraction in first-order logic

Or in graphs / hypergraphs
Ontology: Existential Rules

∀x ∀y ( siblingOf(x,y) → ∃ z (parentOf(z,x) ∧ parentOf(z,y)) )

Simplified form: siblingOf(x,y) → parentOf(z,x) ∧ parentOf(z,y)

- Same as Tuple Generating Dependencies (TGDs)
- See also Datalog+/-
- Same as the logical translation of Conceptual Graph rules
- Generalize Description Logics used for OBDA (DL-Lite, $\mathcal{EL}$)
∀X ∀Y ( B[X, Y] → ∃Z H[X, Z] )

∀x ∀y (siblingOf(x,y) → ∃z (parentOf(z,x) ∧ parentOf(z,y)))
Value Invention

\[ R = \forall x \forall y \ (\text{siblingOf}(x,y) \rightarrow \exists z \ (\text{parentOf}(z,x) \land \text{parentOf}(z,y))) \]

\[ F = \text{siblingOf}(a,b) \]

A rule body \rightarrow head is applicable to a fact \( F \) if there is a homomorphism \( h : \text{body} \rightarrow F \)

Then \( h(head) \) can be « added » to \( F \) with renaming existential variables of head

\[ F' = \exists z0 \ (\text{siblingOf}(a,b) \land \text{parentOf}(z0,a) \land \text{parentOf}(z0,a)) \]
Logical / Graphical Framework

Knowledge Base
- Fact(s)
- Existential Rules
- + Equality atoms
- Constraints

(Union of)
Conjunctive Query
(\(\forall\) \(\exists\) \(f[X]\))

Negative constraint:
\(- (\exists X B[X]) \text{ or } \forall X (B[X] \rightarrow \bot)\)
« B[X] must not be found »

Positive constraint:
\(\forall X \forall Y (B[X, Y] \rightarrow \exists Z H[X, Z])\)
« if B[X,Y] is found then H[X,Z] must also be found »
Similar Framework: Datalog +/-

Knowledge Base

- TGDs
- EGDs
- Negative Constraints

Database

(Union of)
Conjunctive Query

Answers?

[Cali Gottlob Lukasiewicz PODS 2009]

Tuple Generating Dependency = (pure) existential rule

Equality Generating Dependency: \( \forall X \ (B[X] \rightarrow x = e) \)
The Conceptual Graph Origins

- Conceptual graphs introduced in [Sowa 76] [Sowa 84]
- Specific research line by Montpellier’s group since 1992

« Graph-based » knowledge representation and reasoning

« Graph-Based Knowledge Representation: Computational Foundations of Conceptual Graphs », Chein & M..., Springer, 2009
Conceptual Graph Vocabulary:

1. partially (pre-)ordered set of concepts

[screenshots from CoGui, http://www.lirmm.fr/cogui]
Conceptual Graph Vocabulary:
2. partially (pre-)ordered set of relations with their signature [any relation arity allowed]

Logical translation ($\Phi$) of the vocabulary: very simple rules

\[ p < q \quad \forall x_1 \ldots x_k \ ( p(x_1 \ldots x_k) \rightarrow q(x_1 \ldots x_k) ) \]

Signature of $r$ \[ \forall x_1 \ldots x_k \ ( p(x_1 \ldots x_k) \rightarrow t_{i1}(x_1) \ldots t_{ik}(x_k) ) \]
Basic Conceptual Graph

Logical translation ($\Phi$): existentially closed conjunction of atoms

$$\exists x \exists y \left( \text{Girl}(Eva) \land \text{Child}(x) \land \text{Toy}(y) \land \text{Train}(y) \land \text{sisterOf}(Eva,x) \land \text{playWith}(Eva,y) \land \text{playWith}(x,y) \right)$$

Allows to represent facts and conjunctive queries
Homomorphism (with concept/relation preorders integrated)

 Logical soundness [Sowa 84] and completeness [Chein M... 92]:

there is a homomorphism from $Q$ to $F$ iff

$\Phi(Q)$ is entailed by $\Phi(F)$ and $\Phi$(vocabulary)

The Basic CG fragment restricted to binary relations is equivalent to RDFS [Baget ISWC’ 05] [Baget+ ICCS’ 10]
Richer Fragments (nested graphs, rules, constraints, + negation, ...)

- Rule: pair of basic conceptual graphs

\[ \forall x \forall y (\text{Human}(x) \land \text{Human}(y) \land \text{siblingsOf}(x,y) \rightarrow \exists z (\text{Adult}(z) \land \text{parentOf}(z,x) \land \text{parentOf}(z,y))) \]

- Sound and complete forward chaining and backward chaining [Salvat M... 1996]
- Several ways of combining rules and constraints [Baget M... JAIR 2002]

The existential rule framework can be seen as a fragment of CGs with a flat vocabulary
Outline

- Existential rules: a logic- and graph-based framework
- Decidability and algorithmic issues
  - Focus on:
    - tree-shaped saturation in forward chaining
    - piece-based unification in backward chaining
- A (graph) tool for combining decidable classes of rules
Basic Problem

Given a KB $\mathcal{K} = (F, R)$ and a (Boolean) conjunctive query $Q$, is $Q$ entailed by $\mathcal{K}$?

Conjunctive Query Entailment
Forward vs Backward Chaining

**FC**  Fact saturation (« chase », « bottom-up »)

**BC**  Query rewriting (« top-down »)

[Decomposition into 2 steps: DL-Lite]
Decidability Issues

- Entailment is not decidable

- Many decidable classes exhibited in databases and KR

- Three generic kinds of properties ensuring decidability:
  
  - Saturation by Forward Chaining halts (« finite expansion set », \textit{fes})
  
  - Query rewriting by Backward Chaining halts (« finite unification set », \textit{fus})
  
  - Saturation by Forward Chaining may not halt \textit{but} the generated facts have a\emph{tree-like structure} (« bounded treewidth set », \textit{bts})

None of these properties is \textit{recognizable} [Baget+ KR 10] but they provide\textit{generic} algorithms
Decomposition Tree / Treewidth

Decomposition tree:
1) each node (term) appears in a bag
2) each hyperedge (atom) has all its nodes in a bag
3) for each node \(x\), the subgraph induced by the bags containing \(x\) is connected

Width of a tree decomposition = \(\text{max}\) number of nodes in a bag (minus 1)
Treewidth of a graph = \(\text{min}\) width over all decomposition trees of this graph
Bounded Treewidth of the Derived Facts (*bts*)

Essentially [Cali Gottlob Kifer KR’08]

\( \mathcal{R} \) is *bts* if FC with \( \mathcal{R} \) generates facts with bounded treewidth
i.e., for any fact \( F \), there is an integer \( b \) s.t.
any fact \( \mathcal{R} \)-derived from \( F \) has treewidth bounded by \( b \)

*fes (finite saturation)* is included in *bts*
(bound given by the number of terms in the finite « saturated fact »)

The decidability proof does not provide a halting algorithm
(relies on the bounded treewidth model property [Courcelle 90])
(Partial) Inclusion Map of Decidable Classes

Finite saturation (fes)
- wa-GRD
- weakly-acyclic
  - GRD
  - acyclic
    - jointly-acyclic
      - joint-GRD
Finite query rewriting (fus)
- w-sticky-join
- w-sticky
- sticky-join
- sticky
  - domain-r.
Tree-shaped saturation (bts)
- glut-fg
  - jointly-fg
    - weakly-frontier-guarded
      - weakly-guarded
      - frontier-guarded
        - frontier-1

Datalog
- weakly-acyclic
- acyclic
- jointly-acyclic
- GRD
- atomic body
- inclusion dependency
Some Recognizable bts (and not fes) Classes of Rules

**Frontier**: variables shared by the body and the head

Guard only the frontier

- \( r(x,y) \land r(y,z) \rightarrow r(y,u) \)
- \( r(y,u) \land r(z,u) \)

The frontier has size 1

Guard only affected variables from the frontier

- \([\text{Baget+ KR’ 10}]

Guard only affected variables (i.e. possibly mapped to new existentials)

- \([\text{Cali+ KR’ 08}]

An atom in the body guards all the body variables

- \([\text{Cali+ KR’ 08}]

These classes are moreover « greedy bts » => a halting algorithm \([\text{Baget+ IJCAI’ 11}]

\[ r(x,y) \land r(y,z) \land s(x,y,z) \rightarrow r(y,u) \land r(z,u) \]
Greedy *bts*

\[ R_1 = p(x,y) \Rightarrow p(y,z) \]
\[ R_2 = p(x,y) \land q(x,z) \Rightarrow r(x,y,t) \land p(y,t) \]

\[ F = p(a,b) \]

Greedy construction of a decomposition tree of the derived fact with bounded width
For any fact, for each rule application, frontier variables not being mapped to initial terms are *jointly* mapped to variables occurring in atoms added by a single previous rule application.
Main Ideas of the Algorithm for *gbts* (1)

1. Bag pattern = \{ homomorphisms from *part of a rule body* to « current fact » that use some *terms of the bag* \}
   
   \(\rightarrow\) A rule is applicable to the current fact *iff* a bag pattern contains its body
   
   \(\rightarrow\) FC can be performed on the decorated tree

2. Equivalence relation on bags

   Only one bag per equivalence class is developed
   The other nodes are *blocked*

Bounded number of equivalence classes \(\rightarrow\) finite « full blocked tree » \(T^*\)
Main Ideas of the Algorithm for \textit{gbts} (2)

Query this finite decomposition tree

[Baget+ IJCAI 2011] \( Q \) seen as a rule « \( Q \to \text{match} \) »

Q is entailed iff it occurs in a bag pattern
\hspace{1cm}i.e. \( Q \) maps by homomorphism to \( \text{atoms}(T^*) \)

[Thomazo+ KR 2012] offline /online separation

(1) compilation: tree \( T^* \) built independently from any query
(2) querying: any \( Q \) is entailed iff it maps by \( *\text{-homomorphism} \) to \( T^* \)
\hspace{1cm}i.e. \( Q \) maps by homomorphism to a bounded « development » of \( T^* \)
Backward Chaining: Unification Step

$R = r(x) \rightarrow p(x,y)$

$Q = p(u,v) \land p(u,w) \land p(v,w)$

Atomic unification:
$u \rightarrow x \quad v \rightarrow y$

$Q_1 = r(x) \land p(x,w) \land p(y,w)$

Soundness lost!

Indeed let $F = Q_1$
\begin{align*}
\textit{saturation}(F,R) & \equiv F \\
Q & \text{ does not map to } F
\end{align*}

Existentials in rule heads produce a structure that must be taken into account
Key Notion: « Piece »

Given a subset $T$ of its variables, a set of atoms is partitioned into pieces.

A piece = all atoms linked by a « path » of variables not belonging to $T$

$T = \{ u \}$

Piece 1 = \{ $p(u,v)$, $p(v,w)$, $p(w,u)$ \}
Piece 2 = \{ $s(u,u)$ \}

Basic notion for unification in backward chaining, dependency between rules, decomposition of a rule into equivalent rules, …
Piece-Unification (1)

\[ R = q(x,y) \land q(y,x) \rightarrow p(x,z) \land p(z,x) \land p(z,z) \land r(z) \]

\[ Q = p(u,v) \land p(v,w) \land p(w,u) \land s(u,u) \]

A piece-unifier has to map at least one piece of the query to the rule head.

New query
A piece-unifier has to map at least one piece of the query to the rule head

→ failure
Piece-Unification (3)

Initially [Salvat M... ICCS 1996] on conceptual graphs

**Piece-unifier** of a query $Q$ with a rule $R$:

- a substitution $s$ of $\text{frontier}(R)$ by $\text{frontier}(R) + \text{constants}(Q + \text{head}(R))$

- a homomorphism $h$ from $Q' \subseteq Q$ to $s(\text{head}(R))$
  s.t. $Q'$ is a set of pieces according to $s$ and $h$

- [Salvat M... 1996]

  $F, \mathcal{R} |= Q$ iff there is a sequence of piece-unifications that empties $Q$
  (considering facts as rules with an empty body)

- [Baget+ IJCAI 2009] for *fus* existential rules

  $F, \mathcal{R} |= Q$ iff one of the piece-based rewritings of $Q$ maps to $F$
Outline

- Existential rules: a logic- and graph-based framework

- Decidability and algorithmic issues
  - Focus on:
    - tree-shaped saturation in forward chaining
    - piece-based unification in backward chaining

- A (graph) tool for combining decidable classes of rules
Union of Decidable Sets of Rules

- Next question:
  
is the union of two decidable sets of rules still decidable?

Practically:
- can we safely merge several decidable ontologies?
- can we build a decidable hybrid language from two languages whose semantics can be expressed by decidable subsets of rules?

- Bad news:
  
  Almost all classes are pairwise incompatible

- Next question:
  
  which conditions on the interactions between rules ensure compatibility?
A tool: the Graph of Rule Dependencies


R2 depends on R1 if applying R1 may trigger a new application of R2

i.e., there exists a fact F s.t. R1 is applicable to F but R2 is not and there is an application of R1 to F leading to F’ s.t. R2 is applicable to F’

Effective computation of dependencies with a piece-unification test:

R2 depends on R1 iff there is a « piece-unifier » of body(R2) with head(R1)
Combining Decidables Classes with the **Graph of Rule Dependencies**

- **Rules**
- R1 $\rightarrow$ R2: R1 « may trigger » R2 (R2 depends on R1)
Combining Decidable Classes with the **Graph of Rule Dependencies**

If $\text{GRD}(\mathcal{R})$ is **without circuit** then $\mathcal{R}$ is both $fes$ (thus $bts$) and $fus$

- $fes = \text{finite fact saturation}$
- $fus = \text{finite query rewriting}$
- $bts = (\text{possibly infinite}) \text{ tree-shaped saturation}$
Combining Decidable Classes with the Graph of Rule Dependencies

If all strongly connected components of $\text{GRD}(\mathcal{R})$ are \textit{fes} then $\mathcal{R}$ is \textit{fes} [Baget 2004]

The same holds for \textit{fus} (\textit{but not for bts})
Combining Decidable Classes with the Graph of Rule Dependencies

If all strongly connected components of GRD(\(\mathcal{R}\)) are fes then \(\mathcal{R}\) is fes [Baget 2004]

The same holds for fus (but not for bts)
Combining Decidable Classes with the Graph of Rule Dependencies

Let $\mathcal{R}_1 \parallel \mathcal{R}_2$ be a partition of $\mathcal{R}$ s.t. no rule of $\mathcal{R}_1$ depends on a rule of $\mathcal{R}_2$

- If $\mathcal{R}_1$ is $\text{fes}$ and $\mathcal{R}_2$ is $\text{bts}$, then $\mathcal{R}$ is $\text{bts}$
- If $\mathcal{R}_1$ is $\text{bts}$ and $\mathcal{R}_2$ is $\text{fus}$, then $\mathcal{R}$ is decidable

Decidable

Diagram:
- $\text{fes}$ class
- $\text{bts}$ class
- $\text{fus}$ class
- $\text{ab (fus)}$ class
- $\text{dr (fus)}$ class
- $\text{Datalog (fes)}$ class
- $\text{fg (bts)}$ class

Relations shown in the diagram:
- $\text{fes}$ to $\text{fes}$
- $\text{bts}$ to $\text{bts}$
- $\text{fus}$ to $\text{fus}$
- $\text{ab (fus)}$ to $\text{fus}$
- $\text{dr (fus)}$ to $\text{fus}$
- $\text{Datalog (fes)}$ to $\text{fes}$
- $\text{fg (bts)}$ to $\text{bts}$
Conclusion

- An emerging rule-based framework for OBDA
  - simple
  - expressive
  - flexible

- Logic-based and Graph-based

- Currently:
  - A quite clear picture of decidable classes and their complexities
  - First implementations – often for very specific subclasses

- Next challenge: scalability