



HAL
open science

Existential Rules: A Graph-Based View

Marie-Laure Mugnier

► **To cite this version:**

Marie-Laure Mugnier. Existential Rules: A Graph-Based View. Pablo Barceló; Reinhard Pichler. Datalog 2.0, Sep 2012, Vienne, Austria. Springer, 2nd International Workshop on Datalog 2.0, LNCS (7494), pp.21-26, 2012, 10.1007/978-3-642-32925-8_3 . lirmm-00763474

HAL Id: lirmm-00763474

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-00763474v1>

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

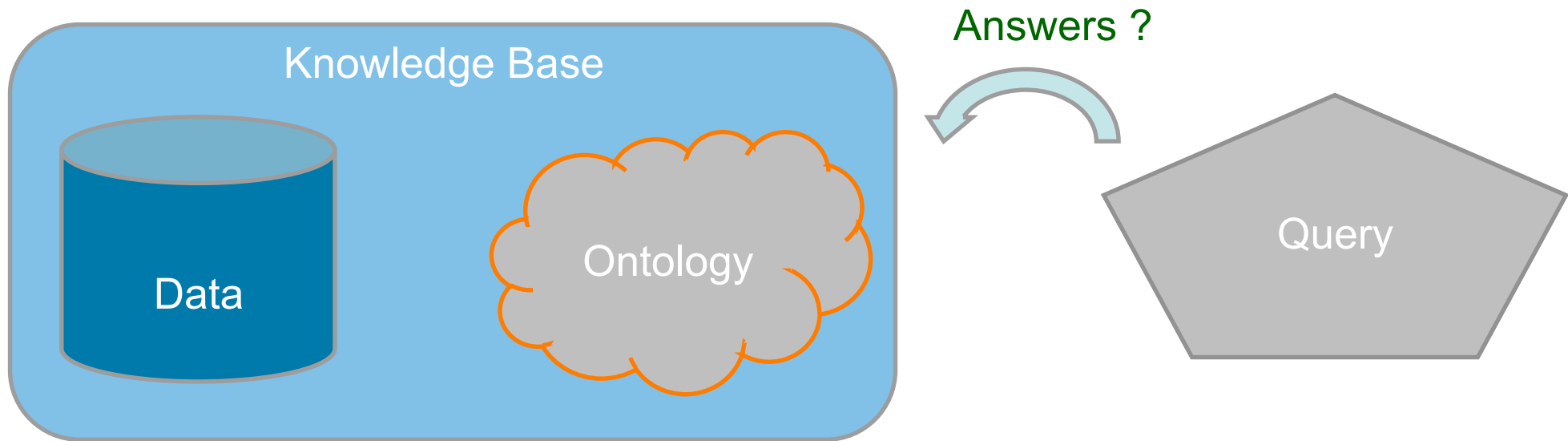
Marie-Laure Mugnier

University of Montpellier



Datalog 2.0, Vienna, 2012

Ontology-based Data Access (OBDA)



Adding an ontological layer:

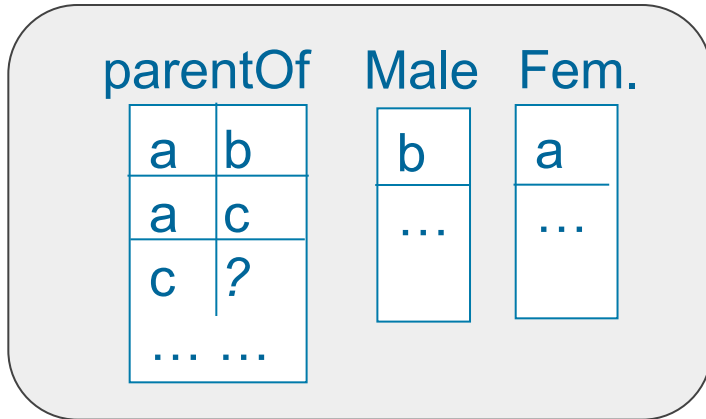
- to **abstract** from a specific database schema
- to provide a **unified view** of multiple sources
- to infer new facts, thus allowing for **data incompleteness**

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

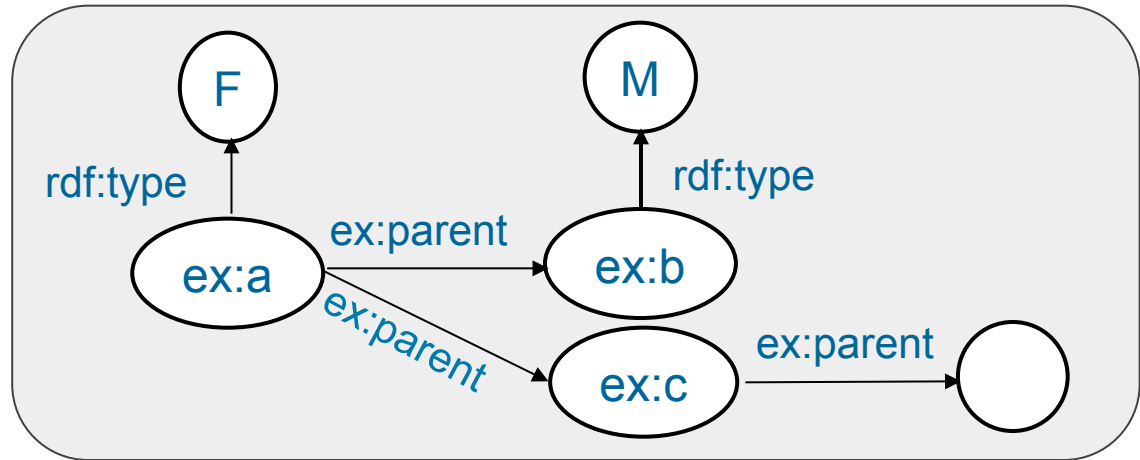
Data / Facts

Relational Database



RDF (Semantic Web)

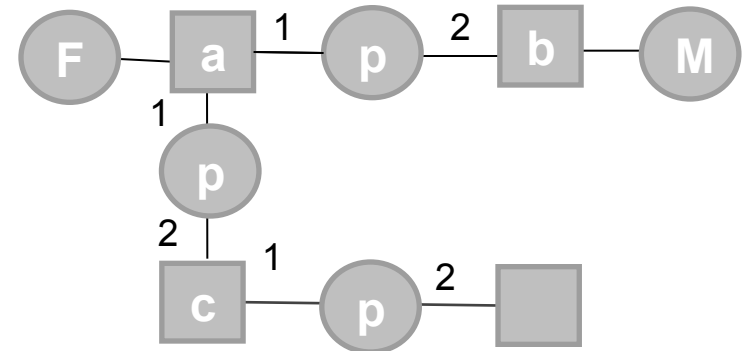
Etc.



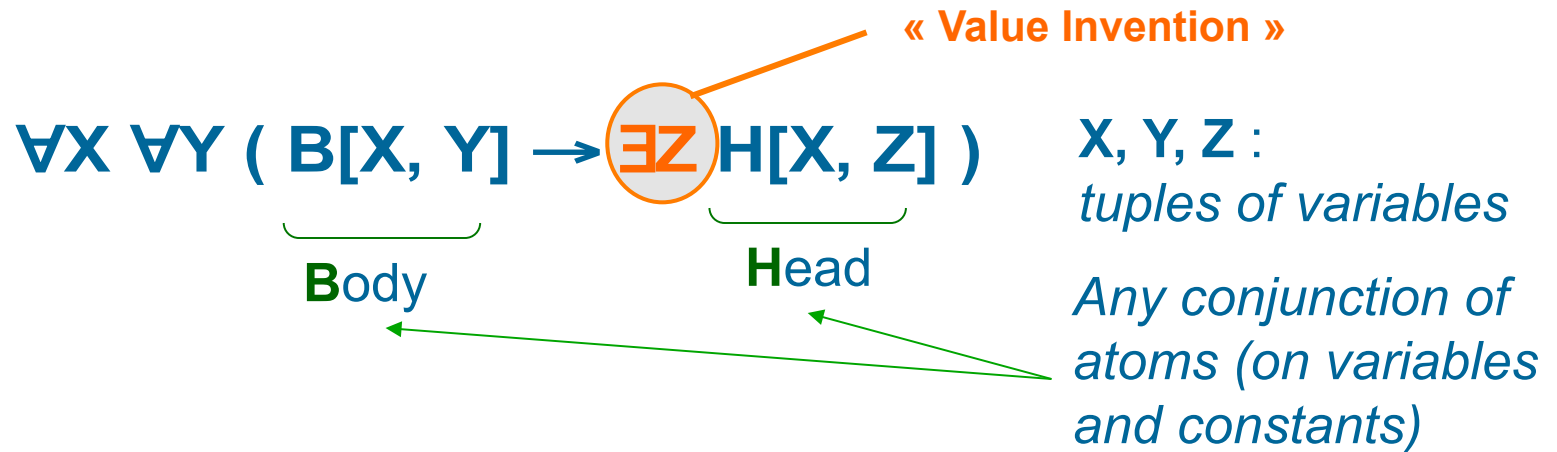
Abstraction in first-order logic

$$\exists x(\text{parentOf}(a,b) \wedge \text{parentOf}(a,c) \wedge \text{parentOf}(c,x) \wedge F(a) \wedge M(b))$$

Or in graphs / hypergraphs



Ontology: Existential Rules



$\forall x \forall y (\text{siblingOf}(x,y) \rightarrow \exists z (\text{parentOf}(z,x) \wedge \text{parentOf}(z,y)))$

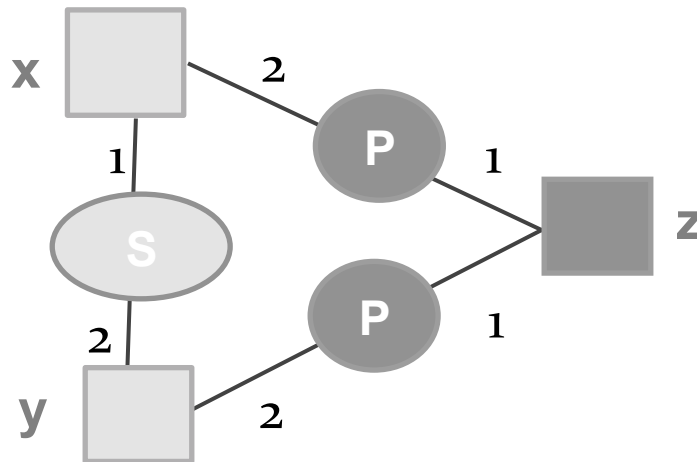
Simplified form: $\text{siblingOf}(x,y) \rightarrow \text{parentOf}(z,x) \wedge \text{parentOf}(z,y)$

- 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})

Ontology: Existential Rules

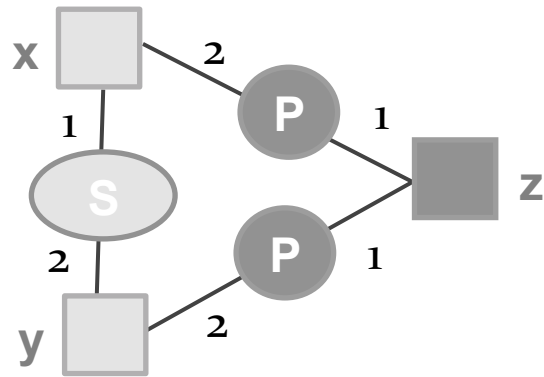
$$\forall X \forall Y (\underbrace{B[X, Y]}_{\text{graph}} \rightarrow \exists Z \underbrace{H[X, Z]}_{\text{graph}})$$

$$\forall x \forall y (\text{siblingOf}(x,y) \rightarrow \exists z (\text{parentOf}(z,x) \wedge \text{parentOf}(z,y)))$$

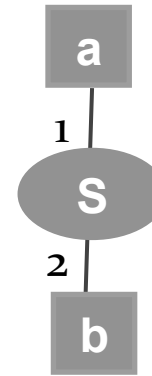


Value Invention

$$R = \forall x \forall y (\text{siblingOf}(x,y) \rightarrow \exists z (\text{parentOf}(z,x) \wedge \text{parentOf}(z,y)))$$

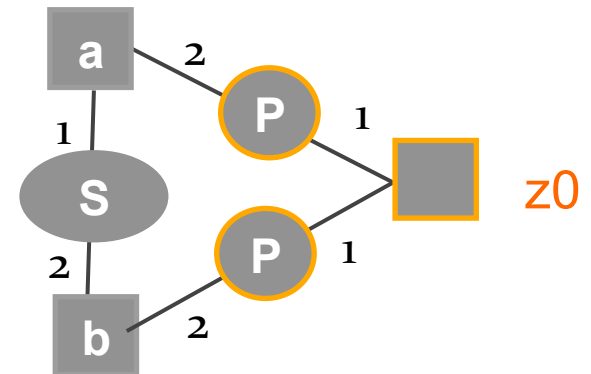
$$F = \text{siblingOf}(a,b)$$


$$h: \text{body} \rightarrow F$$

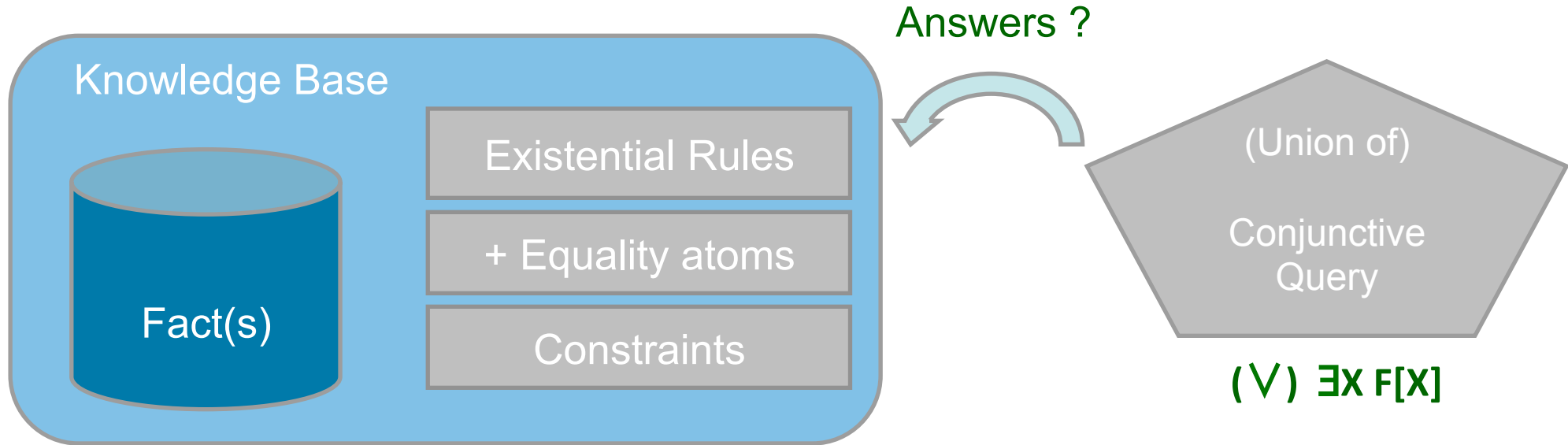
$$h = \{(x,a), (y,b)\}$$


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

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

$$F' = \exists z_0 (\text{siblingOf}(a,b) \wedge \text{parentOf}(z_0,a) \wedge \text{parentOf}(z_0,b))$$


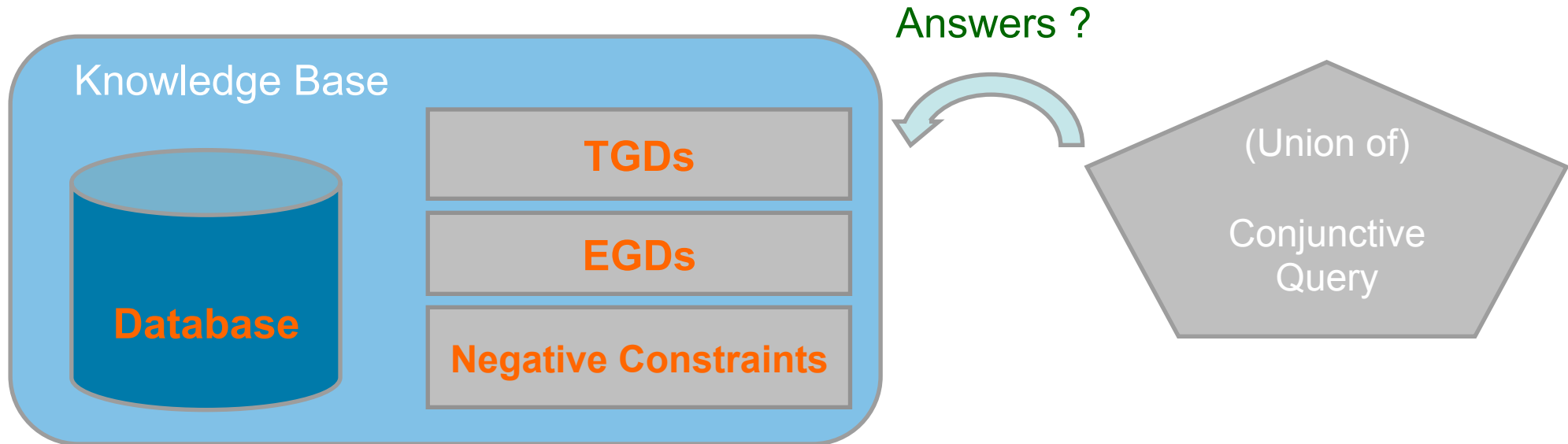
Logical /Graphical Framework



Negative constraint: $\neg (\exists X B[X])$ or $\forall X (B[X] \rightarrow \perp)$
« 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 +/-



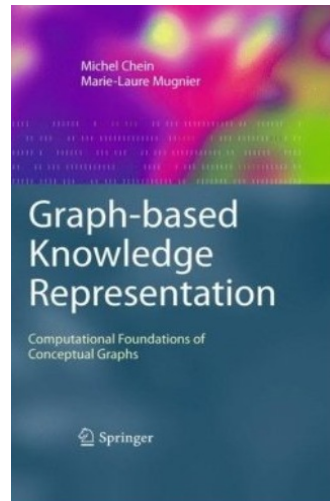
[Cali Gottlob Lukasiewicz PODS 2009]

Tuple Generating Dependency = (pure) existential rule

Equality Generating Dependency: $\forall X (B[X] \rightarrow x = e)$

The Conceptual Graph Origins

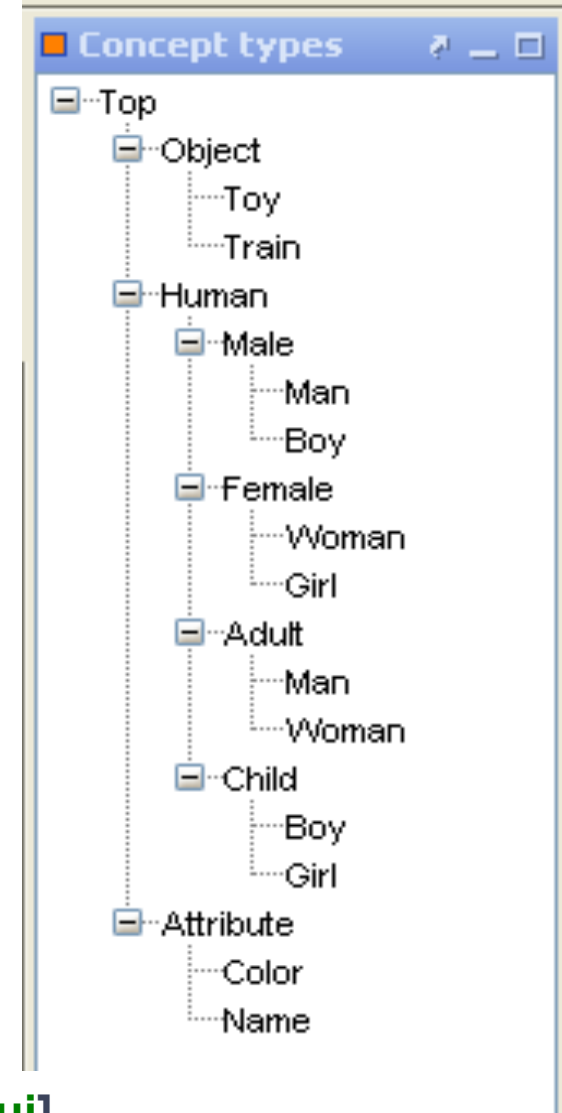
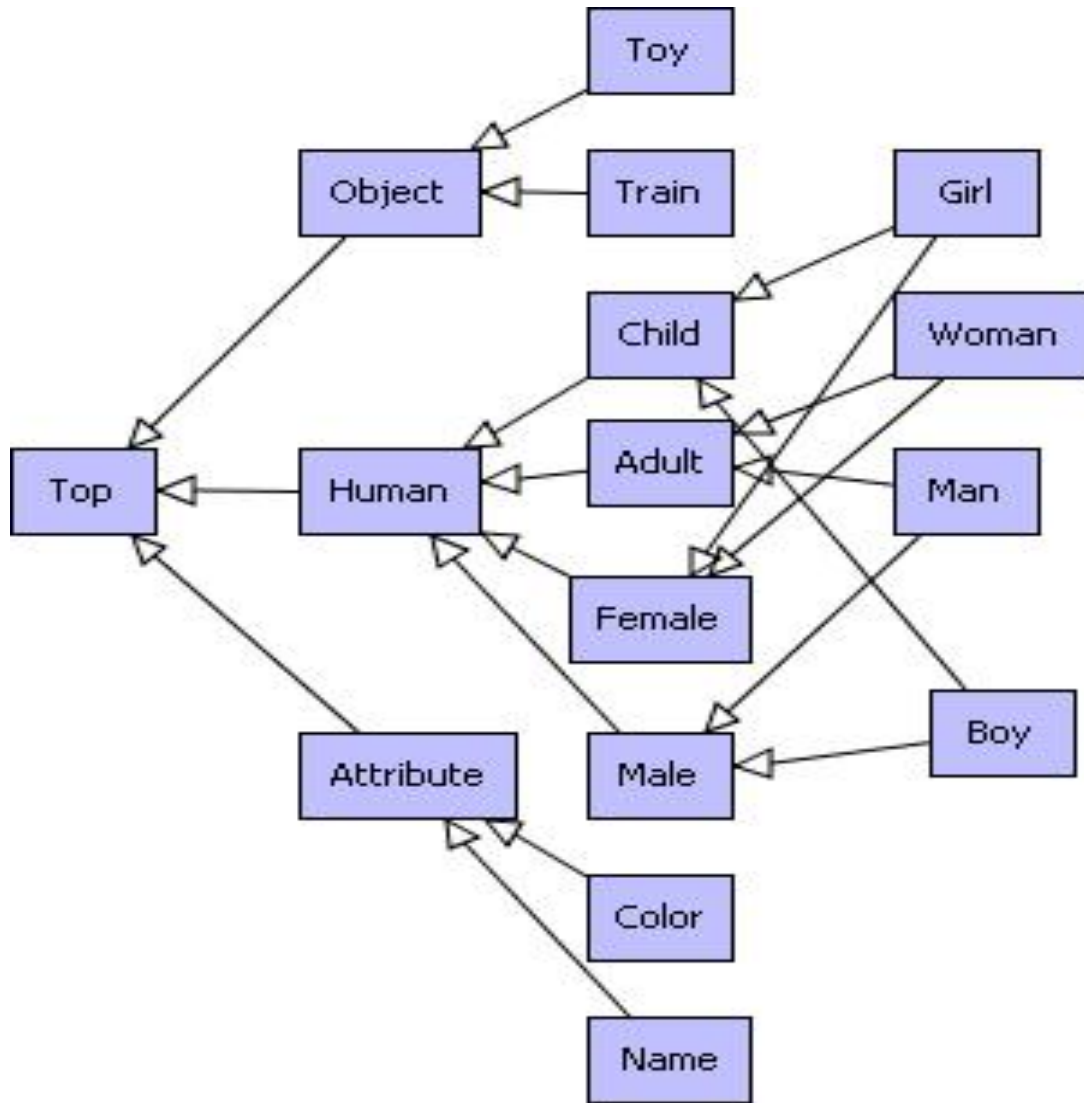
- Conceptual graphs introduced in [Sowa 76] [Sowa 84]
- Specific research line by Montpellier's group since 1992
 - « Graph-based » knowledge representation and reasoning



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

Conceptual Graph Vocabulary:

1. partially (pre-)ordered set of **concepts**

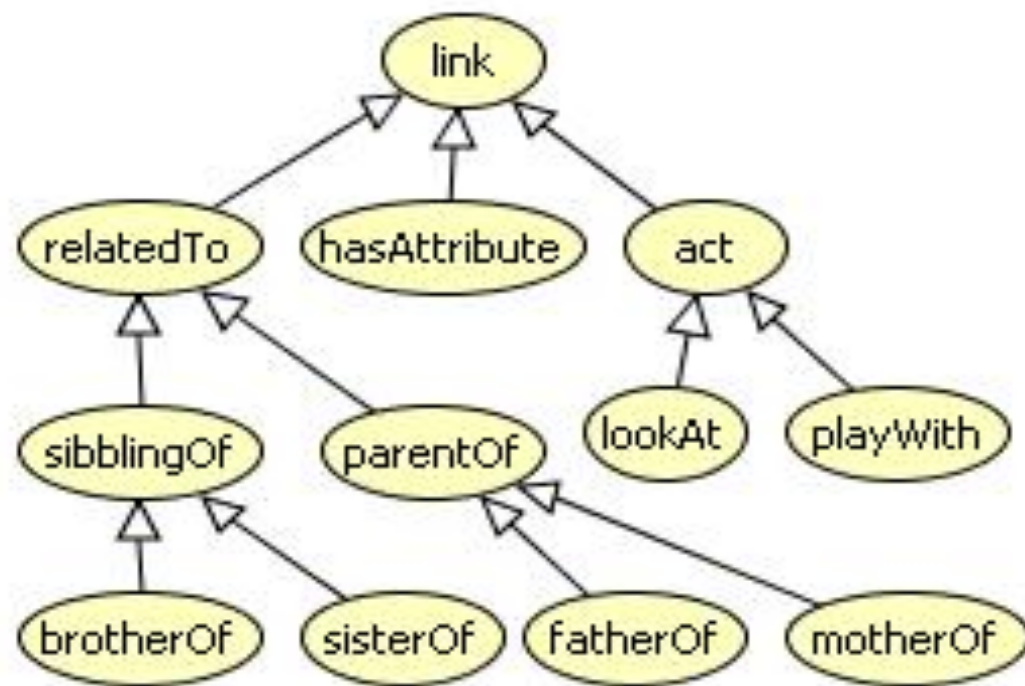


[screenshots from CoGui, <http://www.lirmm.fr/cogui>]



Conceptual Graph Vocabulary:

- partially (pre-)ordered set of **relations** with their **signature**
[any relation arity allowed]



link(Top ,Top)

- relatedTo(Human ,Human)
 - siblingOf(Human ,Human)
 - ... brotherOf(Male ,Human)
 - ... sisterOf(Female ,Human)
 - parentOf(Adult ,Human)
 - ... fatherOf(Man ,Human)
 - ... motherOf(Woman ,Human)
- act(Human ,Top)
 - ... lookAt(Human ,Top)
 - ... playWith(Human ,Object)
 - ... hasAttribute(Top ,Attribute)

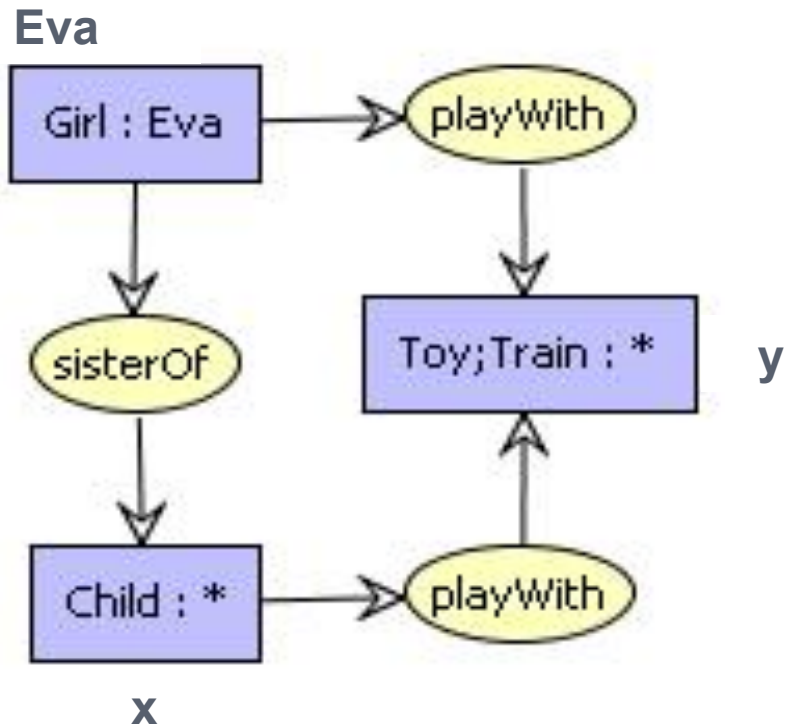
Logical translation (Φ) of the vocabulary: very simple rules

$$p < q \quad \forall x_1 \dots x_k (p(x_1 \dots x_k) \rightarrow q(x_1 \dots x_k))$$

Signature of r $\forall x_1 \dots x_k (p(x_1 \dots x_k) \rightarrow t_{i_1}(x_1) \dots t_{i_k}(x_k))$



Basic Conceptual Graph



[total order on the edges incident to a relation node]

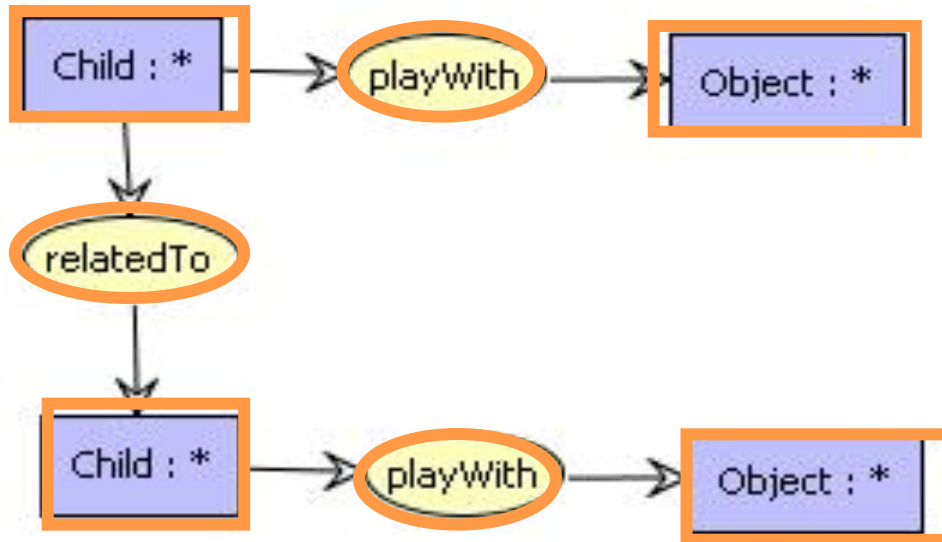
Logical translation (Φ): existentially closed conjunction of atoms

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

Allows to represent **facts** and **conjunctive queries**

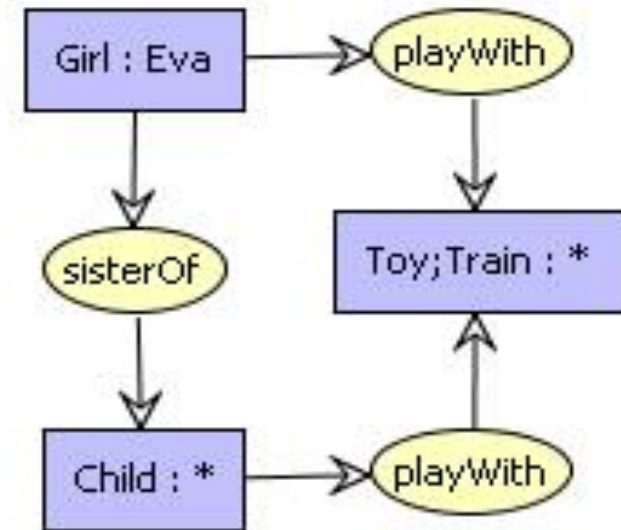


Homomorphism (with concept/relation preorders integrated)



Query Q

Fact F



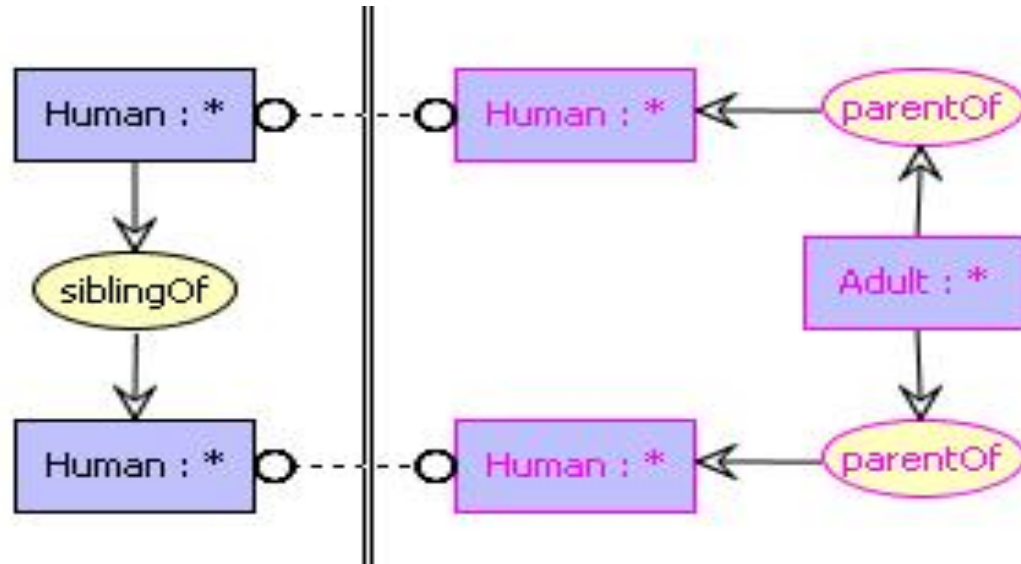
Logical **soundness** [Sowa 84] and **completeness** [Chein M... 92]:
there is a homomorphism from Q to F iff
 $\Phi(Q)$ is entailed by $\Phi(F)$ and $\Phi(\text{vocabulary})$

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



$$\forall x \forall y (\text{Human}(x) \wedge \text{Human}(y) \wedge \text{siblingOf}(x,y) \\ \rightarrow \exists z (\text{Adult}(z) \wedge \text{parentOf}(z,x) \wedge \text{parentOf}(z,y)))$$

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

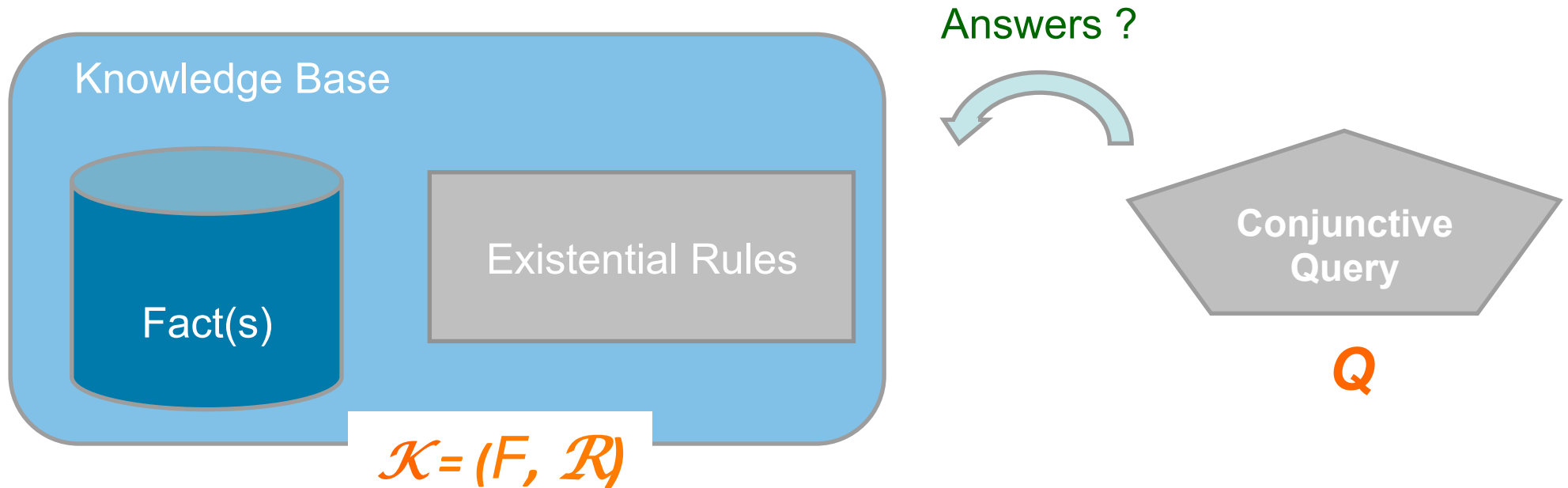
The existential rule framework can be seen as a fragment of CGs with a flat vocabulary



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

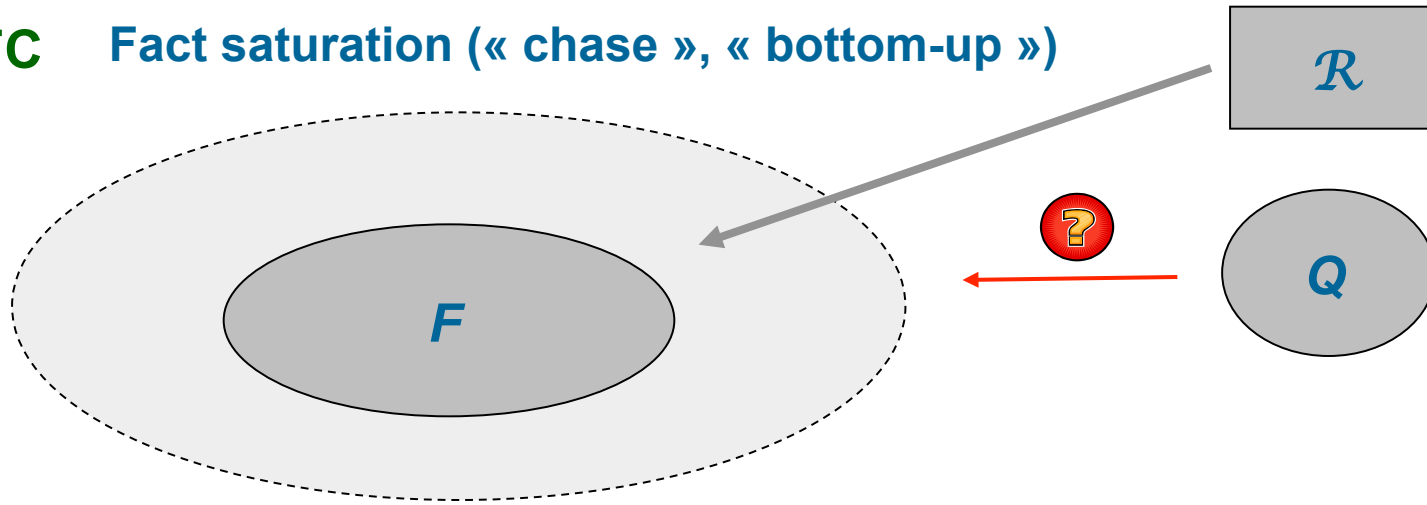


■ Conjunctive Query Entailment

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

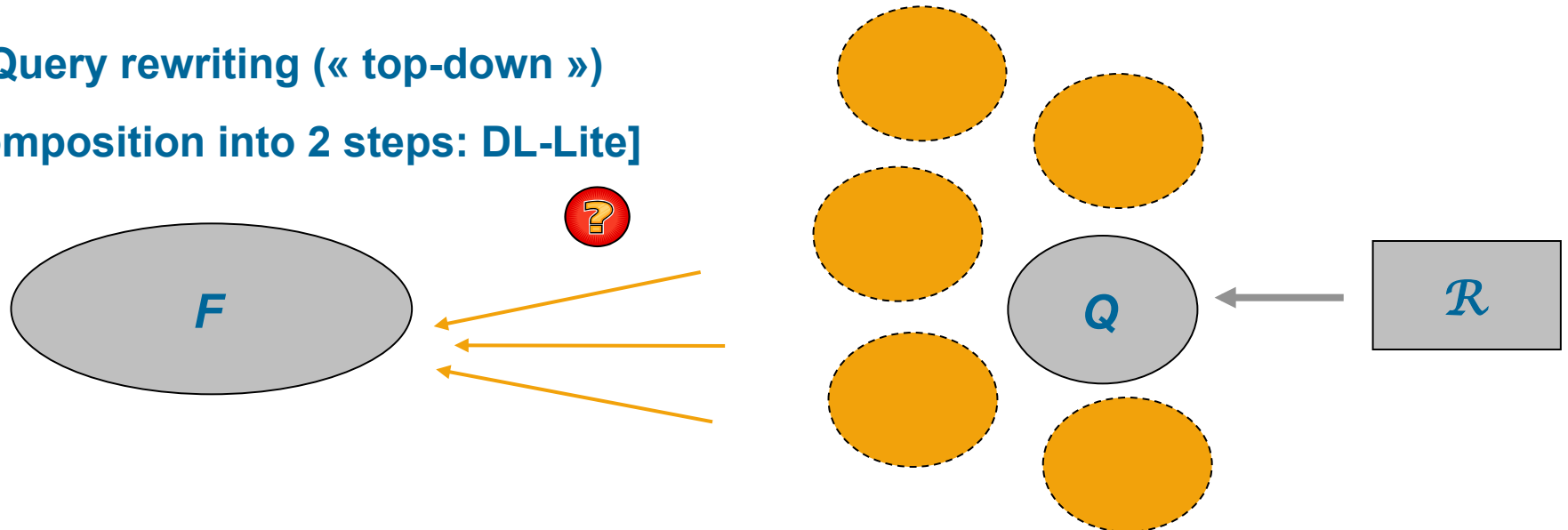
Forward vs Backward Chaining

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



BC Query rewriting (« top-down »)

[Decomposition into 2 steps: DL-Lite]



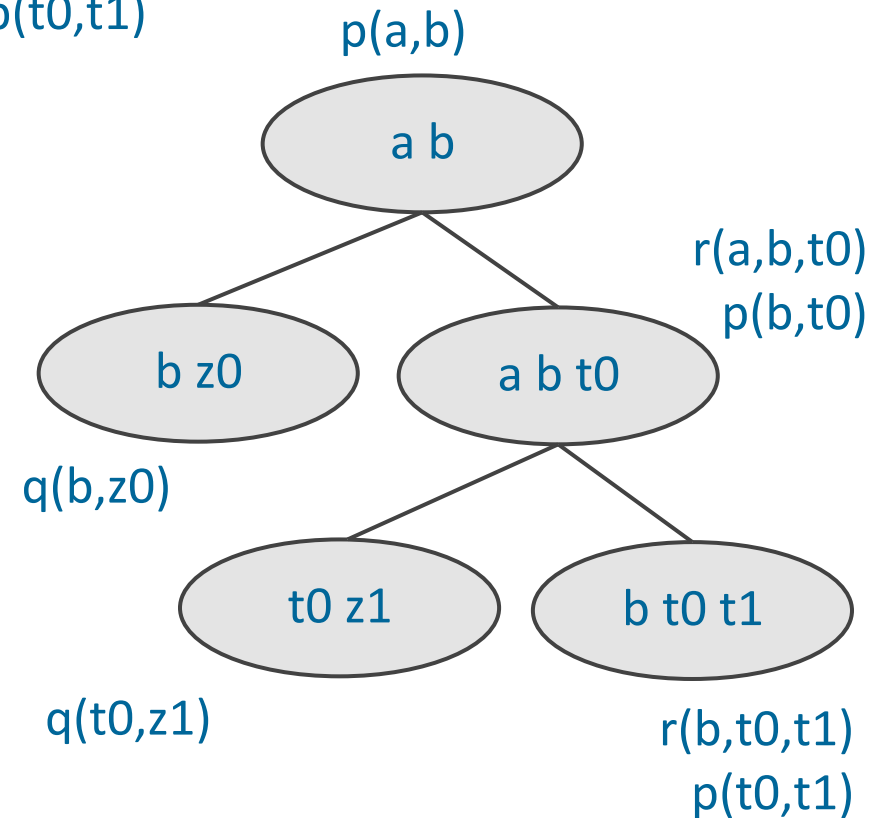
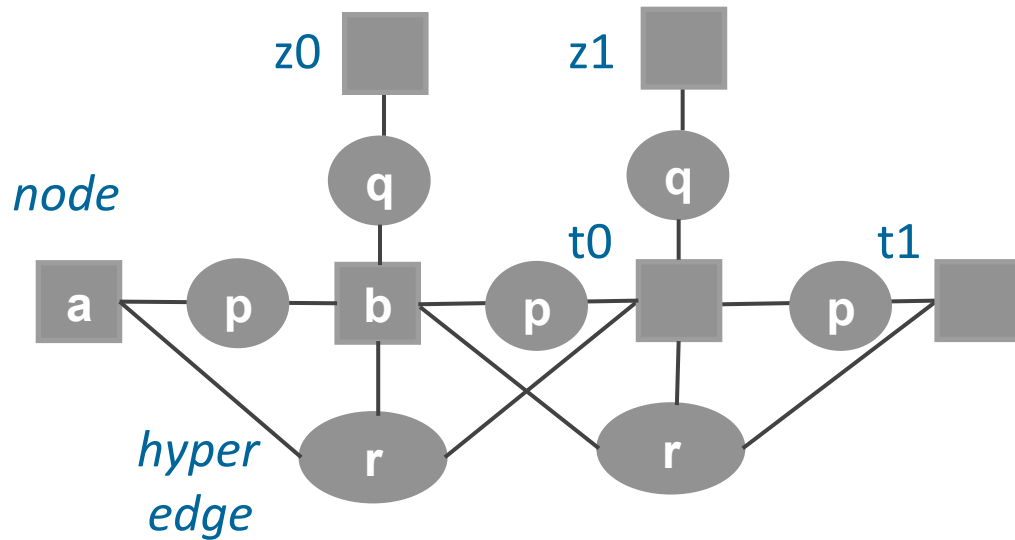
Decidability Issues

- 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 », *fes*)
 - Query rewriting by Backward Chaining halts (« finite unification set », *fus*)
 - Saturation by Forward Chaining may not halt *but* the generated facts have a tree-like structure (« bounded treewidth set », *bts*)

None of these properties is *recognizable* [Baget+ KR 10] but they provide *generic* algorithms

Decomposition Tree / Treewidth

$p(a,b)$ $q(b,z_0)$ $r(a,b,t_0)$ $p(b,t_0)$ $q(t_0,z_1)$ $r(b,t_0,t_1)$ $p(t_0,t_1)$



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

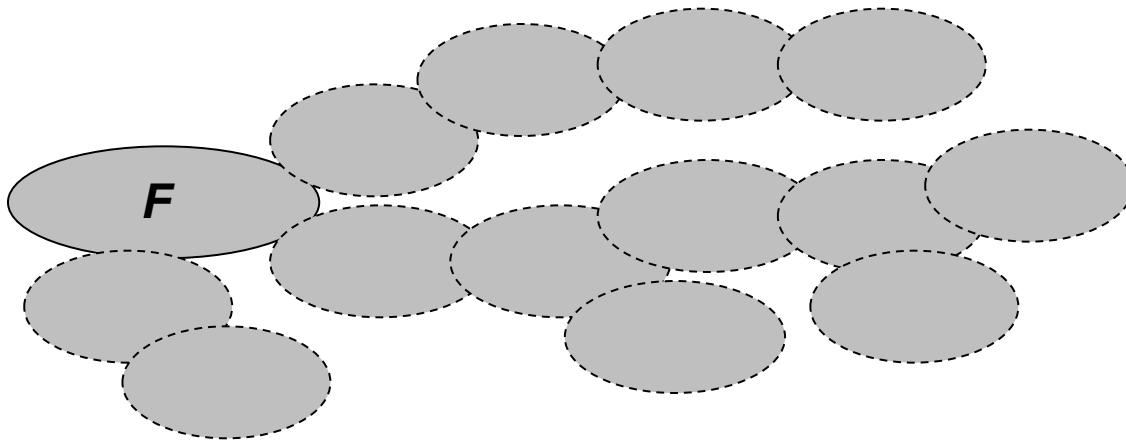
Width of a tree decomposition = *max* number of nodes in a bag (minus 1)

Treewidth of a graph = *min* width over all decomposition trees of this graph

Bounded Treewidth of the Derived Facts (*bts*)

Essentially [Cali Gottlob Kifer KR'08]

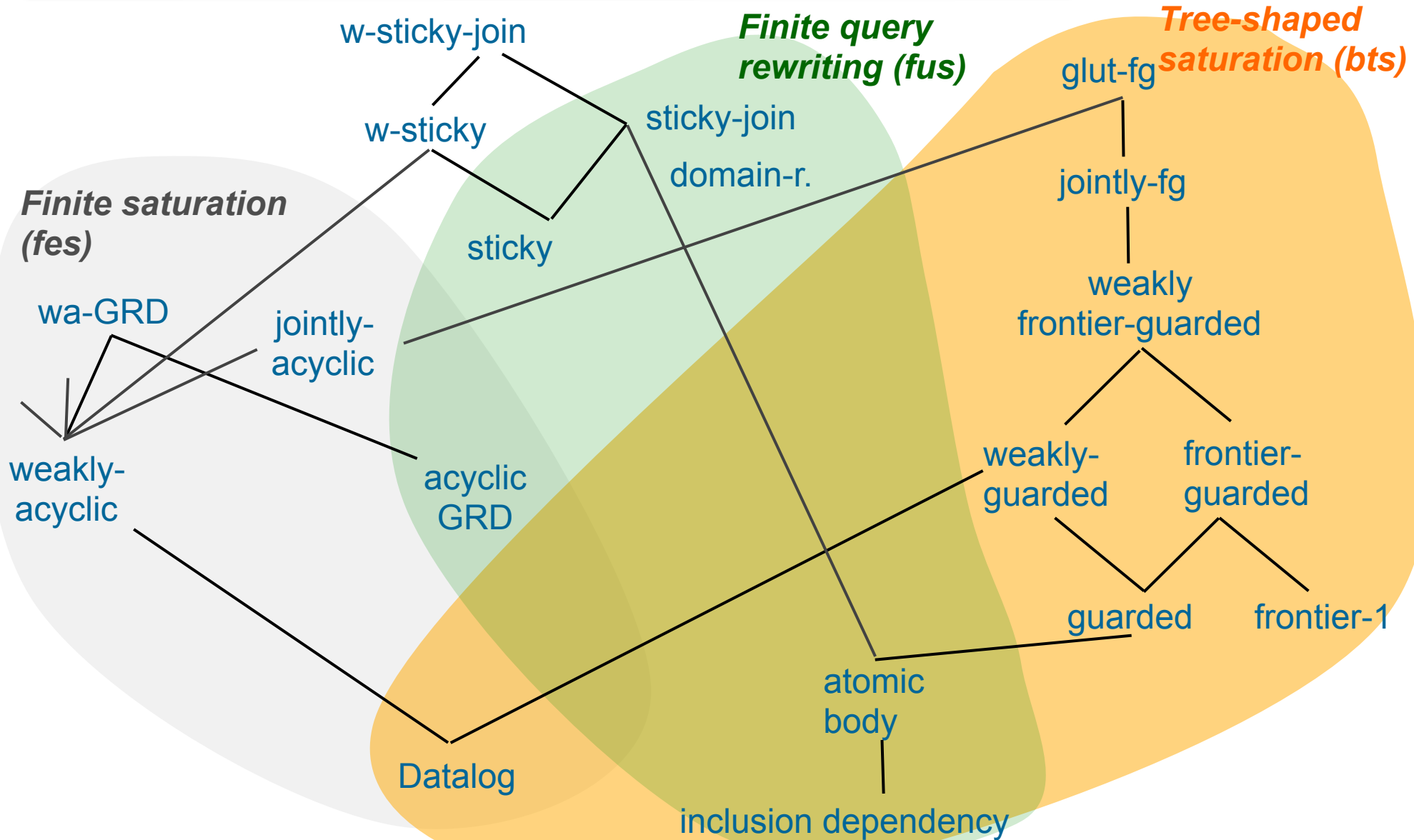
\mathcal{R} is *bts* if FC with \mathcal{R} generates facts with **bounded treewidth**
i.e., for any fact F , there is an integer b s.t.
any fact \mathcal{R} -derived from F has treewidth bounded by b



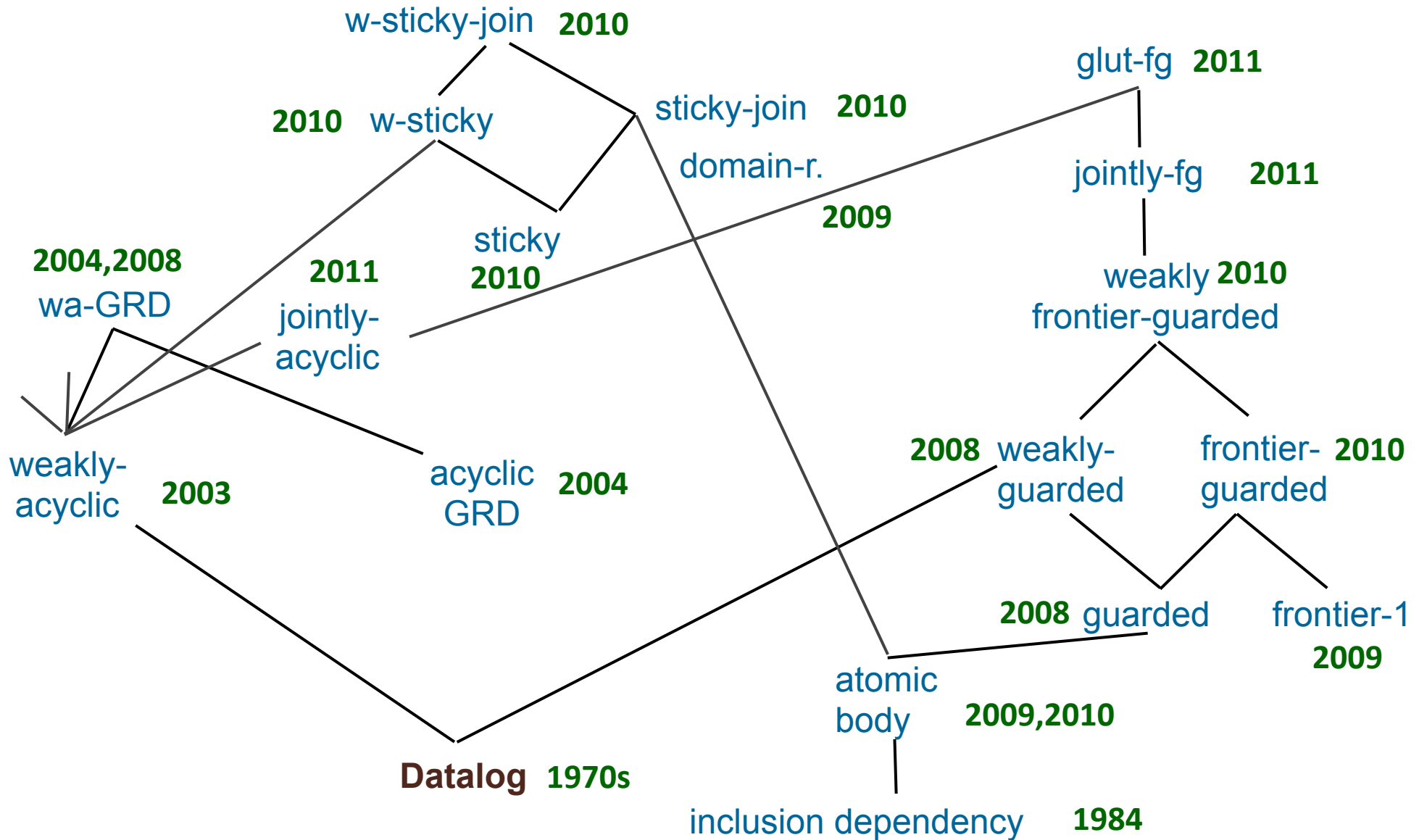
fes (*finite saturation*) is included in *bts*
(bound given by the number of terms in the finite « saturated fact »)

The decidability proof does not provide a halting algorithm
(relies on the bounded treewidth model property [Courcelle 90])

(Partial) Inclusion Map of Decidable Classes



(Partial) Inclusion Map of Decidable Classes



Some Recognizable bts (and not fes) Classes of Rules

Frontier: variables shared by the body and the head

Guard only *affected* variables from the *frontier*

[Baget+ KR' 10]

Guard only the *frontier*

[Baget+ KR' 10]

$r(x,y) \wedge r(y,z) \rightarrow r(y,u) \wedge r(z,u)$

The *frontier* has size 1

[Baget+ IJCAI' 09]

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

[Cali+ KR' 08]

datalog



$r(x,y) \wedge r(y,z) \wedge r(x,z) \rightarrow r(z,u)$

$r(x,y) \wedge r(y,z) \wedge s(x,y,z) \rightarrow r(y,u) \wedge r(z,u)$

An atom in the body *guards* all the body variables

[Cali+ KR' 08]

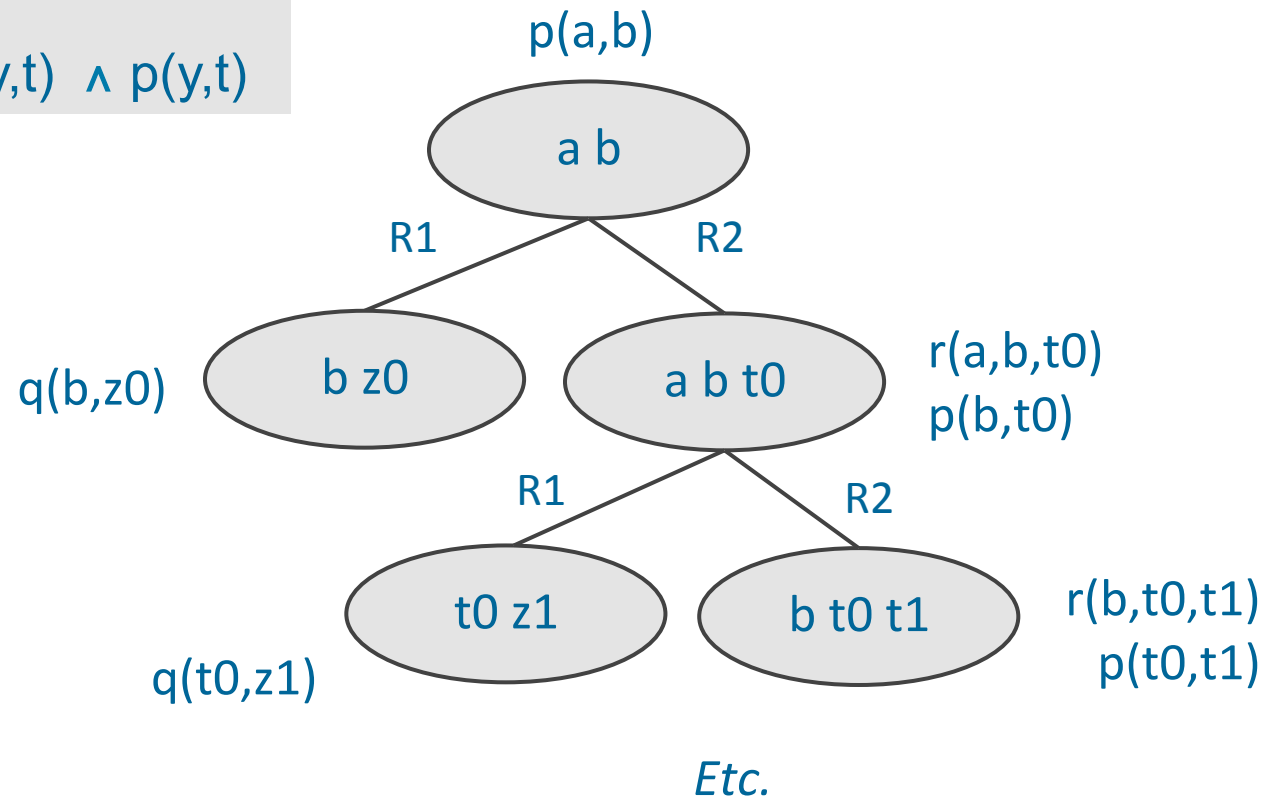
These classes are moreover « **greedy bts** » => a halting algorithm [Baget+ IJCAI' 11]

Greedy *bts*

$$R1 = p(x,y) \rightarrow p(y,z)$$

$$R2 = p(x,y) \wedge q(x,z) \rightarrow r(x,y,t) \wedge p(y,t)$$

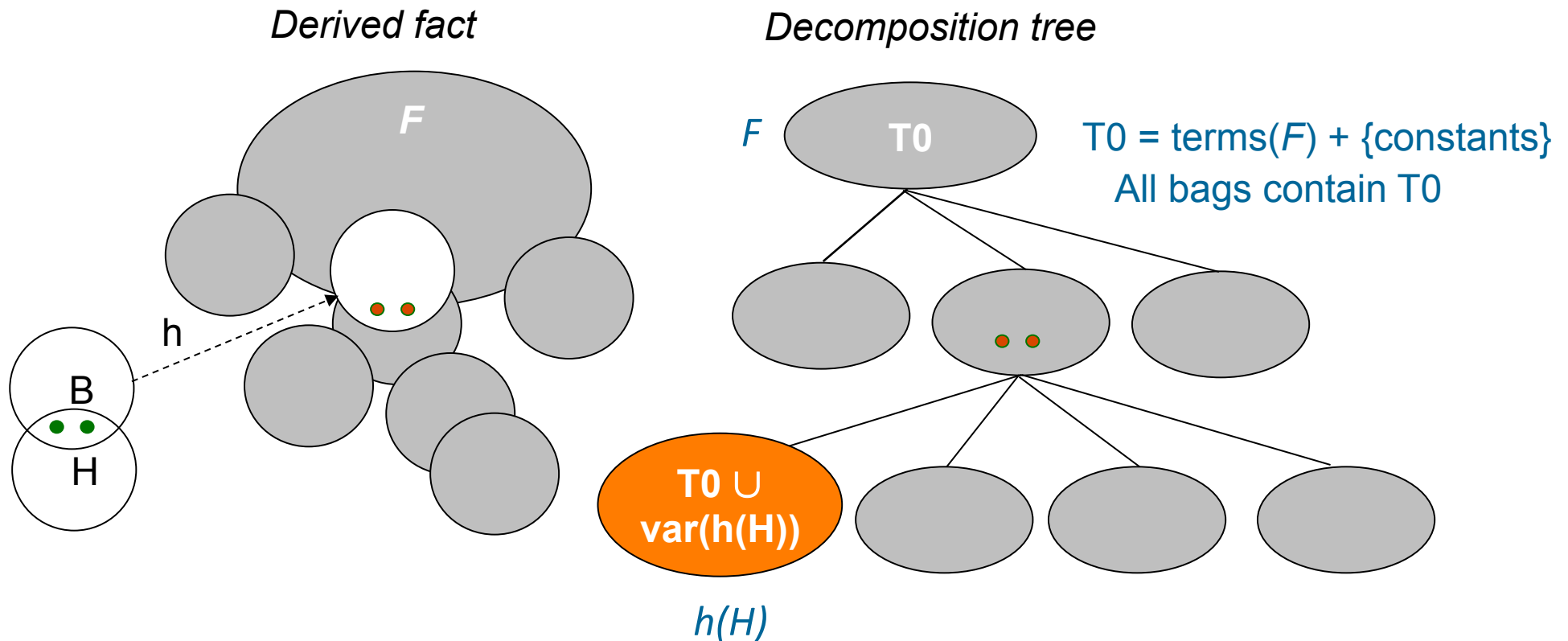
$$F = p(a,b)$$



Greedy construction of a **decomposition tree** of the derived fact
with bounded width

The « Greedy bts » Property [Baget+ IJCAI' 11]

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)

Build a **finite** decomposition tree that encodes a potentially infinite fact

1. **Bag pattern** = { *homomorphisms from part of a rule body to « current fact » that use some terms of the bag* }

→ A rule is applicable to the current fact *iff* a bag pattern contains its body

→ 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 → finite « full blocked tree » T^*

Main Ideas of the Algorithm for *gbts* (2)

Query this finite decomposition tree

[Baget+ IJCAI 2011] Q seen as a rule « $Q \rightarrow match$ »

Q is entailed iff it occurs in a bag pattern

i.e. Q maps by homomorphism to $atoms(T^*)$

[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 **-homomorphism* to T^*

i.e. Q maps by homomorphism to a bounded « development » of T^*

Backward Chaining: Unification Step

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

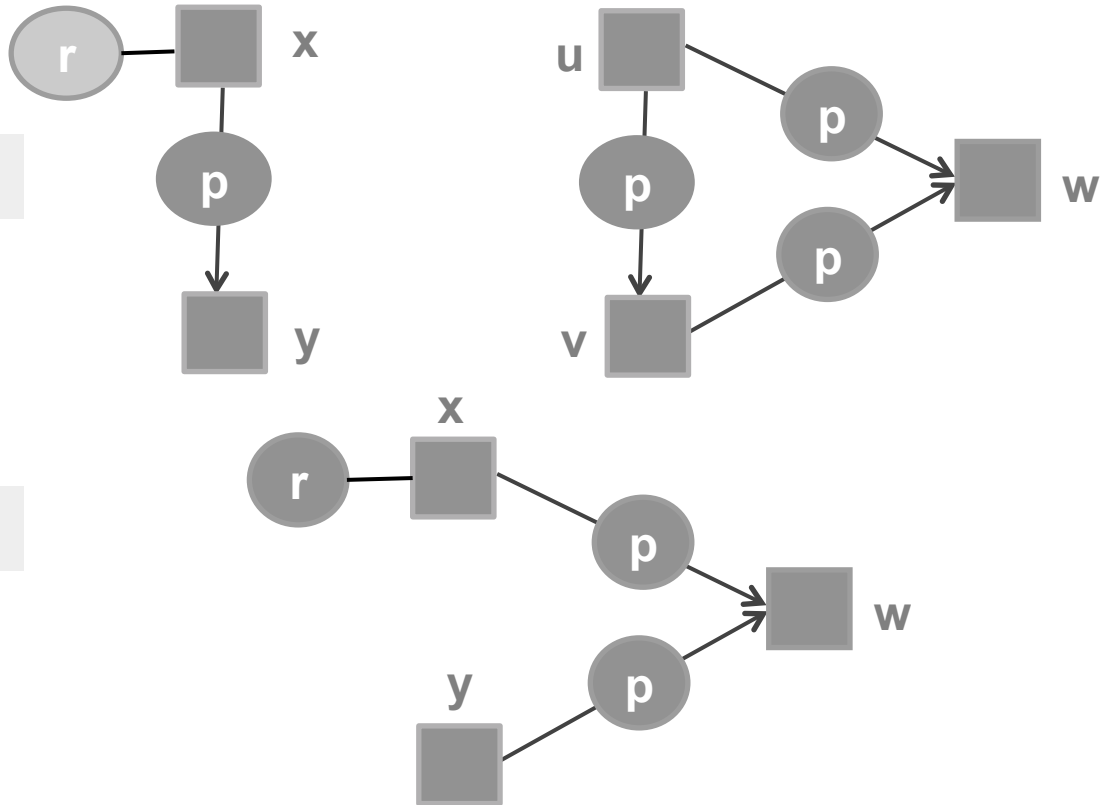
$$Q = p(u,v) \wedge p(u,w) \wedge p(v,w)$$

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

$$Q1 = r(x) \wedge p(x,w) \wedge p(y,w)$$

Soundness lost!

Indeed let $F = Q1$
 $\text{saturation}(F,R) \cong F$
Q does not map to F



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.

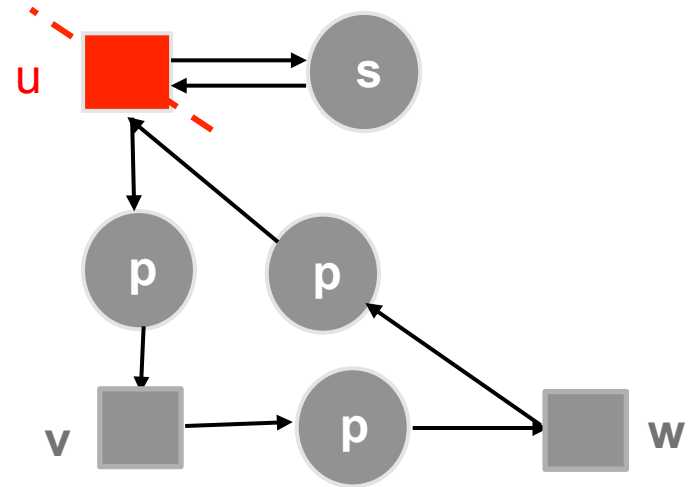
A piece = all atoms linked by a « path » of variables not belonging to T

$p(u,v)$ $p(v,w)$ $p(w,u)$ $s(u,u)$

$T = \{ u \}$

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

Piece 2 = $\{ s(u,u) \}$

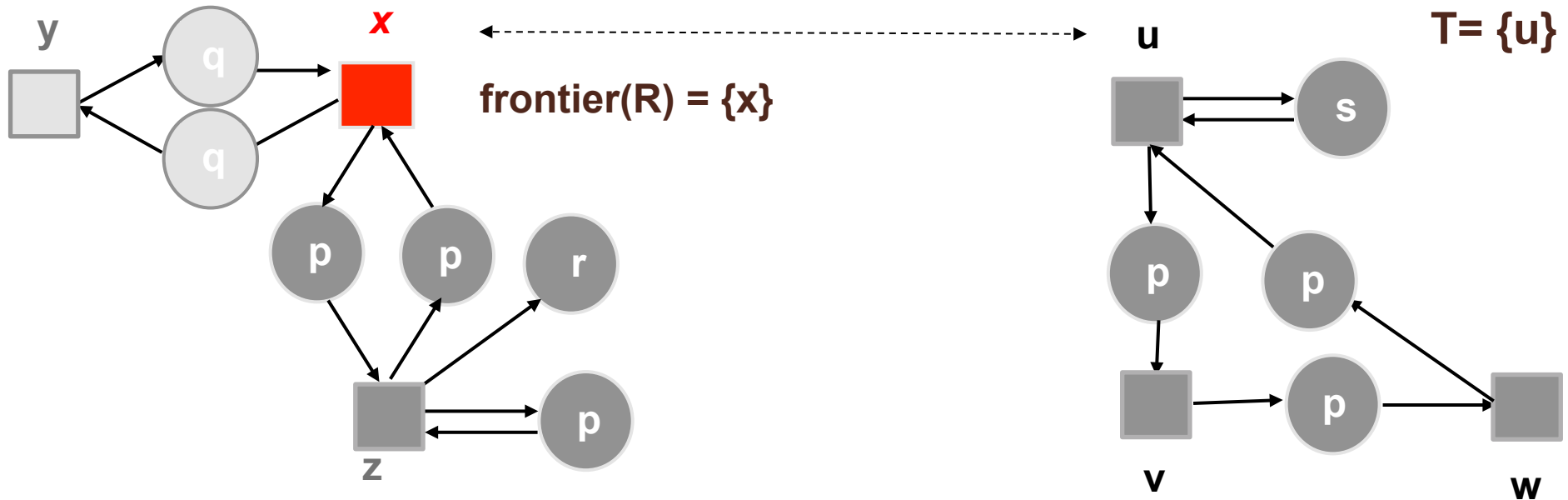


Basic notion for unification in backward chaining, dependency between rules, decomposition of a rule into equivalent rules, ...

Piece-Unification (1)

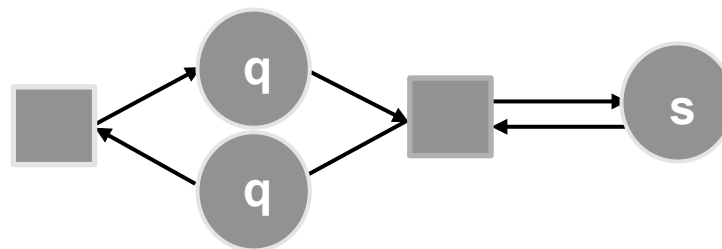
$$R = q(x,y) \wedge q(y,x) \rightarrow p(x,z) \wedge p(z,x) \wedge p(z,z) \wedge r(z)$$

$$Q = p(u,v) \wedge p(v,w) \wedge p(w,u) \wedge s(u,u)$$



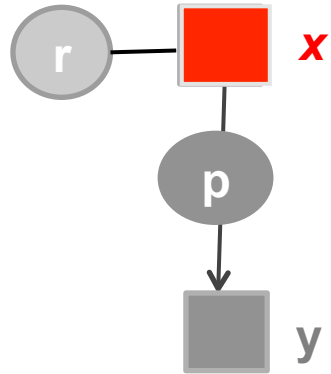
A piece-unifier has to map **at least one piece** of the query to the rule head

New query

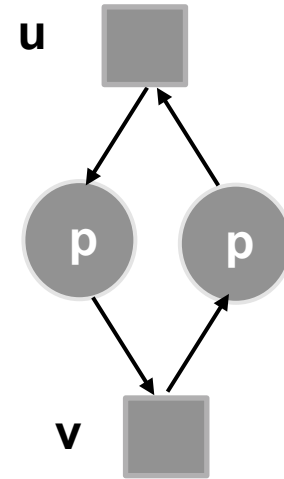


Piece-Unification (2)

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



$$Q = p(u,v) \wedge p(v,u)$$



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

\rightarrow 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]

$F, \mathcal{R} \models Q$ 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

$F, \mathcal{R} \models Q$ 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:

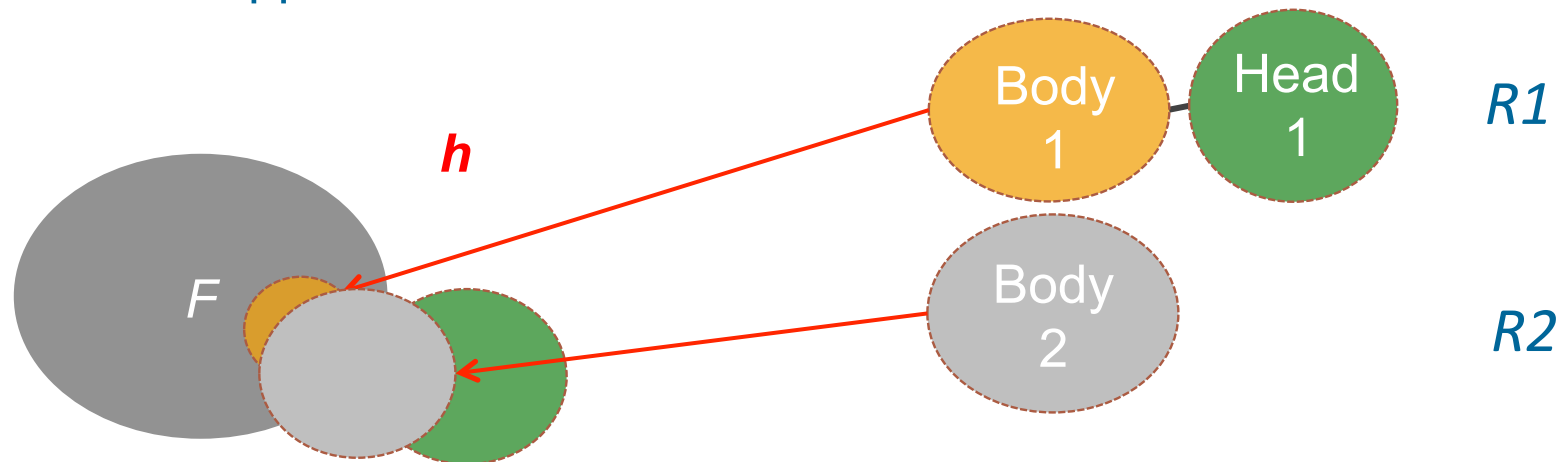
which conditions on the interactions between rules ensure compatibility ?

A tool : the Graph of Rule Dependencies

[Baget KR 2004, Baget+ IJCAI 2009, AIJ 2011]

$R2$ 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:

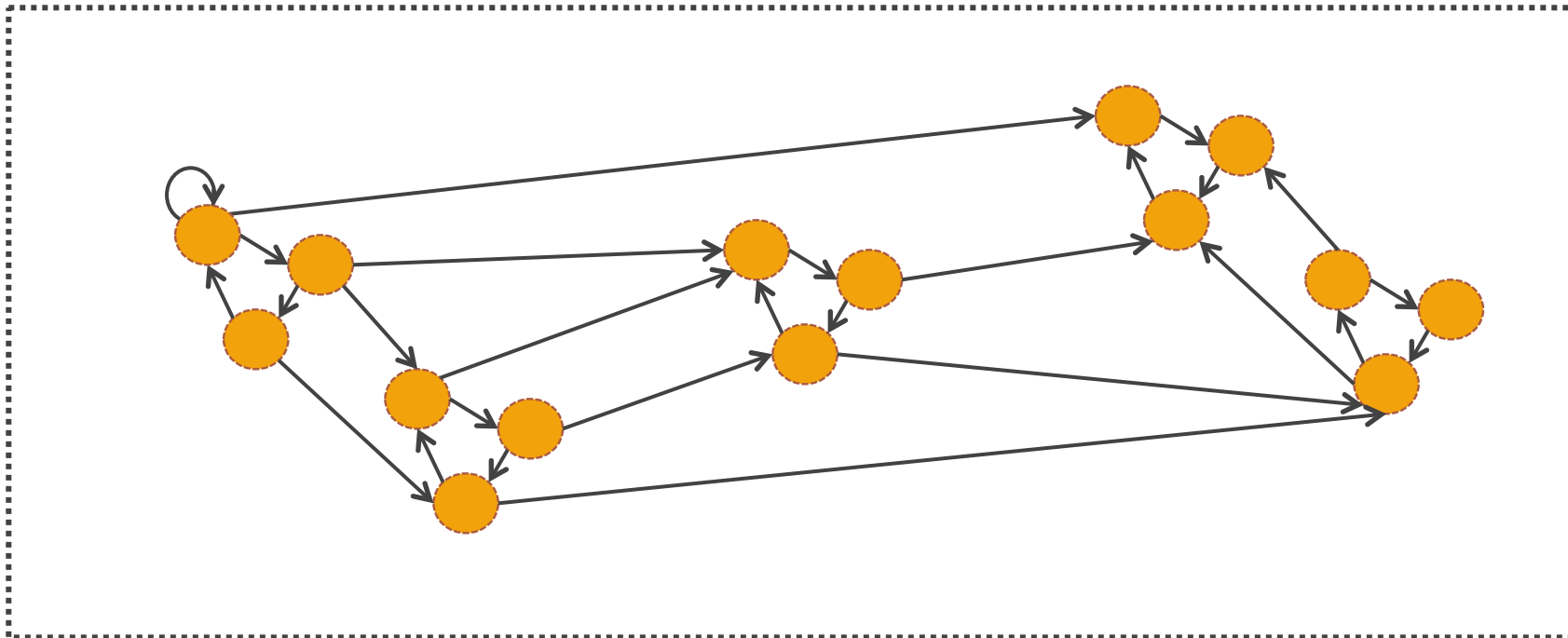
$R2$ depends on $R1$ iff there is a « **piece-unifier** » of $\text{body}(R2)$ with $\text{head}(R1)$

Combining Decidable Classes with the Graph of Rule Dependencies



Rules

$R1 \longrightarrow R2 : R1 \ll \text{may trigger} \gg R2$ ($R2$ depends on $R1$)



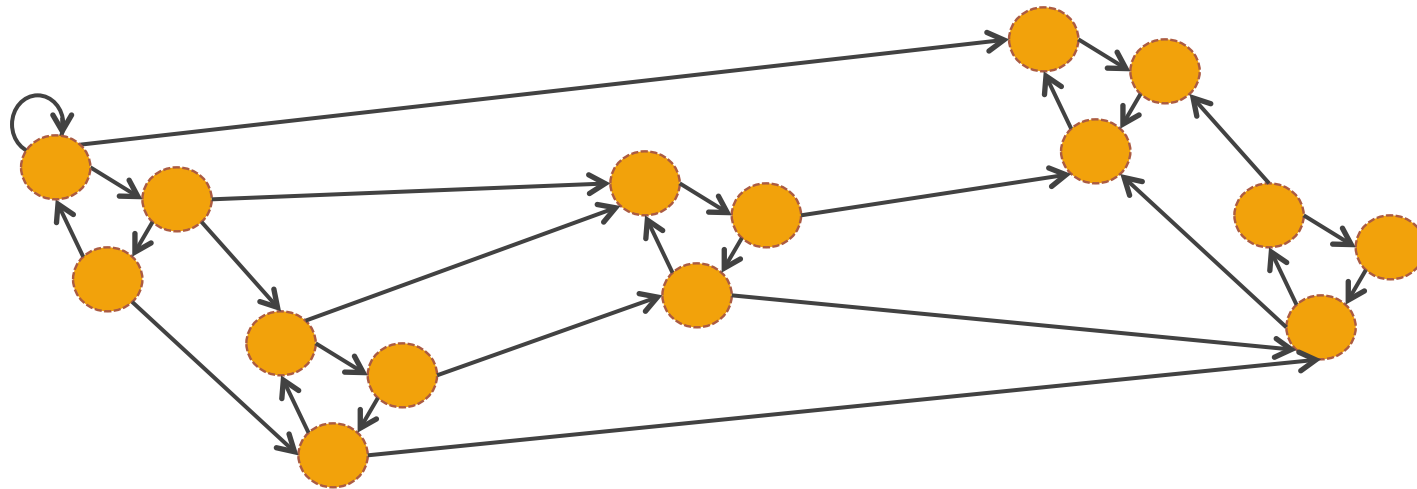
Combining Decidable Classes with the Graph of Rule Dependencies

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

fes = finite fact saturation

fus = finite query rewriting

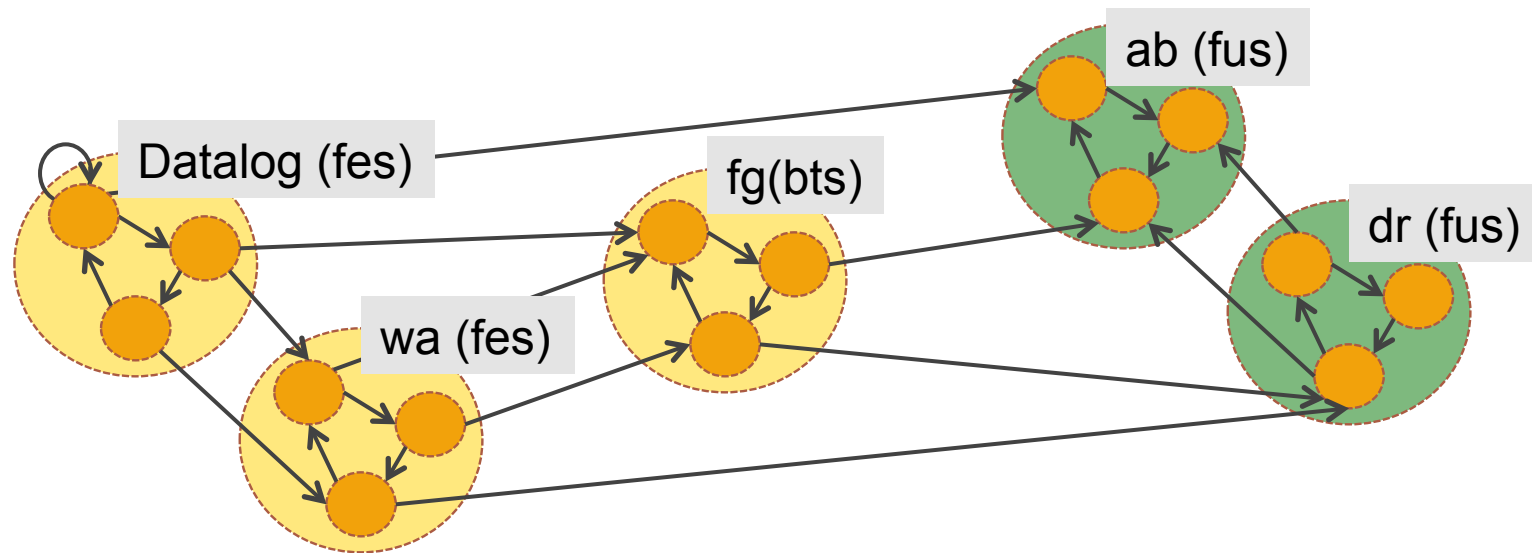
bts = (possibly infinite) tree-shaped saturation



Combining Decidable Classes with the Graph of Rule Dependencies

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

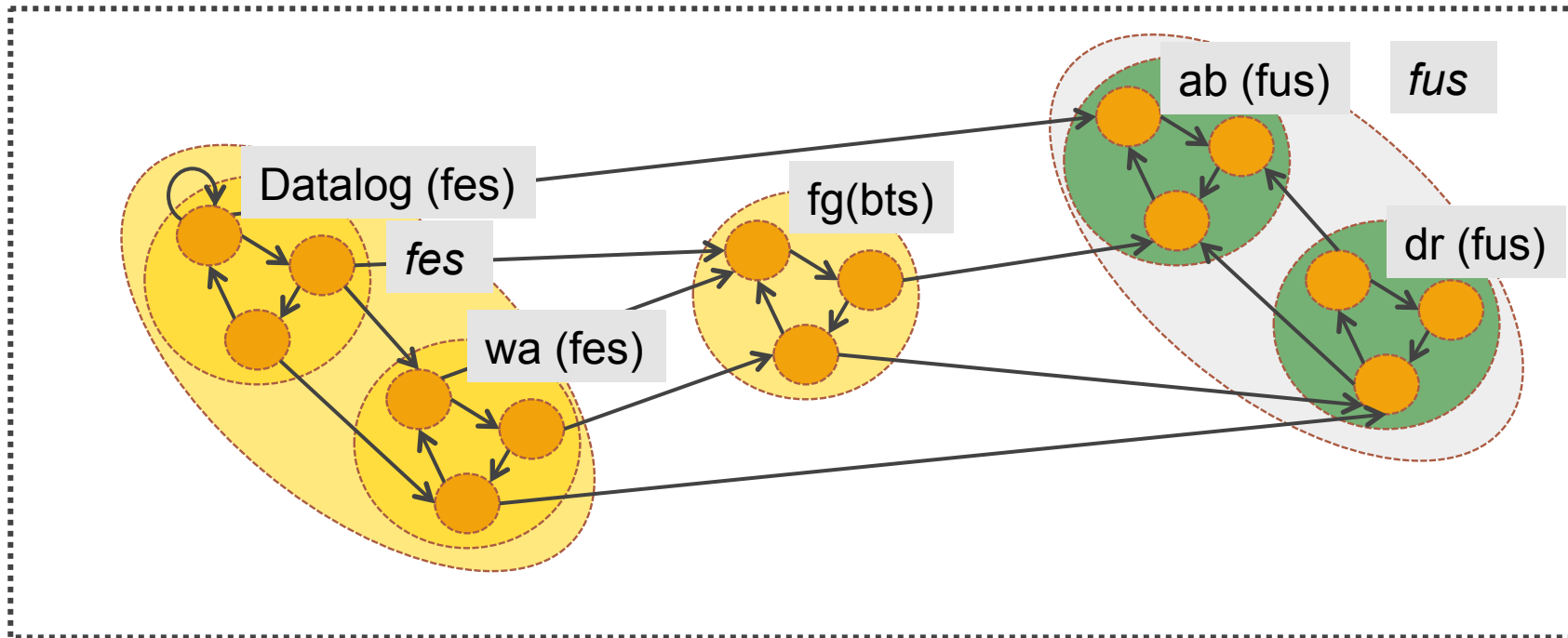
The same holds for *fus* (but not for *bts*)



Combining Decidable Classes with the Graph of Rule Dependencies

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

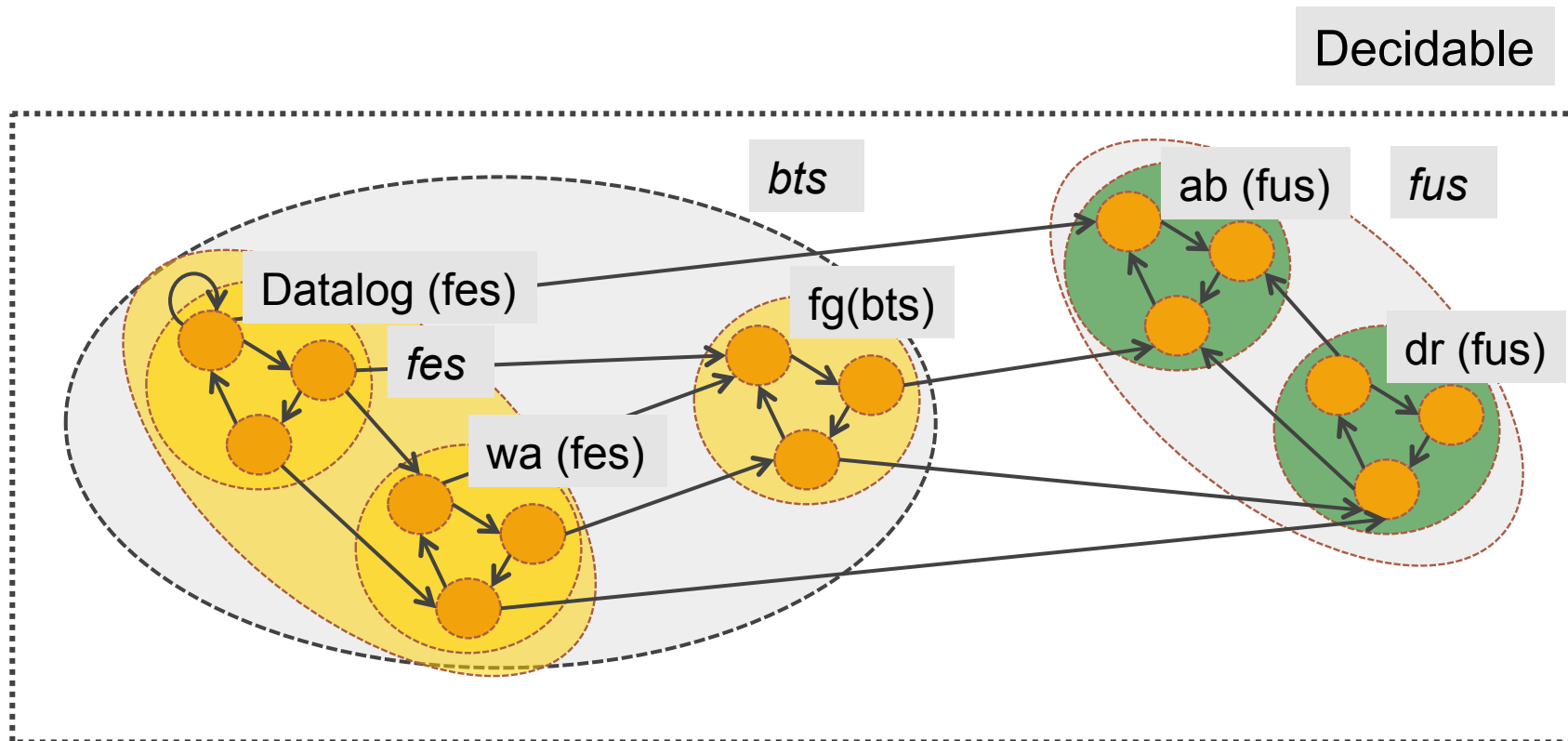
The same holds for *fus* (but not for *bts*)



Combining Decidable Classes with the Graph of Rule Dependencies

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



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