Reasoning on Data:

The Ontology-Mediated Query Answering Problem

Marie-Laure Mugnier

University of Montpellier

UNILOG School – Vichy – 2018

KNOWLEDGE REPRESENTATION AND REASONING (KR)

- A field historically at the heart of **Artificial Intelligence**
- Study **formalisms (or languages)** to
  - **represent** various kinds of human knowledge
  - do **reasoning** on these representations
- along the tradeoff **expressivity / tractability** of reasoning

→ KR languages based on **computational logic**

In this talk: classical first-order logic (FOL)

Major conferences: IJCAI, AAAI, KR

Overview of the tutorial

Part 1: Basics

Knowledge bases, Ontologies
Logical view of Queries and Data

Main KR formalisms to represent and reason with ontologies
Ontology-Mediated Query Answering

Part 2: KR formalisms and algorithmic approaches

Part 3: Decidability issues in the existential rule framework

KNOWLEDGE BASED SYSTEMS

Knowledge Base (KB) ⟷ Reasoning Services

- **General knowledge on the application domain**
  « Cats are Mammals »

  Ontology

- **Factual Knowledge**
  Description of specific individuals, situations, ...

  Félix is a Cat

  Factbase, Database

Knowledge expressed in a KR language

Fundamental tasks

- **Checking the consistency** of the KB
- **Computing answers** to a query over the KB

...

Reasoning algorithms associated with the KR language

What is an Ontology?

In computer science:

a **formal specification of the knowledge of a particular domain**

- which allows for machine processing
- that relies on the semantics of knowledge

> **automated reasoning**

Such a specification consists of

- a **vocabulary** in terms of concepts and relations
- **semantic relationships** between these elements

Examples of Ontologies

- **Medecine and life sciences** :

  hundreds of available ontologies

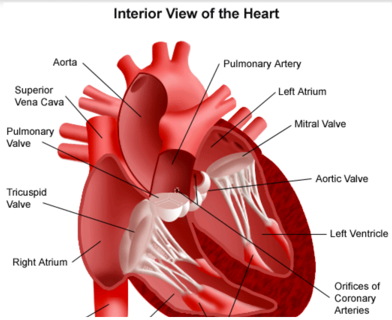
  - general medical ontologies

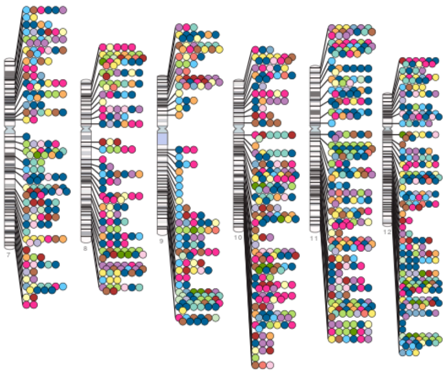
    SNOMED CT (400 000 terms)

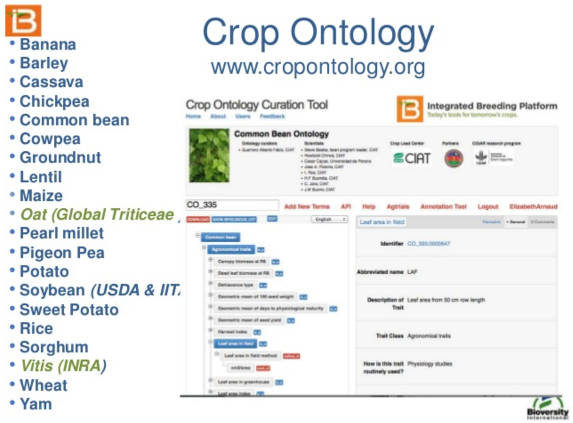
    GALEN (> 30 000 terms)
  - specialized medical ontologies

    FMA (anatomy)

    NCI (cancer), ...
  - biology
  - agronomy







- **Information systems of large organizations and corporations**

At the Core of Ontologies: Concepts / Classes

- **Concept / class** : a category of entities (objects) that share properties

  In FOL: unary predicate: Cat, Mammal

- **Instance** of class: a specific member of this class

  In FOL: a term (variable or constant)

- Fundamental relation on classes : **specialization (subsumption)** («is a kind of », «subclass of »)

C2

↑

C1

Semantics : every instance of C1 is also instance of C2

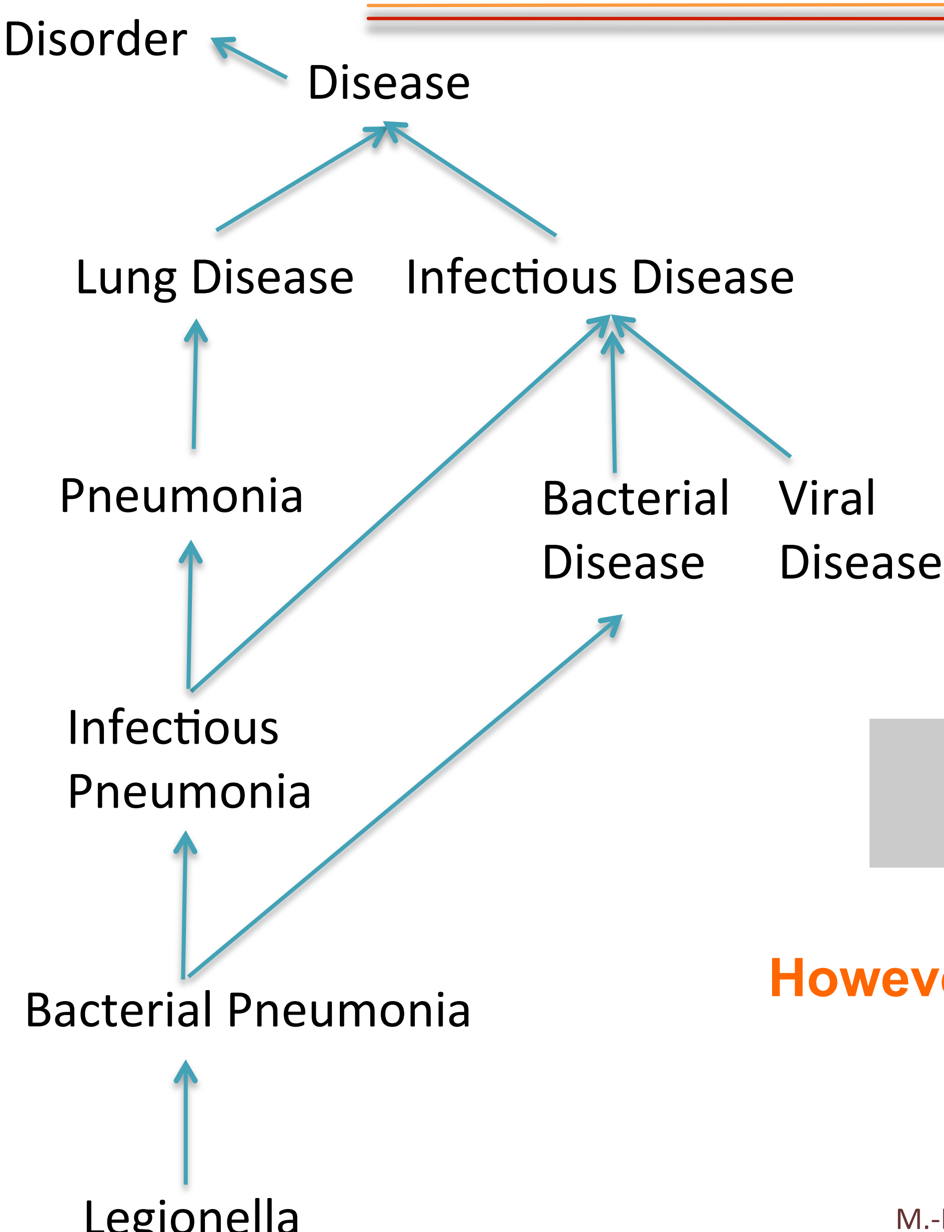
$\forall x\ (C1(x) \rightarrow C2(x))$

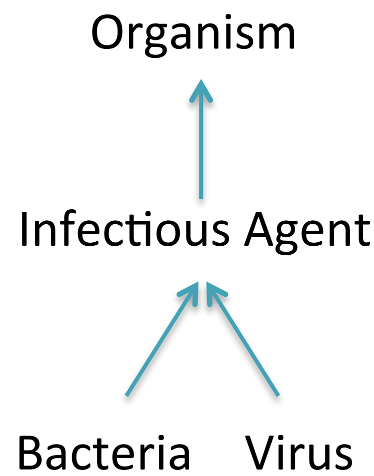
$\forall x\ (Cat(x) \rightarrow Mammal(x))$

C1 **subclass of** C2

At the Heart of Ontologies: Concepts / Classes

(SNOMED CT)





Concepts organized by specialization

However, an ontology is not just a classification!

ONTOLOGIES ARE MUCH MORE THAN CLASSIFICATIONS

« An ontology specifies the **vocabulary** of an application domain and **semantic relationships** between the terms of the vocabulary»

Vocabulary

1. **concepts / classes**
2. relations (between instances)

+ semantic relationships between concepts

+ semantic relationships between relations

+ properties of concepts

+ properties of relations

+ other axioms that more generally express domain knowledge

Relations between instances

Often these are binary relations (also called « roles » or « properties »)

dueTo

Argument 1 : a Disorder
Argument 2 : an Organism or [...]

subrelation of

hasCausativeAgent

Argument 1 : a Disease
Argument 2 : an Organism

$$\forall x \forall y\ (hasCA(x,y) \rightarrow dueTo(x,y))$$

Signature of a relation : assigns a maximum concept to each argument
(« domain » and « range » in OWL)

$$\forall x \forall y\ (hasCA(x,y) \rightarrow Disease(x) \wedge Organism(y))$$

Examples of other frequent types of axioms

- Negative constraints (disjointness between concepts, relations, ...)

Bacteria $\cap$ Virus = $\emptyset$

$\forall x$ (Bacteria(x) $\wedge$ Virus(x) $\rightarrow$ $\perp$)
$\forall x$ (Bacteria(x) $\rightarrow$ $\neg$ Virus(x))

- Necessary and/or sufficient properties of concepts (ex: BacterialDisease)

A bacterial disease is caused by a bacteria

$\forall x$ (BacterialDisease(x) $\rightarrow$ $\exists y$ (Bacteria(y) $\wedge$ hasCausativeAgent(x,y))

- Properties of relations

inverse relations: $\forall x \forall y$ (hasPart(x,y) $\leftrightarrow$ isPartOf(y,x))

symmetry, transitivity, ...

functional relation: $\forall x \forall y \forall z$ (isPartOf(x,y) $\wedge$ isPartOf(x,z) $\rightarrow$ y = z)

What kinds of Languages to Express Ontologies?

Very light languages

Hierarchies of classes
Hierarchies of binary relations (called « properties »)
Signatures of these relations (« domain » and « range »)
→ **OWL DL fragment of RDF Schema** (Semantic Web)

More expressive fragments of first-order logics

Description Logics
Rule-based languages Datalog, existential rules, RDF deductive rules, Answer Set Programming ...

From a logical viewpoint: an ontology is composed of

a **finite set of predicates** (to express concepts and relations)
a **finite set of (closed) formulas** over these predicates
of the form $\forall X$ **(condition[X] → conclusion[X])**

What are Ontologies Good for?

- **provide a common vocabulary**

  → it is easier to share information
  (typically between experts of several domains)

- **constrain the meaning of terms**
  → forces to explicit not-said things and to remove ambiguities
  hence less misunderstandings

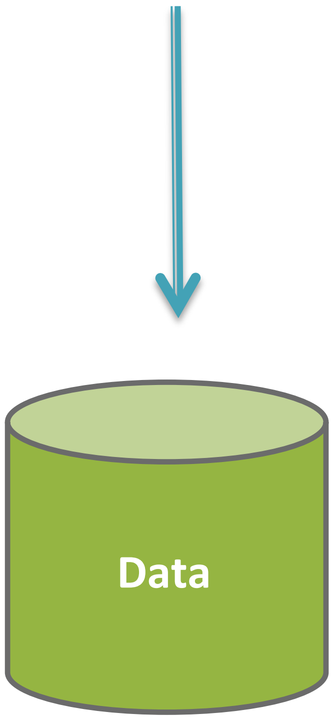
- to do **automated reasoning**, basis of high-level services

  → find implicit links between pieces of knowledge
  → check the consistency of the KB, find errors in modeling
  → enrich data query answering

ONTOLOGY-MEDIATED QUERY ANSWERING (EX: MEDICAL RECORDS)

Query (SQL, SPARQL, MongoDB ...)

« find all patients affected by a lung disease due to a bacteria »



??

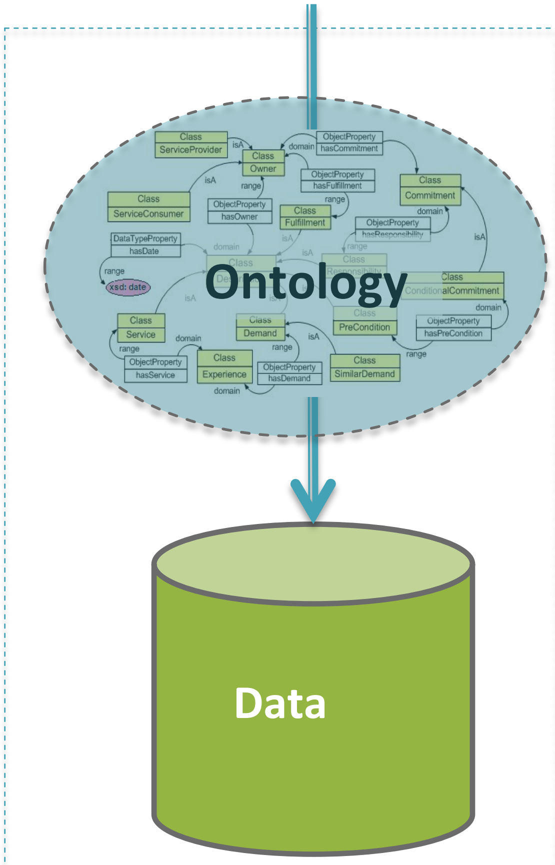
Patient P : Diagnosis = « legionella »

Database (relational, RDF, NoSQL, ...)

Ontology-Mediated Query Answering

Query

« find all patients affected by a lung disease due to a bacteria »



A legionella is bacterial pneumonia
A bacterial pneumonia is a pneumonia
A pneumonia is a **lung disease**
A bacterial pneumonia is caused by a **bacteria**
If *x* is caused by *y* then *x* **is due to** *y*
If the diagnosis of a patient *x* contains a disease y then *x* is **affected by** *y*

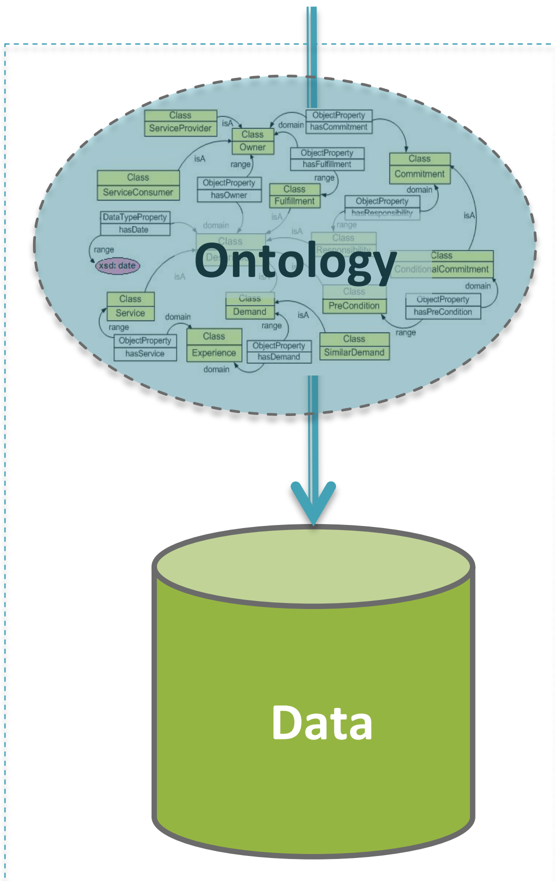
Patient P : Diagnosis = « legionella »

Knowledge Base

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)

Adding an ontological layer on top of data

Query



Knowledge base

1- **Enrich** the vocabulary

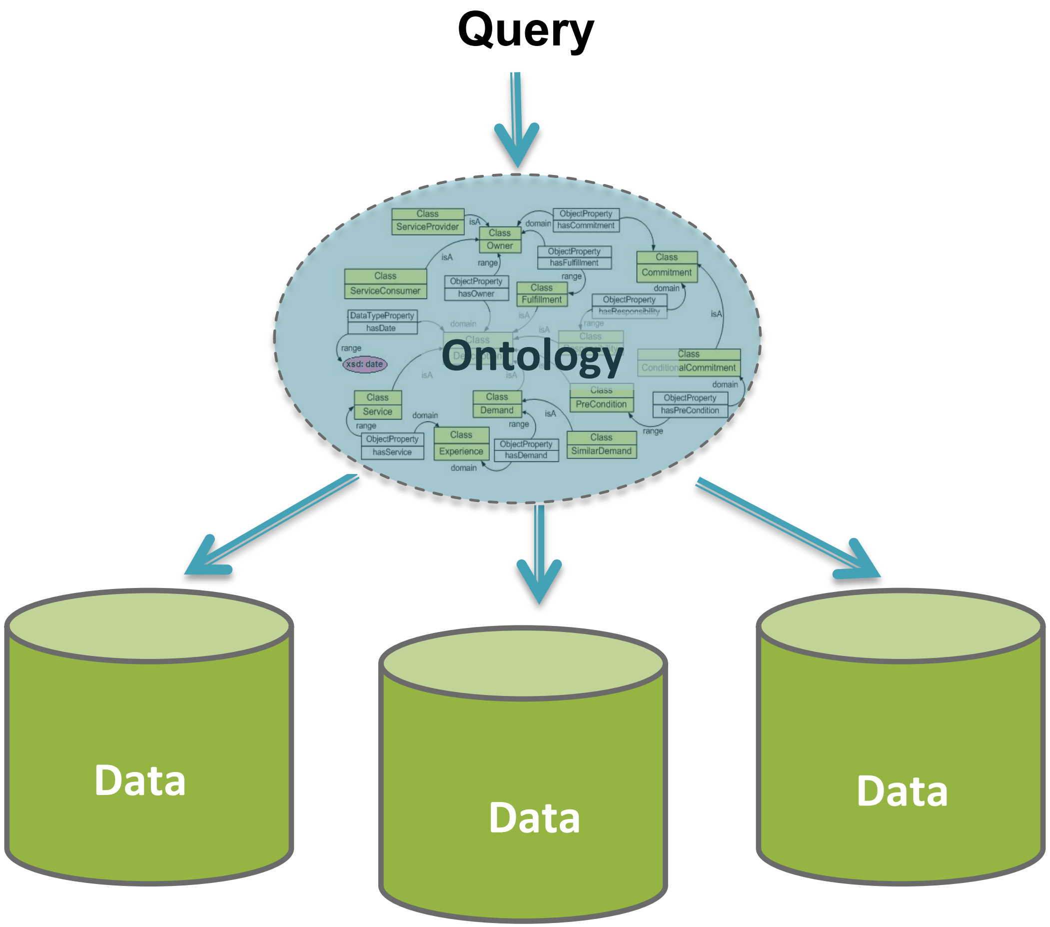
allowing to **abstract** from a specific data storage

2 - **Infer** new facts, not explicitely stored,

allowing for **incomplete data** representation

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)

3 – provide a **unified view** of multiple sources



OMQA Example: Ontological Knowledge

A legionella is bacterial pneumonia

$\forall x\ (Legionella(x) \rightarrow BacterialPneumonia(x))$

A bacterial pneumonia is a pneumonia
A pneumonia is a lung disease
A bacterial pneumonia is caused by a bacteria

$\forall x\ (BacterialPneumonia(x) \rightarrow \exists y\ (hasCausativeAgent(x,y) \wedge Bacteria(y)))$

If *x* is caused by *y* then *x* is due to *y*

$\forall x \forall y\ (hasCausativeAgent(x,y) \rightarrow dueTo(x,y))$

If the diagnosis of a patient *x* contains a disease y then *x* is affected by *y*

$\forall x \forall y\ ((Diagnosis(x,y) \wedge Disease(y)) \rightarrow isAffectedBy(x,y))$

FACTBASE

Factbase : usually a set of **ground atoms** (on the ontological vocabulary) seen as the conjunction of these atoms

« The diagnosis for the patient P is legionella »

F = { Patient(P), Diagnosis(P,M), Legionella(M) }

A **relational database** may naturally be viewed as a factbase

Relational schema : finite set R of relations → predicates
infinite domain of values → constants

Instance of a relation r : finite set of tuples on r → atoms on r

r

attr1	attr2
a1	a2
a2	a3
a1	a1

{ r(a1,a2), r(a2,a3), r(a1,a1) }

Database instance = { instance for each r in R } → factbase

CONJUNCTIVE QUERIES (CQ)

« find all patients affected by a lung disease due to a bacteria »

$q(x) = \exists y\ \exists z\ (Patient(x) \wedge isAffBy(x,y) \wedge LungDisease(y) \wedge dueTo(y,z) \wedge Bacteria(z))$

A **CQ** is an **existentially quantified conjunction of atoms**

The **free variables** are the **answer variables**

If closed formula: **Boolean CQ**

Datalog notation

ans(x) ← Patient(x), isAffBy(x,y), LungDisease(y), dueTo(y,z), Bacteria(z)

Select-Join-Project queries in relational algebra (SQL)

SELECT ... FROM … WHERE *<join conditions>*

SPARQL (semantic web queries)

SELECT … WHERE *<basic graph pattern>*

ANSWERS TO A CONJUNCTIVE QUERY

- The **answer** to a Boolean CQ q in F is *yes* if $F \models q$ *yes = ()*

- Let the CQ $q(x_1,\ldots,x_k)$. A tuple $(a_1, \ldots, a_k)$ of *constants* is an answer to q with respect to a factbase F if $F \models q[a_1,\ldots,a_k]$, where $q[a_1,\ldots,a_k]$ is obtained from $q(x_1,\ldots,x_k)$ by replacing each x_i by a_i

- Let F and q be seen as sets of atoms. A **homomorphism** h from q to F is a mapping from *variables(q)* to *terms(F)* such that $h(q) \subseteq F$

$F \models q()$ iff q can be mapped by homomorphism to F

$(a_1, \ldots, a_k)$ is an answer to $q(x_1,\ldots,x_k)$ w.r.t. F iff there is a homomorphism from q to F that maps each x_i to a_i

Key notion: homomorphism

$q(x) = \exists\, y\ (movie(y) \wedge play(x, y))$

movie(y)
play(x, y)

F

movie(m1)
movie(m2)
movie(m3)
actor(a)
actor(b)
actor(c)
play(a,m1)
play(a,m2)
play(c,m3)

Homomorphism *h* from *q* to *F*:
substitution of var(*q*) *by terms(F)*
such that h(q) $\subseteq$ *F*

h1 : x → a y → m1	h1(q) = movie(m1) ∧ play(a, m1)
h2 : x → a y → m2	h2(q) = movie(m2) ∧ play(a, m2)
h3 : x → c y → m3	h3(q) = movie(x0) ∧ play(c, m3)

Answers: obtained by restricting the domains of homomorphisms to answer variables

x = a
x = c

On the OMQA Example

$q(x) = \exists y \exists z\, (Patient(x) \wedge isAffectedBy(x,y) \wedge LungDisease(y) \wedge dueTo(y,z) \wedge Bacteria(z))$

« find all patients affected by a lung disease due to a bacteria »

Factbase = { Patient(P), Diagnosis(P,M), Legionella(M) }

« The diagnosis for the patient P is legionella »

Legionella *specialisation of* LungDisease *and* BacterialDisease (*and* Disease)

hence LungDisease(M)

hence BacterialDisease(M), Disease(M)

$\forall x\, (BacterianDisease(x) \rightarrow \exists y\, (hasCausativeAgent(x,y) \wedge Bacteria(y)))$

hence hasCausativeAgent(M,b) and **Bacteria(b)**

$\forall x \forall y\, (hasCausativeAgent(x,y) \rightarrow dueTo(x,y))$

hence dueTo(M,b)

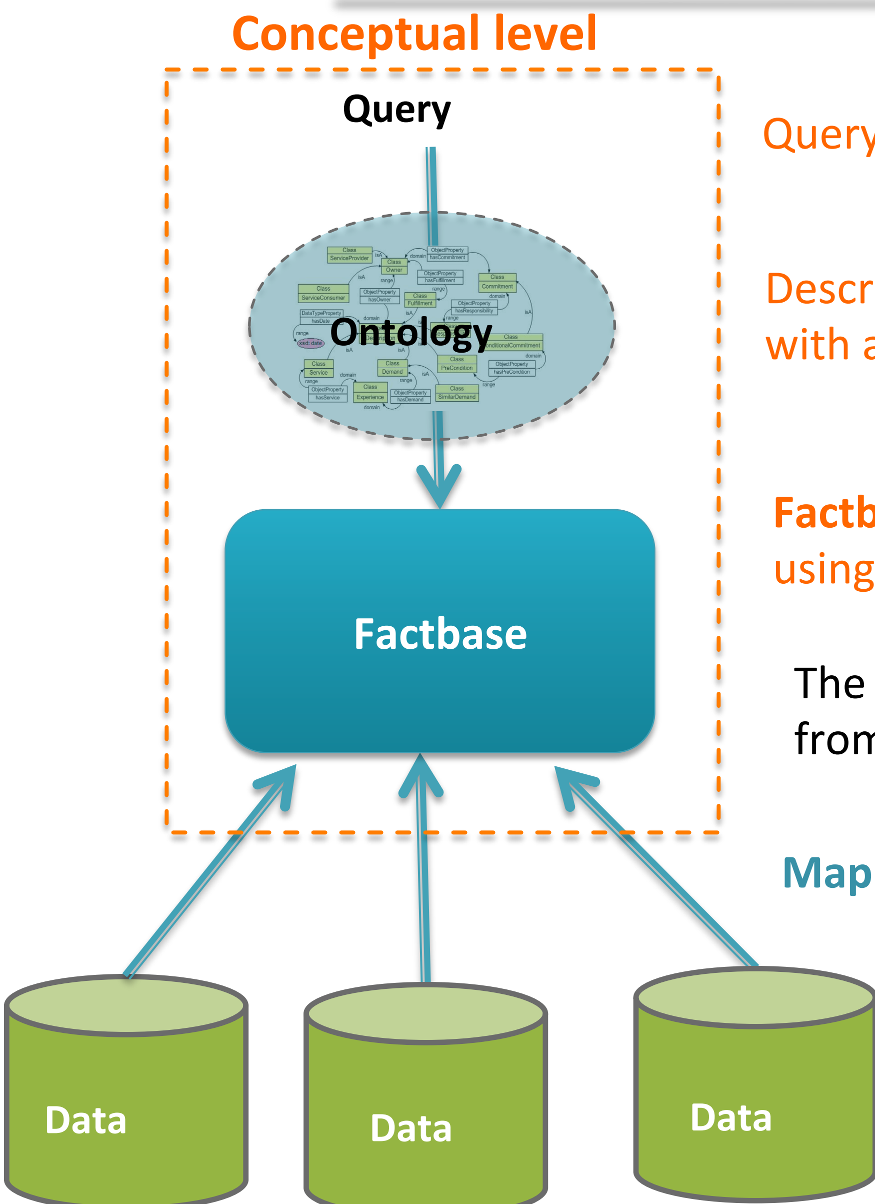
$\forall x \forall y\, ((Diagnosis(x,y) \wedge Disease(y)) \rightarrow isAffectedBy(x,y))$

hence isAffectedBy(P,M)

Answer : x = P

A MORE GENERAL SCHEMA

« Ontology-Based **Data Access** »
[Poggi et al., JoDS, 2008]



Query using the vocabulary of the ontology

Description of the application domain with a high abstraction level

Factbase (possibly virtual)
using the vocabulary of the ontology

The **answers** to the query are **inferred** from the knowledge base

Mappings from data to facts

{ Database query $\rightsquigarrow$ Facts }

Independent and heterogeneous data sources

MAPPINGS

Patient_T [ID_PATIENT, NAME,SSN]

Diagnosis_T[ID_PATIENT, DISORDER]

Patient /1
Diagnosis / 2
Legionella /1

Mapping: database query(X) $\rightsquigarrow$ conjunction with free variables X

q(x): $\exists n \exists s$ Patient_T (x,n,s) $\rightsquigarrow$ Patient(x)

q'(x): $\exists n \exists s$ Patient_T (x,n,s) $\wedge$ Diagnostic_T(x,y) $\wedge$ y = « Legionella »
$\rightsquigarrow$ $\exists z$ (diagnosis(x,z) $\wedge$ legionella(z))

Patient_T

id	name	ssn
P	..	..
..	..	..
..	..	..

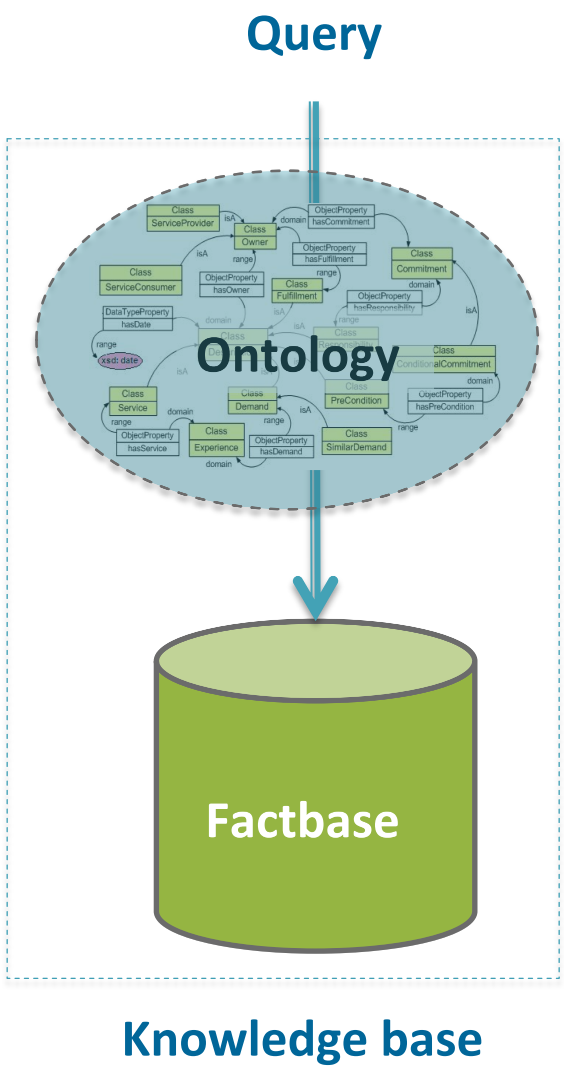
Diagnosis_T

id	dis
P	« Leg. »
..	..
..	..

$\rightsquigarrow$

Patient(P)
Diagnosis(P,M)
Legionella(M)

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)



(Boolean) conjunctive query *q*

Theory *O* in a suitable FOL fragment

Set of ground atoms (or existentially closed formula) *F*

Fundamental **decision problem**

$$O, F \models q \quad ?$$

Overview of the lecture

Part 1: Basics

Part 2: KR formalisms and algorithmic approaches

Outline of description logics – Horn DLs

Existential Rules

Materialization approach (forward chaining)

Query rewriting approach (related to backward chaining)

Part 3: Decidability issues in the existential rule framework

DESCRIPTION LOGICS

- A family of KR languages for representing and reasoning with ontologies
- Mostly correspond to **decidable fragments of FOL**
  (related to modal propositional logic, the guarded fragment of FOL, ...)
- **Variable-free syntax**
- Used to be called « **concept languages** »:
  from concept and role names (unary and binary predicates)
  and a **set of constructors**
  define complex concepts (more recently: complex roles)
- An ontology is a set of axioms that state **inclusions between concepts**
  (and between roles)

Description Logics: Building Blocks (Syntax)

Vocabulary

Atomic concepts: Human, Parent, Student … (unary predicates)

Atomic roles: parentOf, siblingOf, … (binary predicates)

Complex concepts and roles can be built using a **set of constructors (which depends on each particular DL)**

conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)

Human $\sqcap$ $\neg$Parent Female $\sqcup$ Male

restricted forms of existential and universal quantification ($\exists$, $\forall$)

$\exists$parentOf.(Female $\sqcap$ Student) $\forall$ parentOf.Male

inverse of a role ($^-$), composition of roles (o)

$\exists$parentOf$^-$ parentOf o parentOf

Description Logics: Building Blocks (Semantics)

To each **concept** is assigned a **FOL sentence with free variable** x

Human	Human(x)
Human ⊓ ¬Parent	Human(x) ∧¬Parent(x)
∃parentOf.(Female ⊓ Student)	∃y (parentOf(x,y) ∧ Female(y) ∧ Student(y))
∀ parentOf.Female	∀ y (parentOf(x,y) → Female(y))

To each **role** is assigned a **FOL sentence with 2 free variables** x and y

parentOf o parentOf	∃z (parentOf(x,z) ∧ parentOf(z,y))

Description Logics: Knowledge Base

Knowledge Base = TBox (ontology) + ABox (factbase)

Tbox: axioms of the form **$C1 \sqsubseteq C2$** $\forall x\ (\ fol(C1) \rightarrow fol(C2)\ )$

or **$r1 \sqsubseteq r2$** $\forall x \forall y\ (\ fol(r1) \rightarrow fol(r2)\ )$

$Human \sqsubseteq Male \sqcup Female$	$\forall x\ (Human(x) \rightarrow Male(x) \vee Female(x))$
$Adult \sqsubseteq \neg Child$	$\forall x\ (Adult(x) \wedge Child(x) \rightarrow \bot)$
$Parent \sqsubseteq \exists parentOf$	$\forall x\ (Parent(x) \rightarrow \exists y\ parentOf(x,y)\ )$
$HappyFather \sqsubseteq \forall parentOf.Female$	$\forall x\ (HP(x) \rightarrow (\forall y(parentOf(x,y) \rightarrow Female(y))$
$Human \sqsubseteq \exists parentOf^{-}.Human$	$\forall x\ (Human(x) \rightarrow \exists y\ (parentOf(y,x) \wedge Human(y)))$
$parentOf \circ parentOf \sqsubseteq ancestorOf$	$\forall x \forall y\ (\exists z(parentOf(x,z) \wedge parentOf(z,y)) \rightarrow ancestorOf(x,y)$

Abox : set of ground facts parentOf(A,B), Female(A), …

Description Logics: standard reasoning tasks

Standard reasoning tasks on a KB ($\mathcal{T}$,$\mathcal{A}$)

- Concept subsumption — $\mathcal{T} \vDash C \sqsubseteq D$?
- Concept satisfiability — is C satisfiable w.r.t. $\mathcal{T}$?
- KB satisfiability — is ($\mathcal{T}$,$\mathcal{A}$) satisfiable ?
- Instance checking — $(\mathcal{T},\mathcal{A}) \vDash C(b)$, where b is a constant?

All these tasks can be expressed in terms of KB (un)satisfiability provided that the constructors in the considered DL allow for it

Concept subsumption	$\mathcal{T} \vDash C \sqsubseteq D$ iff $(\mathcal{T}, \{C(a), \neg D(a)\})$ unsatisfiable
Concept satisfiability	C satisfiable w.r.t. $\mathcal{T}$ iff $(\mathcal{T}, \{C(a)\})$ satisfiable
Instance checking	$(\mathcal{T},\mathcal{A}) \vDash C(b)$ iff $(\mathcal{T},\mathcal{A} \cup \{\neg C(b)\})$ unsatisfiable

Query answering beyond instance checking?
cannot be reduced to the standard reasoning tasks

EVOLUTION OF DLS

Standard expressive DL $\mathcal{ALC}$

- Concepts: $C \ := \top \mid A \mid C_1 \sqcap C_2 \mid \exists R.C \mid \neg C \mid C_1 \sqcup C_2 \mid \forall R.C$

- TBox axioms: only concept inclusions

Satisfiability and instance checking in $\mathcal{ALC}$ are:
EXPTIME-complete in combined complexity
coNP-complete in data complexity

Even worse if we add inverse roles: 2EXPTIME-complete in combined complexity

Two complexity measures for Query Answering Problems

Problem: Given a KB = (O, F), with O the ontology and F the factbase, and a query q, is q entailed by the KB?

Combined complexity (usual complexity measure)

The input is O, F and q

E.g.,
q Boolean CQ, F factbase
Does $F \vDash q$?

Data complexity

NP-complete (combined)
PTime (data)

The input is F

(O and q supposed to be fixed)

This distinction comes from database theory:
the size of the query is negligible compared to the size of the data

Evolution of DLs

Standard expressive DL $\mathcal{ALC}$

- Concepts: $C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists R.C \mid \neg C \mid C_1 \sqcup C_2 \mid \forall R.C$
- TBox axioms: only concept inclusions

Satisfiability and instance checking in $\mathcal{ALC}$ are:
EXPTIME-complete in combined complexity
coNP-complete in data complexity

Even worse if we add inverse roles: 2EXPTIME-complete in combined complexity

Two factors led to the **evolution of description logics**:

1. practical use (e.g. SNOMED CT): people mostly use conjunction and existential quantification
2. complexity too high for query answering problems

New DLs with Lower Complexity

DL-Lite$_{\mathcal{R}}$

$$B_1 \sqsubseteq B_2 \quad B_1 \sqsubseteq \neg B_2 \quad S_1 \sqsubseteq S_2 \quad S_1 \sqsubseteq \neg S_2$$

where

$$B := A \mid \exists S \qquad S := R \mid R^-$$

Large ABoxes
Query answering

$\mathcal{EL}$

$$C_1 \sqsubseteq C_2$$

where

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists R.C$$

Large TBoxes
Classification

Common feature: no disjunction (no « true » negation)

Then a satisfiable KB has a **unique canonical model** *M:*

For any Boolean CQ *q*, KB $\vDash$ *q* iff *M* is a model of *q*

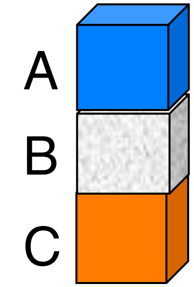
Reasoning techniques for these lighter DLs
are very similar to forward or backward chaining in rule-base systems

COMPLEXITY INTRODUCED BY DISJUNCTION OR NEGATION

KB ($\mathcal{T}$,$\mathcal{A}$)

$\mathcal{T}$: $\top \sqsubseteq$ Blue $\sqcup$ Other

$\mathcal{A}$: Blue(A), Other(C), on(A,B), on(B,C)



q(): $\exists x \exists y$ (Blue(x) $\wedge$ on(x,y) $\wedge$ Other(y))

To answer q, we have to consider two cases:

in each model of the KB, either Blue(B) or Other(B) holds

Similarly if we replace $\mathcal{T}$ by: $\neg$Blue $\sqsubseteq$ Other (equivalent axiom)

Note that Other $\sqsubseteq$ $\neg$Blue is harmless: it is just a disjointness constraint

In summary

DL ontology (TBox) has axioms of the form

$\forall x\, (fol(C_1) \rightarrow fol(C_2))$

$\forall x \forall y\, (fol(r_1) \rightarrow fol(r_2))$ where fol(r) is a path of atomic roles or their inverses

DLs essentially satisfy the **tree model property**:

if a KB is satisfiable then it has a « tree-shaped » model

With the new DLs: left and right parts of the implication are both **existentially quantified conjunctions of atoms**

called « **Horn description logics** »

WHY « HORN DLS » ON AN EXAMPLE

$\mathcal{EL}$ Axiom $$C \sqcap \exists R.\top \sqsubseteq \exists S.\exists R.B$$

FOL translation $$\forall x((C(x) \land \exists y R(x,y)) \to \exists u(S(x,u) \land \exists v(R(u,v) \land B(v))))$$

$$\forall x \forall y((C(x) \land R(x,y)) \to \exists u \exists v(S(x,u) \land R(u,v) \land B(v)))$$

prenex form $$\forall x \exists u \exists v \forall y(\neg C(x) \lor \neg R(x,y) \lor (S(x,u) \land R(u,v) \land B(v)))$$

Let us skolemize (*u* and *v* resp. replaced by $f_1(x)$ and $f_2(x)$):

$$\forall x \forall y(\neg C(x) \lor \neg R(x,y) \lor (S(x,f_1(x)) \land R(f_1(x), f_2(x)) \land B(f_2(x))))$$

we obtain a set of 3 Horn clauses (with skolem terms)

$$(\neg C(x) \lor \neg R(x,y) \lor S(x,f_1(x))) \quad (\neg C(x) \lor \neg R(x,y) \lor R(f_1(x), f_2(x))) \quad (\neg C(x) \lor \neg R(x,y) \land B(f_2(x))))$$

Hence the name **Horn description logics**

Existential Rules

$\forall X\ \forall Y\ (\ \underbrace{Body\ [X,Y]}\ \rightarrow\ \exists\ Z\ \underbrace{Head\ [X,Z]}\ )$

X, Y, Z :
sets of variables

any **positive conjunction** (without functional symbols except constants)

$\forall x\ (\ actor(x) \rightarrow\ \exists\ z\ play(x,z)\ )$

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

we often simplify by omitting universal quantifiers

Key point: ability to assert **the existence of unknown entities**

Crucial for representing ontological knowledge in **open domains**

See « **value invention** » in databases

DATA / FACTS

Relational database

Movie

m_id	
m1	...
m2	...
?x	

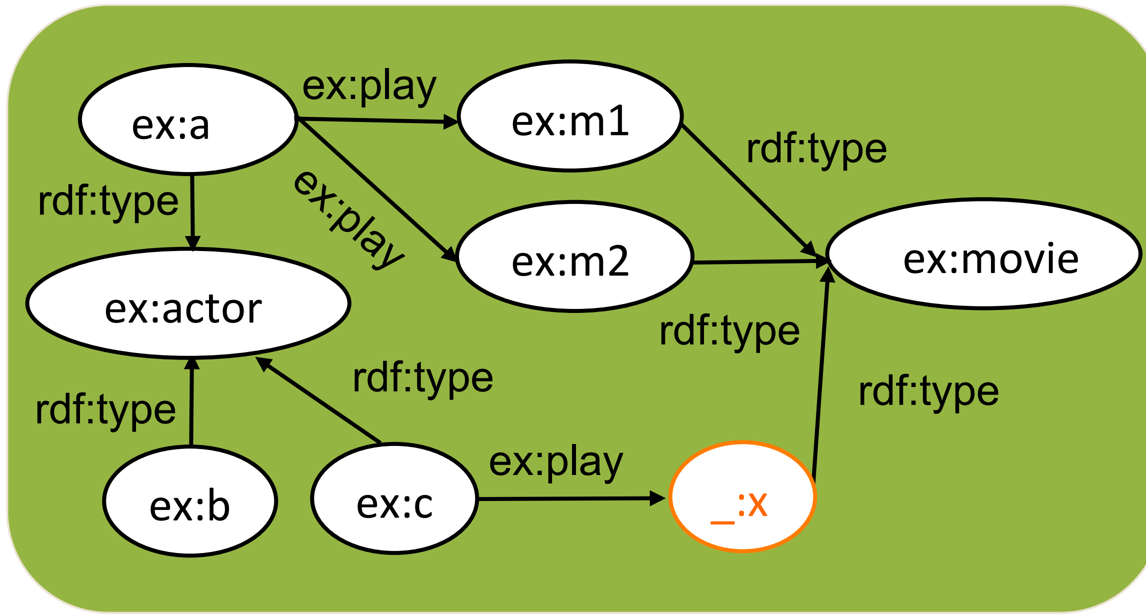
Actor

a_id	
a	...
b	...
c	

Play

m_id	a_id
a	m1
a	m2
c	?x

RDF



Etc.

Abstraction in first-order logic (FOL)

$$\exists x\ (\ movie(m1) \wedge movie(m2) \wedge movie(x)$$
$$actor(a) \wedge actor(b) \wedge actor(c)$$
$$play(a,m1) \wedge play(a,m2) \wedge play(c,x)\ )$$

We generalize here
the classical notion of a fact
by allowing existential variables

fact / factbase =
existentially closed conjunction of atoms

LABELLED HYPERGRAPH / GRAPH REPRESENTATION

- A fact or a set of facts can be seen as a **set of atoms**

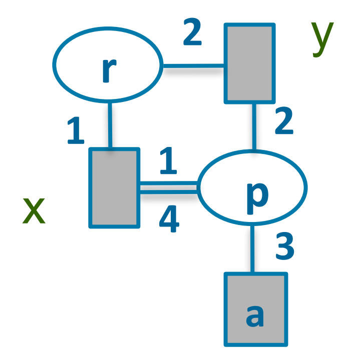
movie(m1), movie(m2), movie(x), actor(a), actor(b), actor(c), play(a,m1), play(a,m2), play(c,x)

→ hence a **hypergraph**

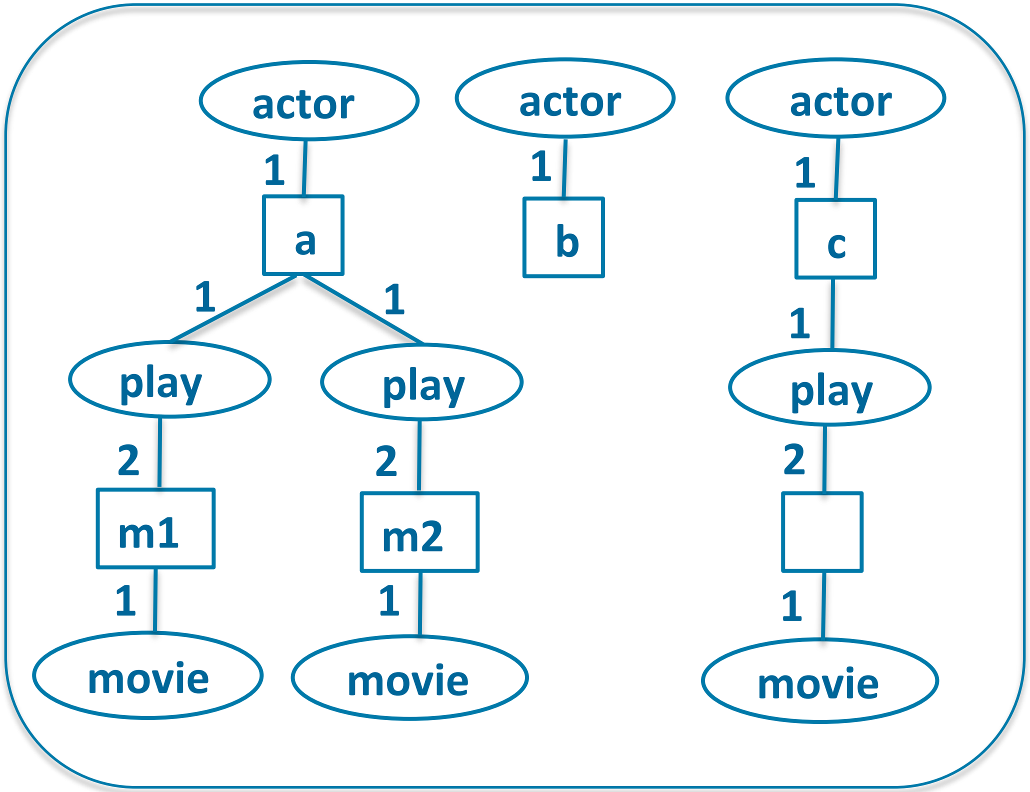
or its associated **bipartite (multi-)graph**

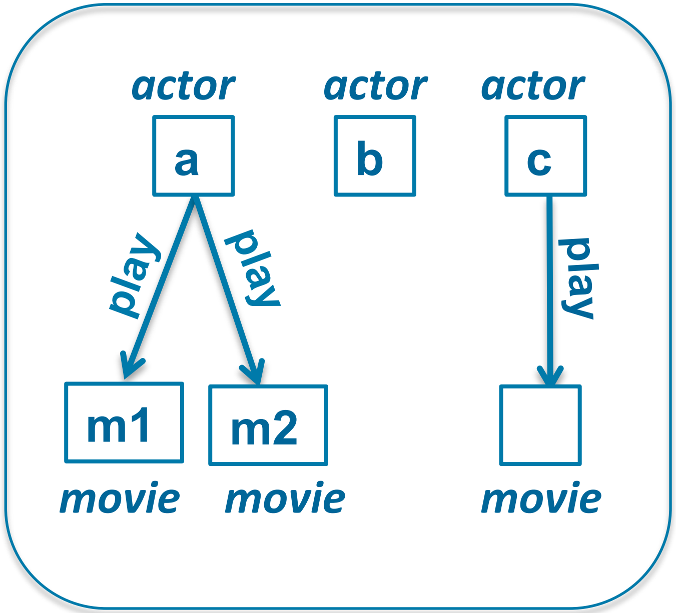
- one (labelled) node per term
- one (labelled) node per atom (~ hyperedge)
- totally ordered edges

p(x,y,a,x), r(x,y)



movie(m1), movie(m2), movie(x), actor(a), actor(b), actor(c), play(a,m1), play(a,m2), play(c,x)



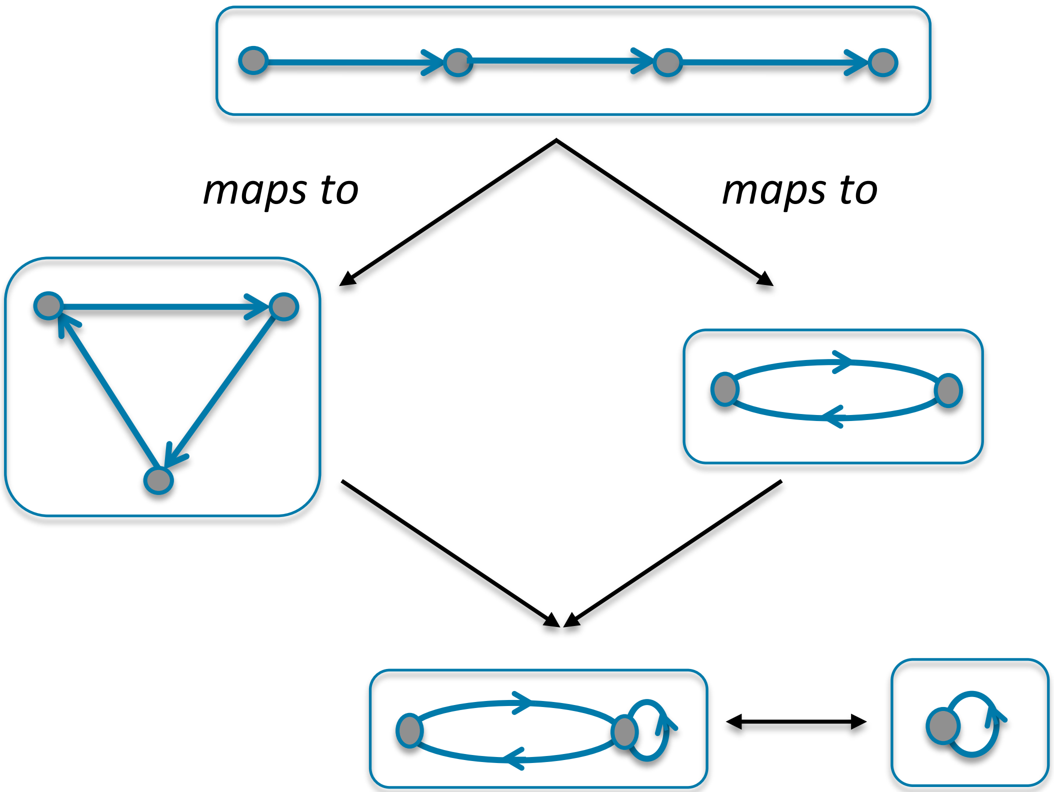


If predicates are at most binary:
atom nodes can be replaced by **labels** and **directed edges**

Graph homomorphisms (1)

- Let $G_1=(V_1,E_1)$ to $G_2=(V_2,E_2)$ be classical graphs.

Homomorphism *h* from G_1 to G_2: mapping from V_1 to V_2 s. t.
for every edge (u,v) in E_1, (h(u),h(v)) is in E_2

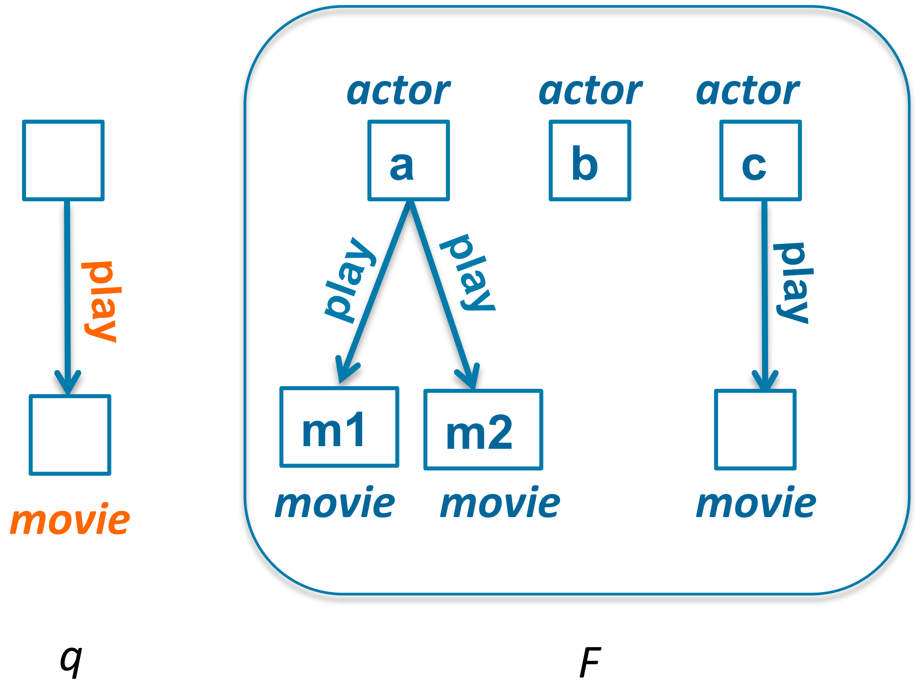


Graph homomorphisms (2)

- Let $G_1=(V_1,E_1)$ to $G_2=(V_2,E_2)$ be classical graphs.

Homomorphism *h* from G_1 to G_2: mapping from V_1 to V_2 s. t.
for every edge (u,v) in E_1, (h(u),h(v)) is in E_2

- If there are labels: they have to be ``kept'' as well

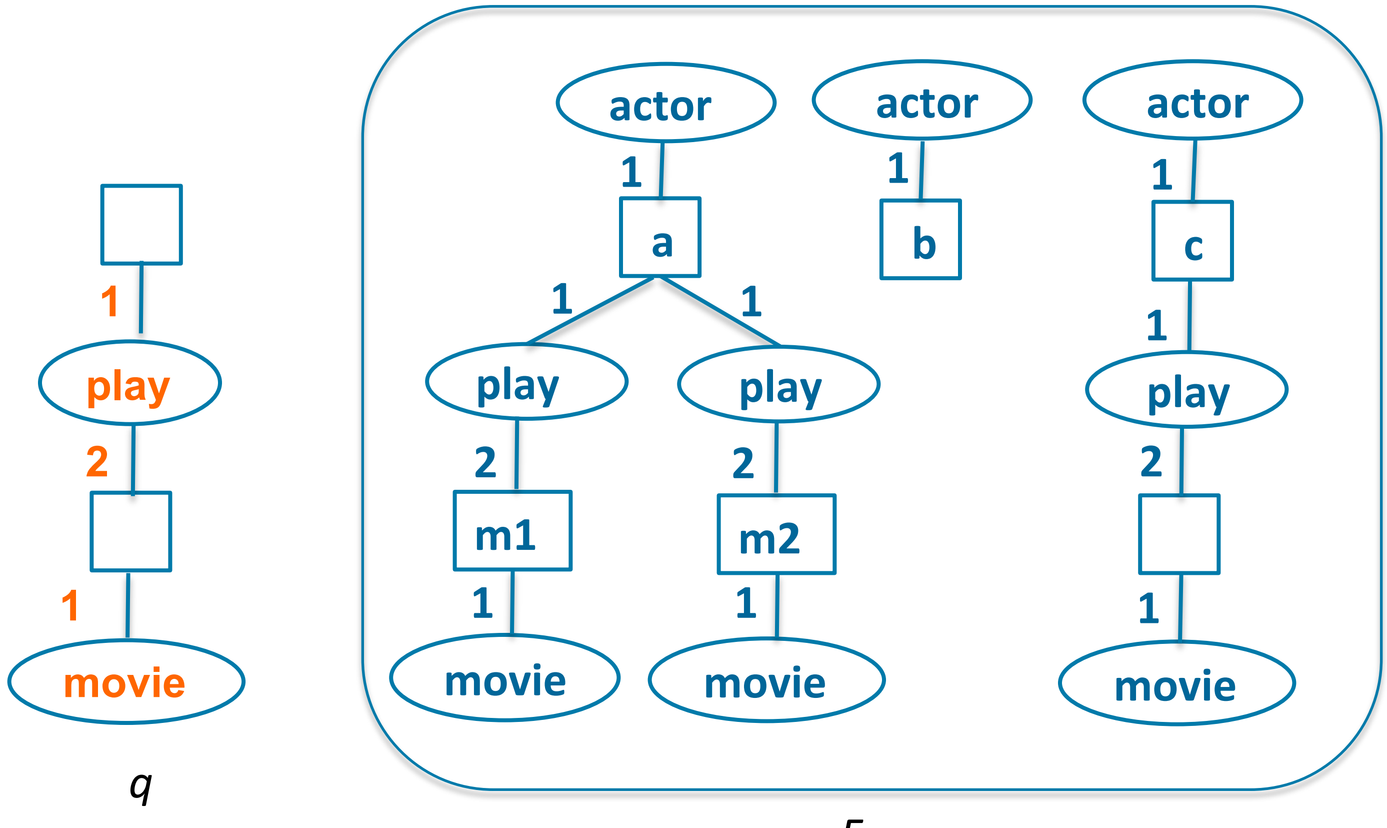


GRAPH HOMOMORPHISMS (3)

- Let $G_1=(V_1,E_1)$ to $G_2=(V_2,E_2)$ be classical graphs.

Homomorphism *h* from G_1 to G_2: mapping from V_1 to V_2 s. t.
for every edge (u,v) in E_1, (h(u),h(v)) is in E_2

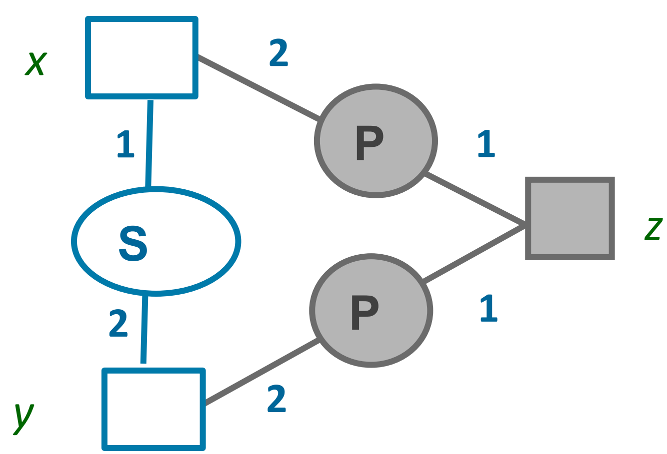
- If there are labels: they have to be ``kept'' as well

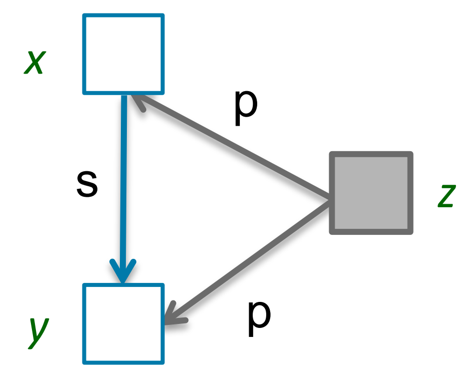


Graph View of Existential Rules

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

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





The rule head has 2 kinds of variables:
- **frontier**: shared with the body
- **existential** (new ``blank'' nodes)

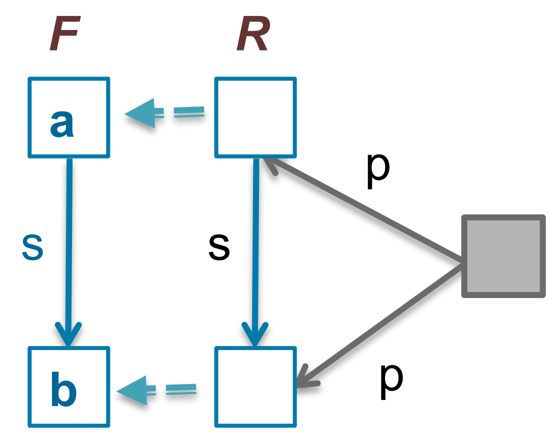
Generation of fresh (unknown) individuals

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

F = siblingOf(a,b)

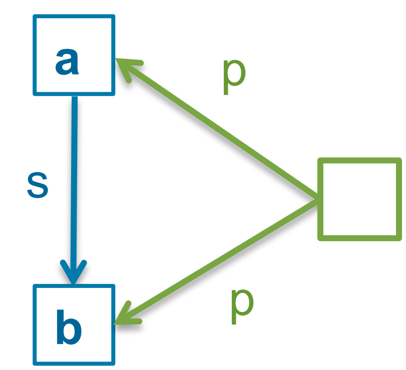
R is **applicable** to *F* if there is a **homomorphism** *h* from *body(R)* to *F*

x → a
y → b

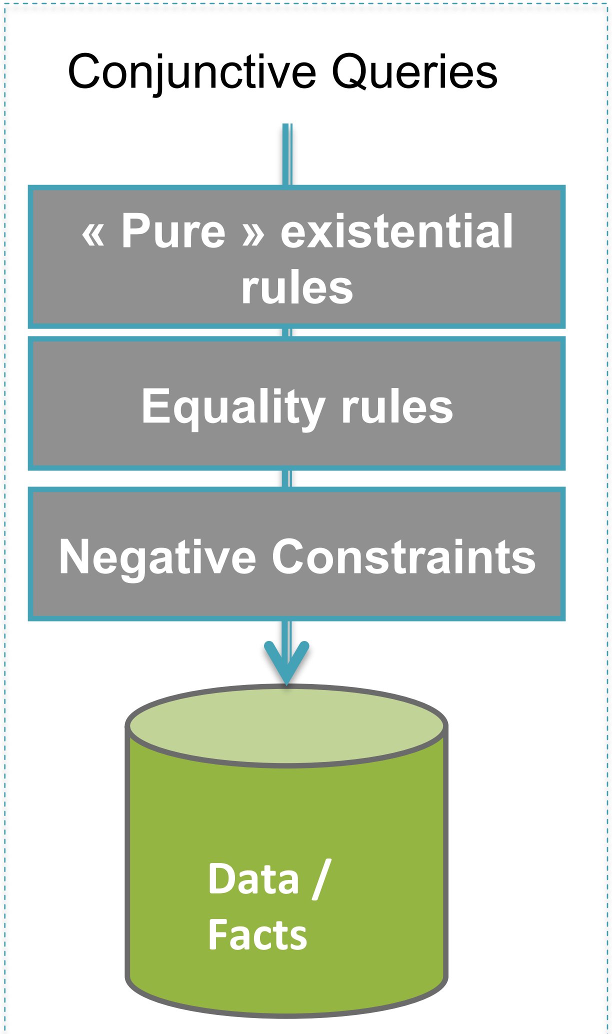


Applying *R* to *F* w.r.t. *h* produces *F* ∪ *h(head(R))* *where a new variable is created for each* existential *variable in R*

F' = **∃ z0** (siblingOf(a,b) ∧ parentOf(z0,a) ∧ parentOf(z0,b))



Existential Rule Framework (Logical / Graphical)



$q(x) = \exists y\, (movie(y) \wedge play(x, y))$

$\forall x\, (\, actor(x) \rightarrow \exists z\, (movie(z) \wedge play(x,z))\,)$

$\forall x\, \forall y\, \forall z\, (\, movie(y) \wedge director(x,y) \wedge director(z,y) \rightarrow x = z\,)$

$\forall x\, (\, movie(x) \wedge person(x) \rightarrow \bot\,)$

movie(m1)
play(a,m1)
play(c, *x*)
...

MULTIPLE THEORETICAL FOUNDATIONS

Conceptual Graphs
[Sowa 1984]
[Chein Mugnier 1992, 2009]

logical translation

∀ ∃ -rules, existential Rules [Baget+ IJCAI 2009]

Datalog+/- [Cali+ PODS 2009]

Datalog (70-80s)

+ « value invention »

+ «unrestricted cycles » on variables
+ unbounded arity

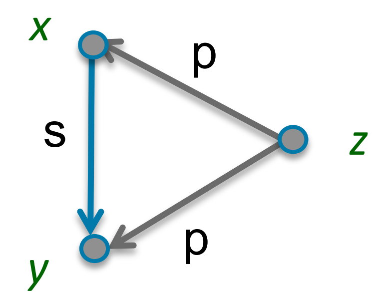
Lightweight Description Logics,
e.g. OWL 2 tractable profiles
More generally, Horn DLs

- Same logical form as « Tuple-Generating Dependencies » (TGDs) long studied in relational databases

EXISTENTIAL RULES ARE MORE EXPRESSIVE THAN HORN-DLS

- The FOL translation of Horn DLs yields existential rules
- Existential rules are strictly more expressive:

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



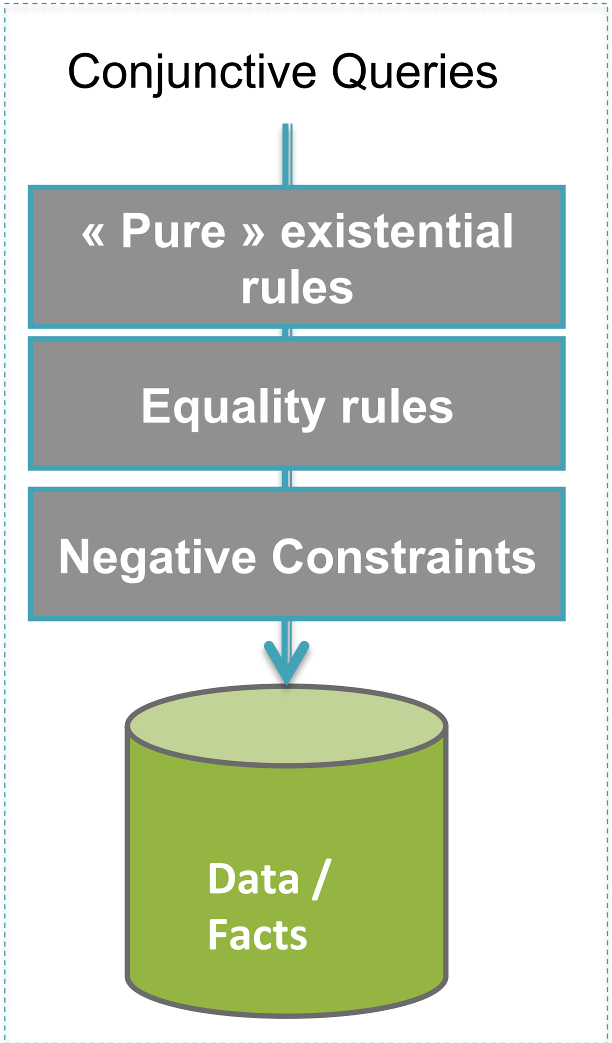
cannot be expressed in most DLs because of the « cycle on variables »
(needs role composition: $s \sqsubseteq p \circ p$ *)*

More complex interactions between variables cannot be expressed at all in DLs

- The unbounded predicate arity allows for more flexibility:
  - → direct translation of database relations
  - → adding contextual information is easy (provenance, trust, etc.)

Unsurprisingly, this added expressivity has a cost

Existential Rule Framework



- **Fundamental decision problem**

  Input: $\mathcal{K}= (F, \mathcal{R})$ knowledge base
  q Boolean conjunctive query

  Question: is q entailed by $\mathcal{K}$?

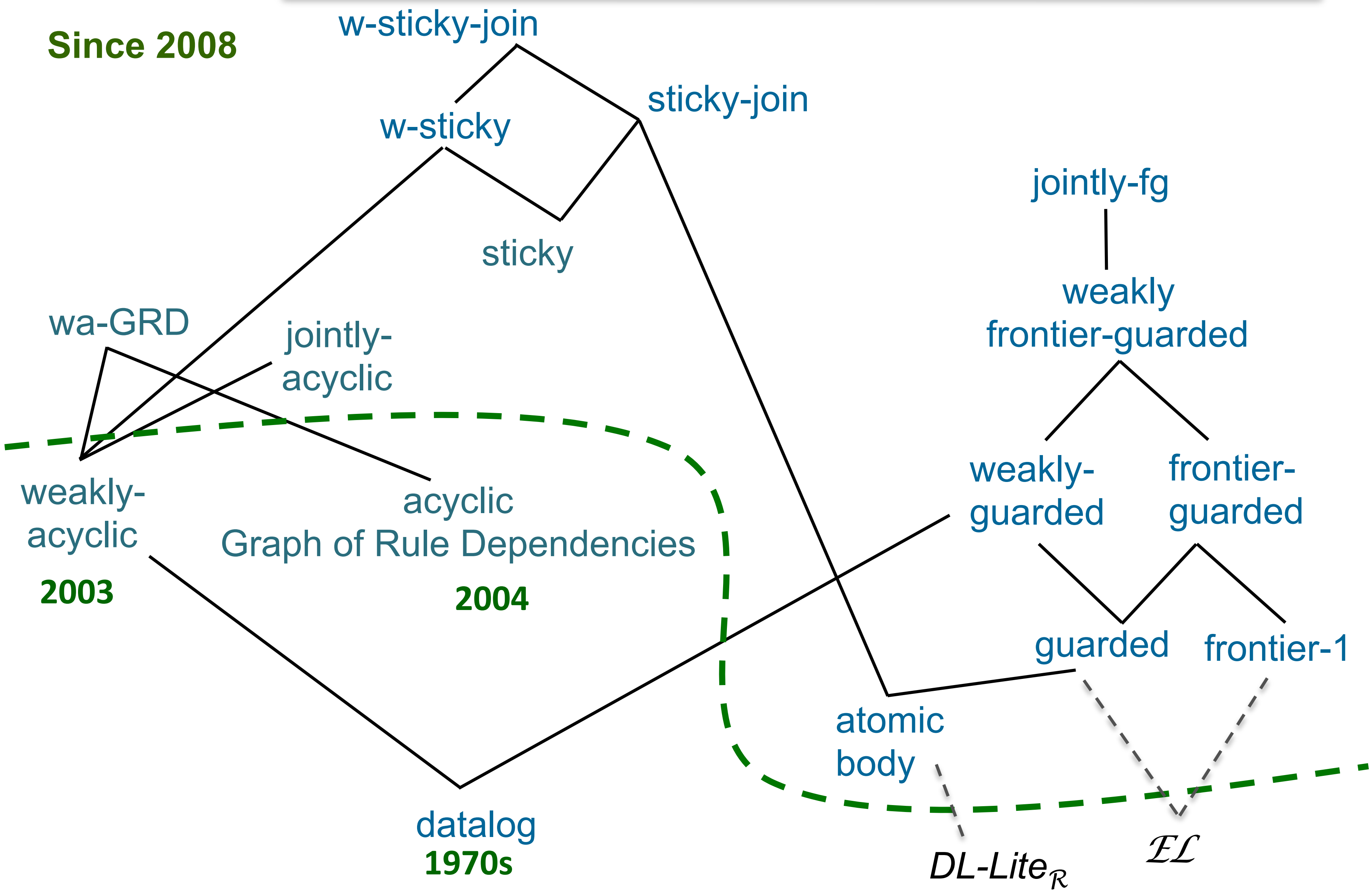
- This problem is **not decidable**

  f.i. [Beeri Vardi ICALP 1981] on TGDs

  even with a single rule [Baget & al. KR 2010]

→ find « decidable » classes of rules
with good expressivity/tractability tradeoff

(PARTIAL) MAP OF DECIDABLE CLASSES



FUNDAMENTAL NOTIONS FOR REASONING IN **FOL(∃,∧)**

- Back to the **positive conjunctive existential fragment** of FOL: **FOL(∃,∧)**
- Allows to express **facts** and (Boolean**)** **conjunctive queries**
- Such formulas can be seen as sets of atoms, labelled graphs, relational structures, ...
- **Homomorphism** is a fundamental notion in this fragment:
  - An interpretation I is a model of a sentence f iff there is a homomorphism from f to I
  - One can define homomorphisms between interpretations. Then:
    If I_1 maps to I_2 then, for any f, I_1 model of $f \Rightarrow I_2$ model of f
  - To a formula f, we assign its isomorphic model *M(f)* (aka canonical model)

MODEL ISOMORPHIC TO A FOL($\exists$,$\wedge$) FORMULA

To a formula f in FOL($\exists$,$\wedge$), we assign its **isomorphic model *M(f)***
also called **canonical model**

$$f = \exists x \exists y \exists z \; (p(x,y) \wedge p(y,z) \wedge r(x,z,a))$$

M(f): $D = \{dx, dy, dz, a\}$

$p^{M(f)} = \{ (dx,dy), (dy,dz) \}$

$r^{M(f)} = \{ (dx, dz, da) \}$

The canonical model *M(f)* is **universal**: for all *M'* model of *f*, *M(f)* maps to *M'*

for any f and g in FOL($\exists$,$\wedge$), $\boldsymbol{g \models f}$ iff *M(g)* is a model of f iff f maps to g

ADDING RANGE-RESTRICTED (= DATALOG) RULES TO FACTS

$\mathcal{K} = (F, \mathcal{R})$ where

$\mathcal{R}$ is a set of **range-restricted rules** (i.e., var(head) $\subseteq$ var(body))

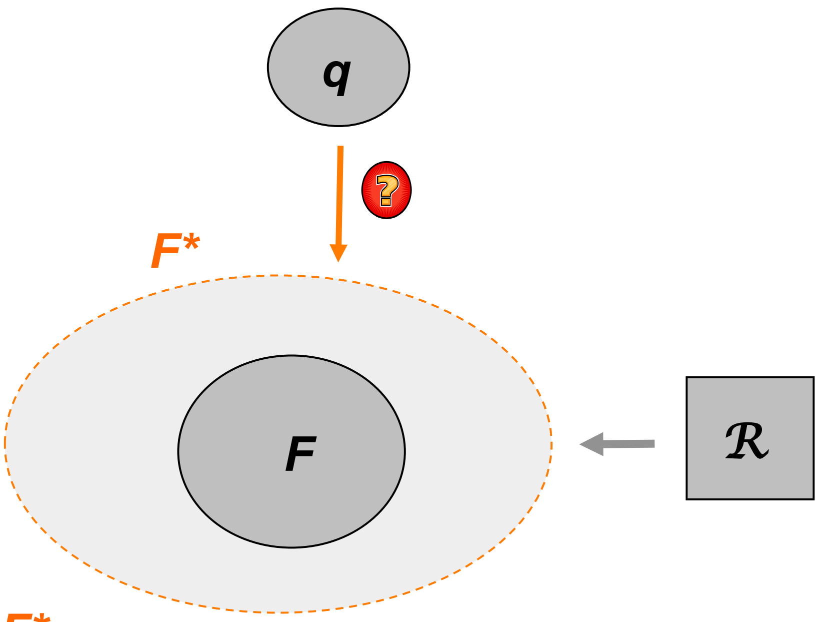
F is a factbase (rules with an empty body): ground atoms

By applying rules from $\mathcal{R}$ starting from F, a unique result is obtained:

the **saturation** of F (denoted by F^*)

F^* is finite since no new variable is created

F^* is a core (no redundancies)



The nice properties of FOL($\exists$, $\wedge$) are kept:

F^* is a universal model of $\mathcal{K}$

Hence: for any CQ q, $\mathcal{K} \models q$ **iff q maps to F^***

KNOWLEDGE BASES WITH EXISTENTIAL RULES

$\mathcal{K} = (F, \mathcal{R})$ where

$\mathcal{R}$ is a set of **existential rules**

F is a factbase (rules with an empty body): existential conjunctions of atoms

Main change: *F** can be **infinite**

R = person(x) → ∃ y hasParent(x,y) ∧ person(y)

F = person(a)

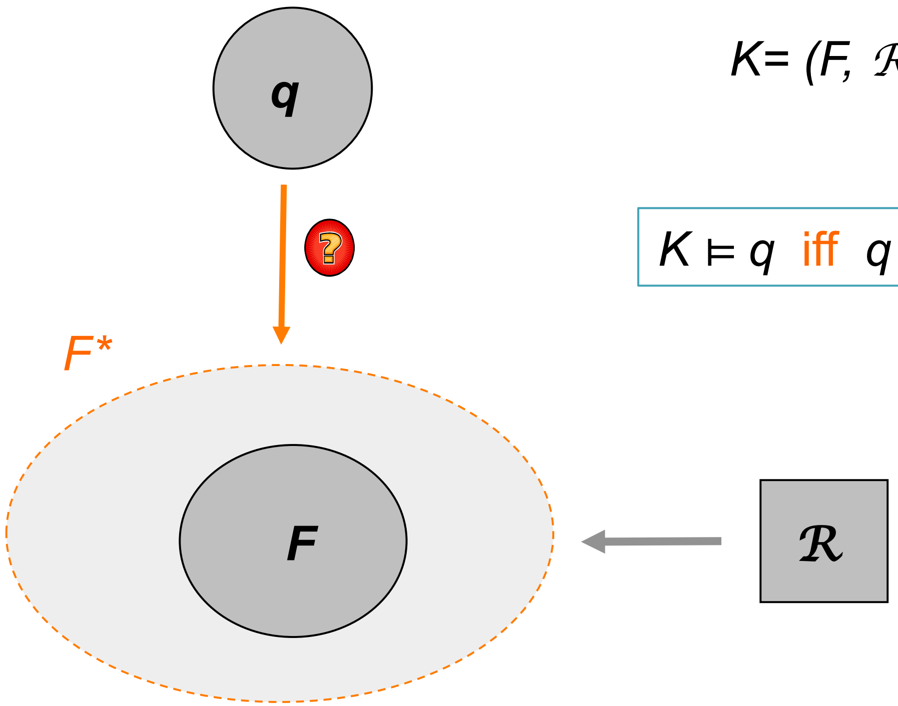
∧ person(y0) ∧ hasParent(a, y0)

∧ person(y1) ∧ hasParent(y0, y1) Etc.

but it remains a **universal model**

hence **$\mathcal{K} \models q$ iff *q* maps to *F****

Approach 1 to Rules : Forward chaining / Materialisation



$K = (F, \mathcal{R})$

« bottom-up »
« chase » (TGDs)

$K \vDash q$ iff q maps by homomorphism to F^*

Pros: materialisation offline, then online query answering is fast

Cons: volume of the materialisation
needs writing access rights to the data
not feasible if data is distributed among several databases
not adapted if data change frequently

EXAMPLE (MATERIALIZATION)

$\forall x\ (\text{movieActor}(x) \rightarrow \exists z\ (\text{movie}(z) \wedge \text{play}(x,z)))$

movie(m1)
movie(m2)
movie(x0)
movieActor(a)
movieActor(b)
play(a,m1)
play(a,m2)
play(c,x0)

movie(*z0*)
play(b,*z0*)

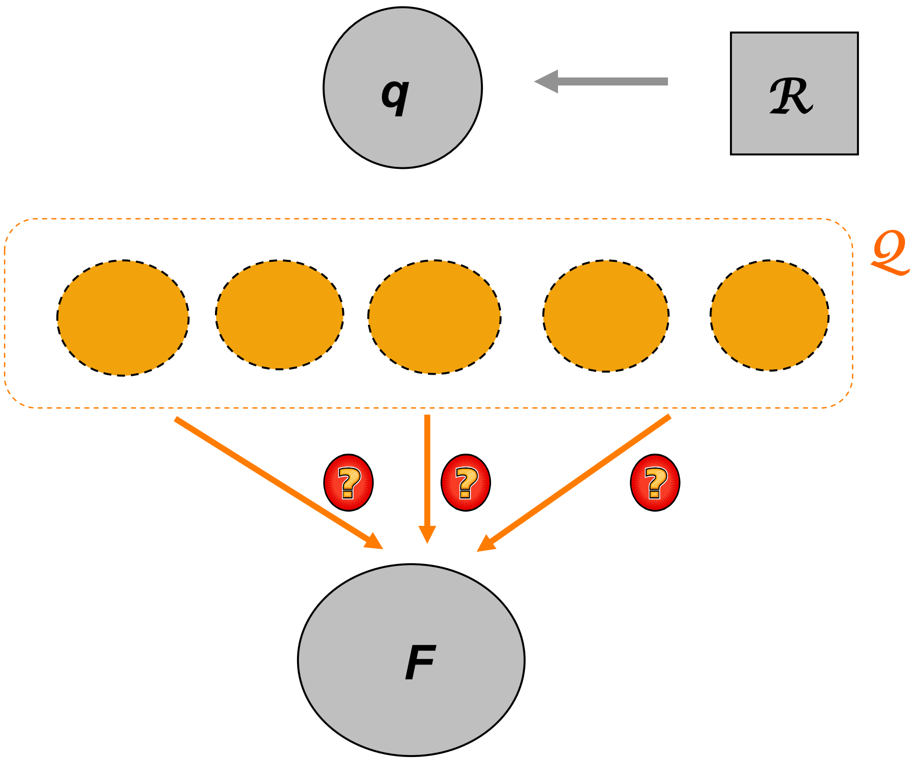
Saturation

$q(x) = \exists y\ (\text{movie}(y) \wedge \text{play}(x, y))$

« find those who play in a movie »

x = a	y = m1
x = a	y = m2
x = b	y = z0
x = c	y = x0

Approach 2 to Rules : Backward Chaining / Query Rewriting



$K = (F, \mathcal{R})$

« top-down » decomposition into 2 steps [DL-Lite]

Rewriting into a set of CQs, seen as a **union of conjunctive queries (UCQ)**

and more generally into a « first-order » query (core SQL query)

Query rewriting is independant from any factbase. For **any** F**,**

$F, \mathcal{R} \vDash q$ iff $F \vDash \mathcal{Q}$ (i.e., if $\mathcal{Q}$ is a UCQ: there is $q_i \in \mathcal{Q}$ with $F \vDash q_i$)

Pros: independent from the data
Cons: rewriting done at query time, easily leads to huge and unusual queries

EXAMPLE

$\forall x\,(\text{movieActor}(x) \rightarrow \exists z\,(\text{movie}(z) \wedge \text{play}(x,z)))$

movie(m1)
movie(m2)
movie(x0)
movieActor(a)
movieActor(b)
play(a,m1)
play(a,m2)
play(c,x0)

$q(x) = \exists y\,(\text{movie}(y) \wedge \text{play}(x, y))$

« find those who play in a movie »

$\text{Rew}_q(x) = \exists y\,(\text{movie}(y) \wedge \text{play}(x, y)) \vee \text{movieActor}(x)$

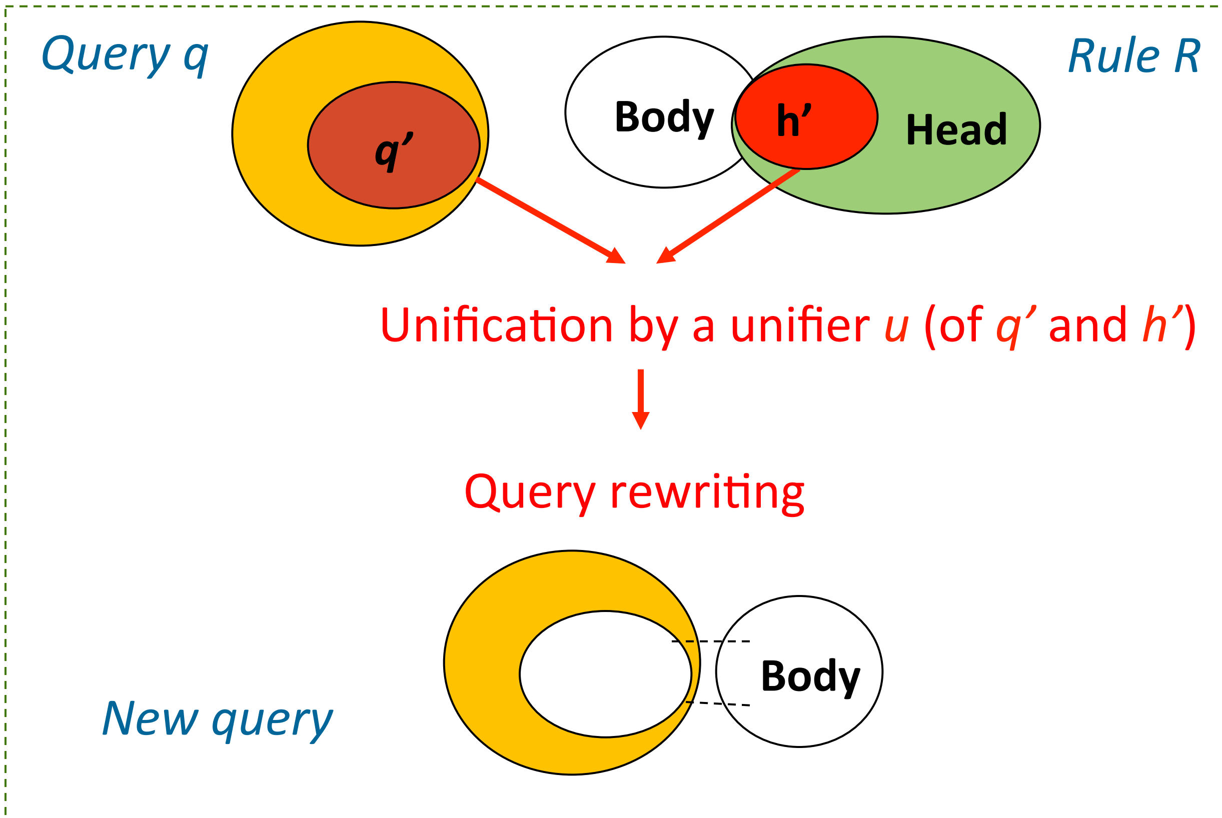
Query rewriting

x = a	y = m1
x = a	y = m2
x = c	y = x0

x = a
x = b

BACKWARD CHAINING SCHEME

Basic step:

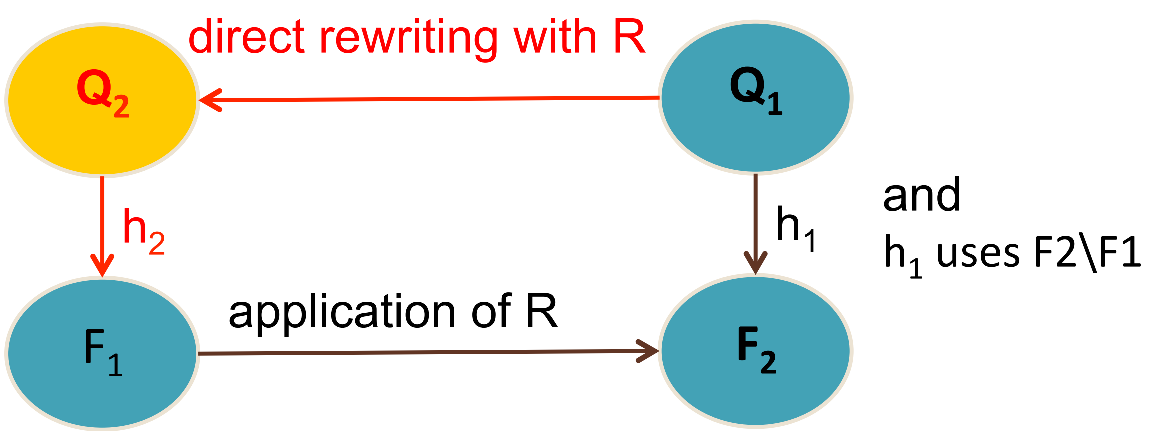


Direct rewriting of q with R and u = $u(q \setminus q') \cup u(body(R))$

Basic Properties (1)

Let F_2 be obtained from F_1 by the application of Rule R
Let a query Q_1 that maps to F_2 by a homomorphism that uses at least one atom brought by R

Then there is Q_2, a direct rewriting of Q_1 with R, such that Q_2 maps to F_1

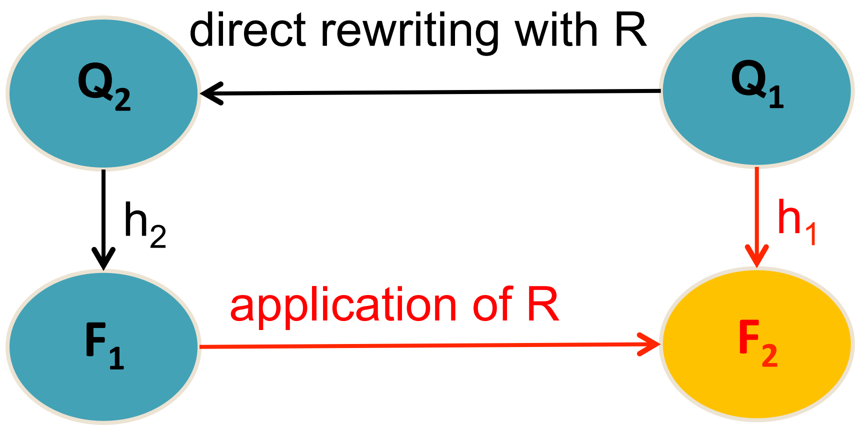


The reciprocal property holds

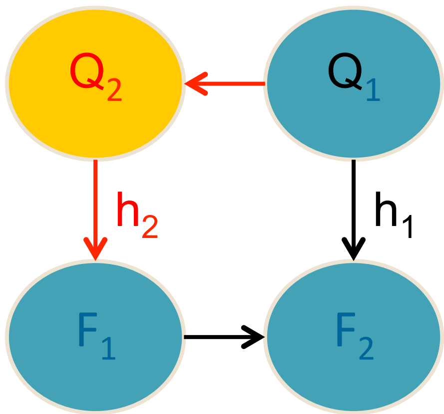
Basic Properties (2)

Let Q_2 be a direct rewriting of Q_1 with Rule R
Let F_1 be a factbase such that Q_2 maps to F_2

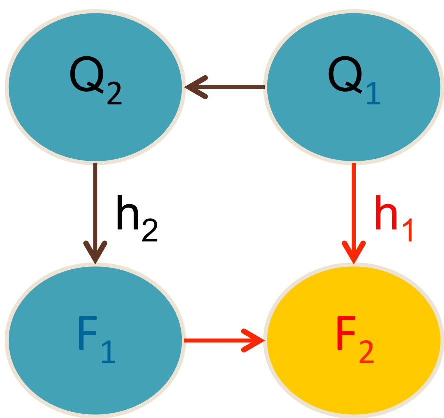
Then there is an application of R to F1 that produces F2 such that Q_2 maps to F_1



Equivalence Derivation / Rewriting Sequences



s.t. h_1 uses $F_2 \setminus F_1$



For any conjunctive query *q*, for any factbase *F*,
for any set of rules:

there is a homomorphism from *q* to *F'*, where *F'* is obtained from *F* by a rule application sequence of length ≤ *n*

iff

there is a homomorphism from *q'* to *F*, where *q'* is obtained from *q* by a rewriting sequence of length ≤ *n*

Taking into account existential variables in rule heads (1)

- We want a complete set of sound rewritings (set of CQs):

  q_i s.t. for any F, if $F \vDash q_i$ then $F, \mathcal{R} \vDash q$

R = person(x) → ∃ y hasParent(x,y)

q = hasParent(v,w), dentist(w)

u = { x ↦ v, y ↦ w }

rew(q,R,u) = q_i = person(v), dentist(w)

q_i is **unsound**:

F = person(Maria), dentist(Giorgos)

$F \vDash q_i$ however *(F,{R})* does not entail q

(1) If *w* in *q* is unified with an existential variable of *R*, then all atoms in which *w* occur must be part of the unification

TAKING INTO ACCOUNT EXISTENTIAL VARIABLES IN RULE HEADS (2)

$R = p(x) \rightarrow \exists z1 \exists z2\ r(x,z1), r(x,z2), s(z1,z2)$

$q = r(v,w), s(w,w)$

$u = \{x \mapsto v, z1 \mapsto w, z2 \mapsto w\}$

$rew(q,R,u) = q_i = p(v)$

q_i is **unsound**:

$F = p(a)$

$F \models q_i$ however *(F,{R})* does not entail *q*

(2) An existential variable of *R* cannot be unified with another term in head(*R)*

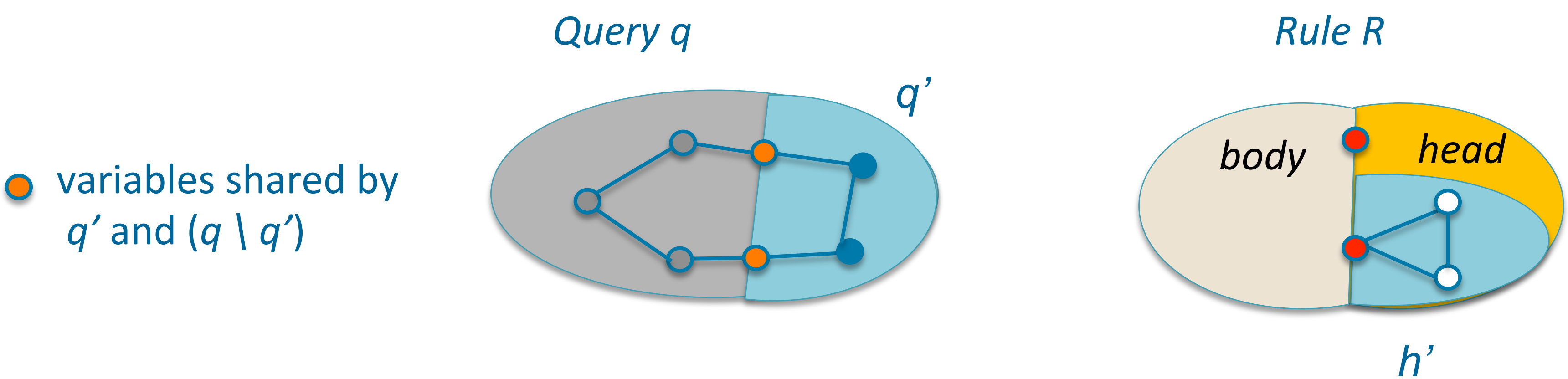
PIECE-UNIFIER (FOR BOOLEAN CQS)

A **piece-unifier** ***u*** of ***q'*** $\subseteq$ *q* and **h'** $\subseteq$ head(R)

is a substitution of *var(q' + h')* by *terms(q'+ h')* [if x is unchanged, we write u(x) = x]

such that :

- *u(q') = u(h')*
- existential variables of *h'* are unified only with variables of *q'* that do not occur in (q \ q')
  (i.e., if *x* is existential and *u(x) = u(t)*, then *t* is a variable of *q'* and not of (q \ q'))



To extend the notion to general CQs:
universal variables cannot be unified with answer variables

EXAMPLE

R = twin(x,y) → ∃ z motherOf(z,x) ∧ motherOf(z,y)

q = motherOf(v,w) ∧ motherOf(v,t) ∧ Female(w) ∧ Male(t) ?

R = twin(x,y) → ∃ z motherOf(z,x) ∧ motherOf(z,y)

q = motherOf(v,w) ∧ motherOf(v,t) ∧ Female(w) ∧ Male(t) ?

piece-unifier $u_2 = \{z \mapsto v, x \mapsto w, y \mapsto t\}$

rewrite(q,R,u_2) = twin(w,t) ∧ Female(w) ∧ Male(t)

R = twin(x,y) → ∃ z motherOf(z,x) ∧ motherOf(z,y)

q = motherOf(v,w) ∧ motherOf(v,t) ∧ Female(w) ∧ Male(t) ?

piece-unifier $u_1 = \{z \mapsto v, x \mapsto w, y \mapsto w\}$

rewrite(q,R,u_1) = motherOf(v,t) ∧ Female(w) ∧ Male(w) ∧ twin(w,w)

If we rewrite again this query we could remove the first atom

What if we skolemized Rules?

R = person(x) → ∃y hasParent(x,y)
q = hasParent(v,w), dentist(w)
u = { x ↦ v, y ↦ w }
rew(q,R,u) = q_i = person(v), dentist(w)

q_i is **unsound**:
F = person(Maria), dentist(Giorgos)
$F \vDash q_i$ however $(F,\{R\})$ does not entail q

Skolem(R) = person(x) → hasParent(x,f(x))

Classical most general unifier of hasParent(x,f(x)) and hasParent(v,w): v ↦ x and w ↦ f(x)

rew(q,R,u) = dentist(f(x)) ∧ person (x) which cannot be unified with a rule head
(would not be kept in the ouput since it contains a skolem function

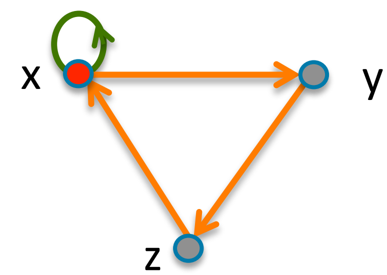
We could skolemize the rules and rely on usual m.g.u.
then keep only rewritings without skolem function
but this would create useless intermediate rewritings

WHY « PIECES »?

A **piece** is a unit of knowledge brought by a rule:

- **Frontier variables** (and constants) act as **cutpoints** to decompose rule heads into pieces (« minimal non-empty subsets glued by existential variables »)

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



- A rule with *k* pieces can be decomposed into *k* rules, one for each piece, while keeping the same body

$$b(x) \rightarrow \exists y \exists z\ p(x,y) \wedge p(y,z) \wedge p(z,x)$$

$$b(x) \rightarrow q(x,x)$$

- It cannot be further decomposed (except by introducing new predicates)

DECOMPOSITION OF RULES INTO ATOMIC HEAD RULES (1)

$R: \quad b(x) \rightarrow \exists y\ \exists z\ p(x,y) \wedge p(y,z) \wedge p(z,x)$

rule with single-piece head

Decomposition into rules with atomic head by introducing a fresh predicate

$R_0: \quad b(x) \rightarrow \exists y\ \exists z\ p_R(x,y,z)$

$R_1: \quad p_R(x,y,z) \rightarrow p(x,y)$

$R_2: \quad p_R(x,y,z) \rightarrow p(y,z)$

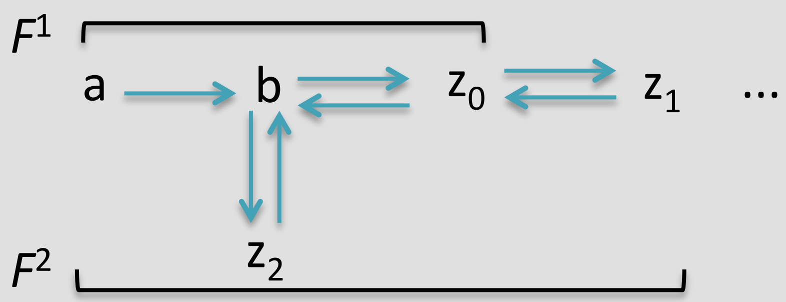
$R_3: \quad p_R(x,y,z) \rightarrow p(z,x)$

We lose the structure of the head

- much less efficient query rewriting
- may even lead to lose the property of having a finite universal model (if the set of rules has this property)

DECOMPOSITION OF RULES INTO ATOMIC HEAD RULES (2)

F : p(a,b) R : p(x,y) → ∃ z p(y,z), p(z,y)



$F^2 \equiv F^1$ (F^2 maps to F^1)

hence $F^* \equiv F^1$

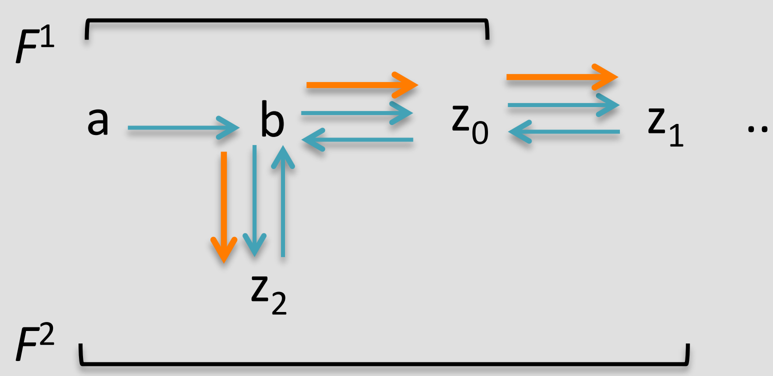
Finite universal model

After decomposition into atomic head rules:

R_0 : p(x,y) → ∃ z $p_R(y,z)$

R_1 : $p_R(y,z)$ → p(y,z)

R_2 : $p_R(y,z)$ → p(z,y)



$F^2 \not\equiv F^1$

No finite universal model

OVERVIEW OF THE LECTURE

Part 1: Basics

Part 2: KR formalisms and algorithmic approaches

Part 3: Decidability issues in the existential rule framework

Undecidability of the fundamental problem

Generic properties that ensure decidability

Main « concrete » decidable classes of existential rules

SATURATION MAY NOT HALT

R = person(x) → hasParent(x,y) ∧ person(y)

F = person(a)

∧ person(y0) ∧ hasParent(a, y0)

∧ person(y1) ∧ hasParent(y0, y1)

No redundancies are added
The KB has no finite universal model

However, here: query rewriting with R is finite for any q

QUERY REWRITING MAY NOT HALT

R = friend(u,v) ∧ friend(v,w) → friend(u,w)

q = friend(Giorgos,Maria)

q_1 = friend(Giorgos, v0) ∧ friend (v0,Maria)

q_2 = friend(Giorgos, v1) ∧ friend(v1, v0) ∧ friend (v0,Maria)

*$q_{2'}$ = friend(*Giorgos*, v0) ∧ friend(v0, v1) ∧ friend (v1,Maria)*

q_2 and $q_{2'}$ are equivalent

q_3 = friend(Giorgos, v2) ∧ friend(v2, v1) ∧ friend(v1, v0) ∧ friend (v1,Maria)

Etc.

There is an infinite number of non-redundant rewritings

However, here: saturation with *R* is finite for any *F*

There are cases where both processes do not halt (even if the factbase is known)

UNDECIDABILITY OF THE FUNDAMENTAL PROBLEM

Fundamental decision problem
Input: $K= (F, \mathcal{R})$ knowledge base, q Boolean conjunctive query
Question: is q entailed by K ?

This problem is **undecidable** (only semi-decidable)

E.g. proof by **reduction from the word problem in a semi-Thue system**

Input: a set G of rules of the form $w_i \rightarrow w_j$, 2 words w_0 and w_f

Question: is it possible to derive (exactly) w_f from w_0 using the rules in G?

There is a *one-step derivation* from a word w to w' if
there is a rule $w_i \rightarrow w_j$ in G, and $w = w_1 w_i w_2$, $w' = w_1 w_j w_2$

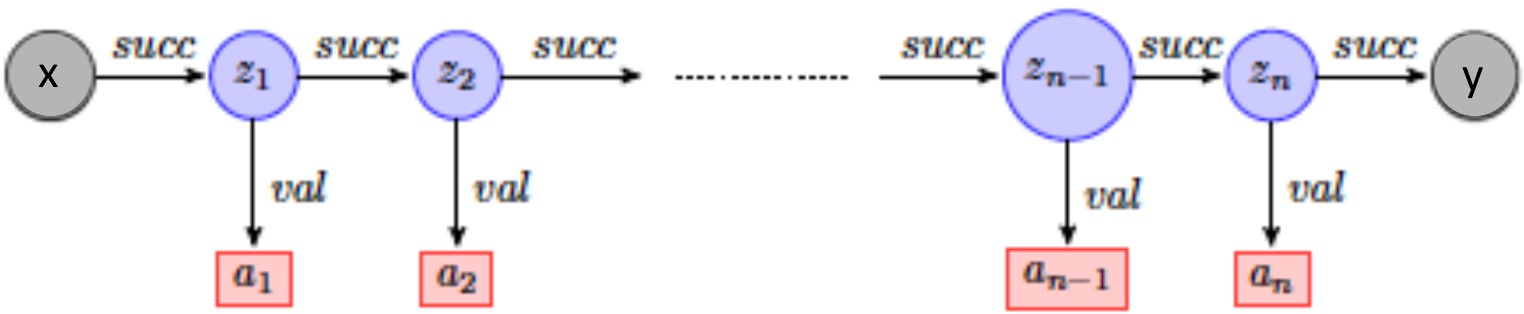
w' is *derived from* w if
there is a (finite) sequence of one-step derivations from w to w'

REDUCTION FROM THE WORD PROBLEM

From G, w_0 and w_f we build a KB (F, $\mathcal{R}$) and a Boolean CQ q

Vocabulary constants: the letters occuring in G, w_0 and w_f
+ two special constants B and E
binary predicates: succ and val

To a word w = $a_1 \ldots a_n$ we assign the following graph T(w,x,y)
where the z_i are existential variables and x,y are free

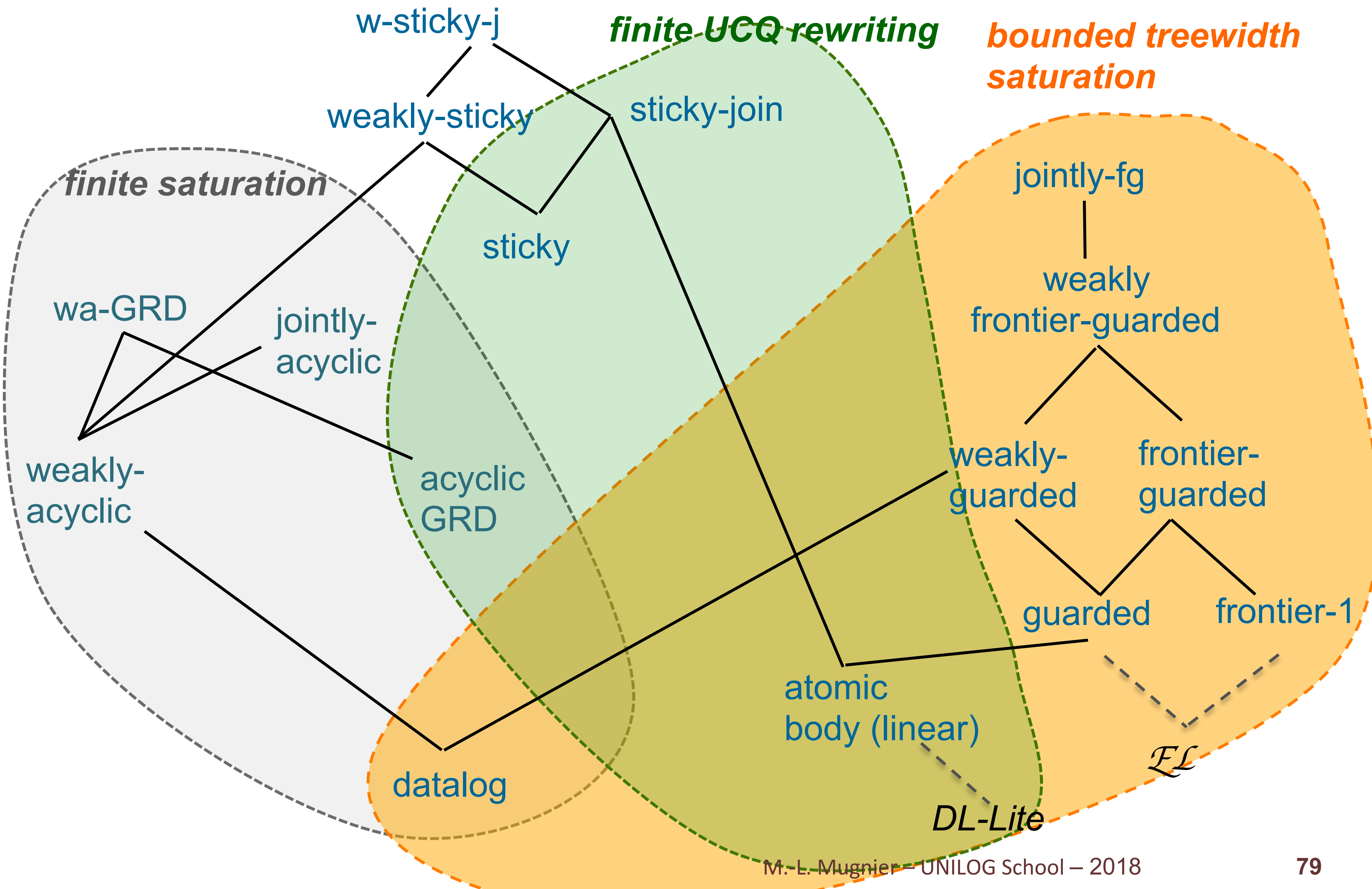


Factbase F = T(w_0, B, E) **Query** q = T(w_f, B, E)

Set of rules $\mathcal{R}$ is obtained by translating each rule $w_i \rightarrow w_j$ into the existential rule
$\forall x \ \forall y \ (T(w_i,x,y) \rightarrow T(w_j,x,y))$

Key: any word w derivable from w_0 with G corresponds to a path T(w, B, E) in the saturation of F by R, and reciprocally

(PARTIAL) MAP OF DECIDABLE CASES



Generic Properties that ensure decidability

Three generic kinds of properties ensuring decidability:

- Saturation by Forward Chaining halts for any factbase
  (« finite expansion set », *fes*)

- Query rewriting halts for any conjunctive query
  (« finite unification set », *fus*, or UCQ-rewritability)

- Saturation by Forward Chaining may not halt *but* for any factbase
  the generated facts have a tree-like structure (« bounded treewidth set », *bts*)

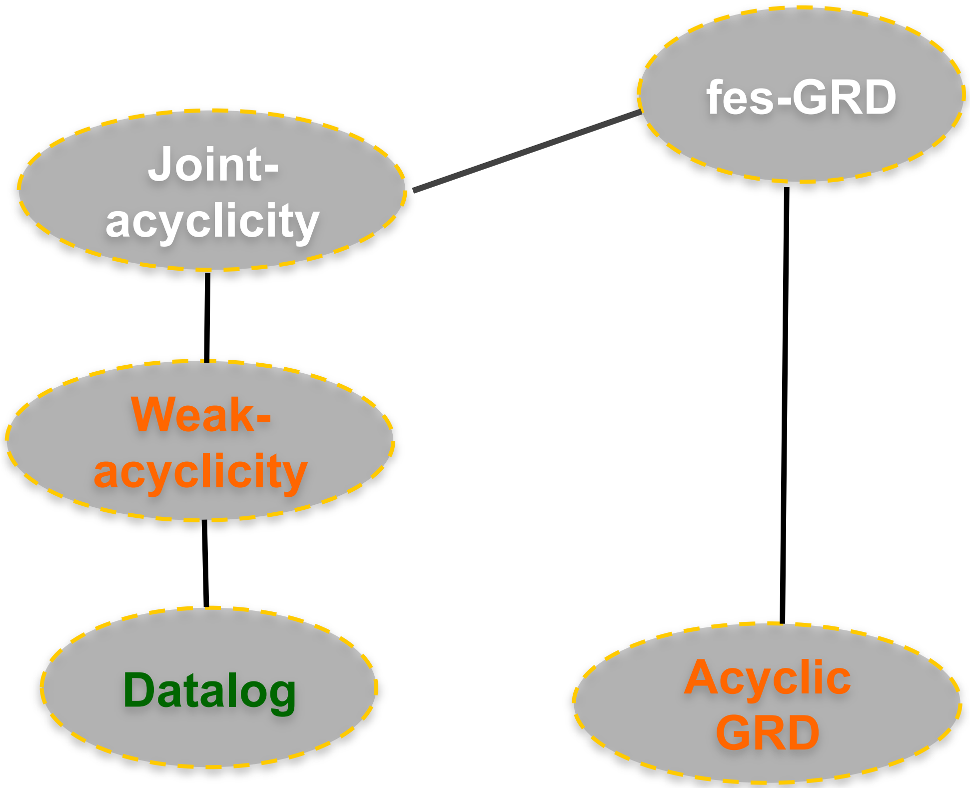
None of these properties is *recognizable* [Baget+ KR 10]

but these properties provide *generic* algorithmic schemes

Main Classes with Finite Saturation *(fes)*

Acyclic *existential dependency* graph
[Krötzsch+ IJCAI' 11]

Acyclic *position dependency* graph
[Deutsch+ ICDT' 03]
[Fagin+ ICDT 03]



GRD with *fes* strongly connected components
[Baget KR' 04]

Acyclic Graph of *Rule Dependencies*
[Baget KR' 04]

No existential variables

Position dependency graph: nodes are positions in predicates
edges show how existential variables are propagated

Graph of rule dependencies: nodes are rules
edges express that a rule may lead to trigger a rule

WEAK-ACYCLICITY

Position dependency graph

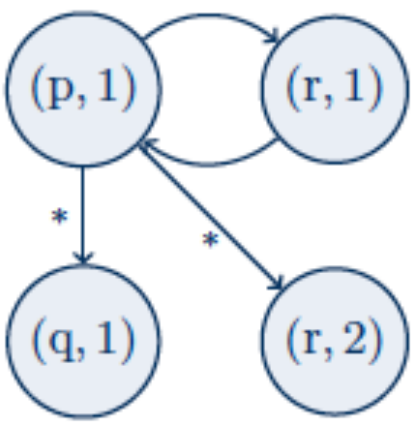
nodes: positions (p,i) in predicates

edges: for each frontier variable x in position (p,i) in a rule body

- an edge from (p,i) to each position (q,j) of x in the rule head
- a special edge from (p,i) to each position of an existential in the rule head

$\mathcal{R}$ is **weakly-acyclic** if its position graph contains no circuit with a special edge (*)

R_1: $p(x) \rightarrow \exists y\ r(x,y) \wedge q(y)$
R_2: $r(x,y) \rightarrow p(x)$



weakly acyclic

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

not weakly acyclic

special edge (p,1) → (r,1) due to R_1
edge (r,1) → (p,1) due to R_2

Acyclic Graph of Rule Dependency

Graph of Rule Dependencies

nodes: the rules

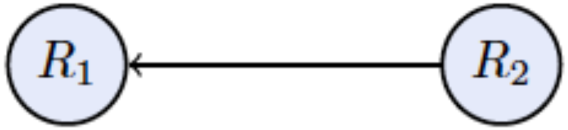
edges: an edge from R_i to R_j if an application of R_i may lead to trigger a new application of R_j (« R_j depends on R_i »)

Dependency can be effectively computed by checking if there is a piece-unifier of body(R_j) and head(R_i)

R_1: $p(x) \rightarrow \exists y\ r(x,y) \wedge q(y)$
R_2: $r(x,y) \rightarrow p(x)$

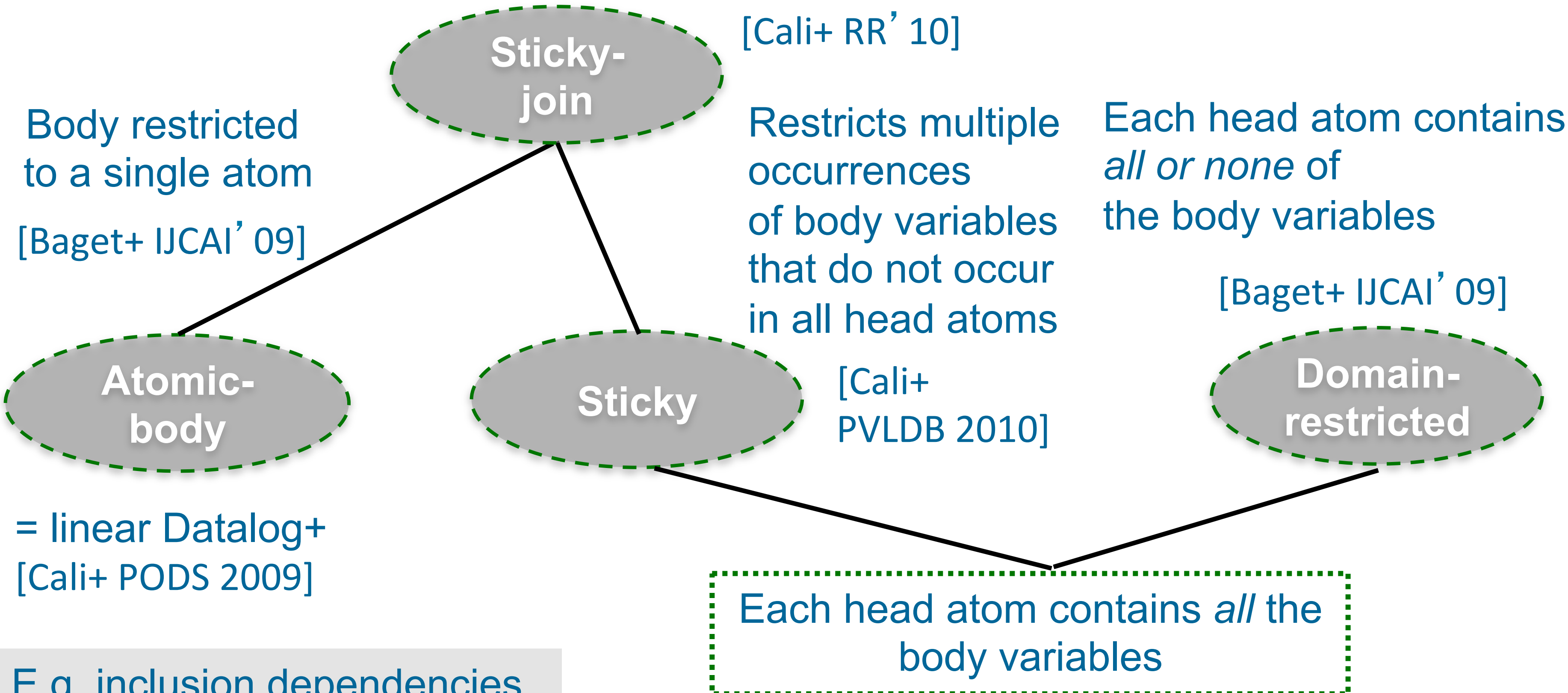
Cyclic GRD since R_1 and R_2 depend on each other

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



These examples show that weak-acyclicity and acyclic GRD are incomparable criteria
Common generalizations of these two notions have been defined

Main Classes with Finite Query Rewriting *(fus)*



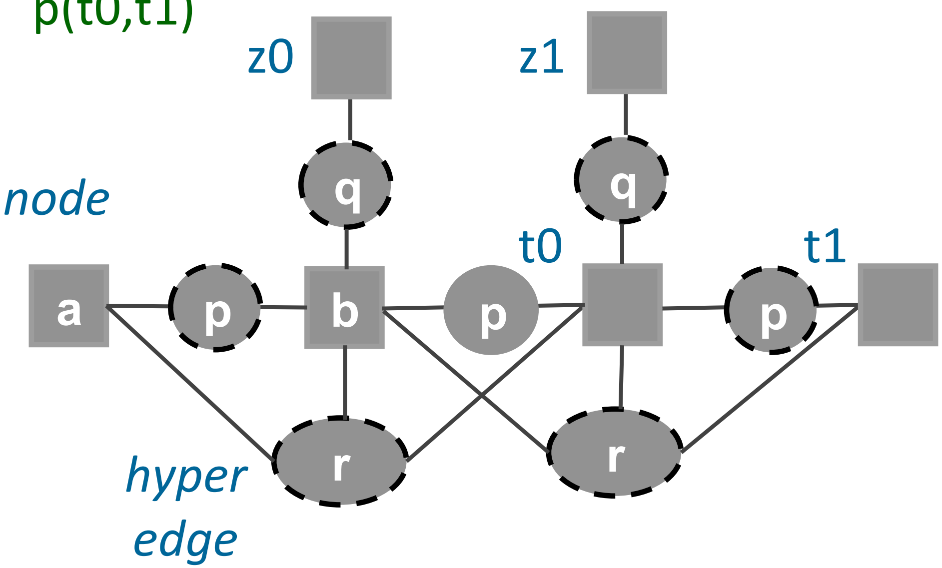
E.g. inclusion dependencies, necessary properties of concepts / relations

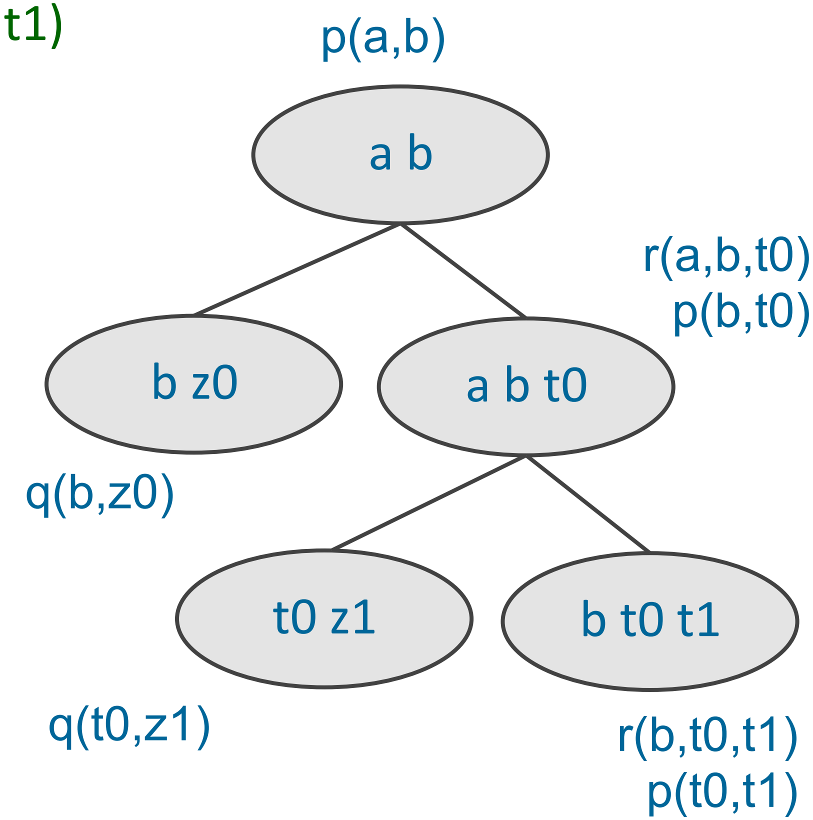
E.g. *concept product*
Elephant(x) ∧ Mouse(y) → bigger-than(x,y)

E.g. Human(x) → parentOf(y,x) ∧ Human(y)
is atomic-body, sticky and domain-restricted

Decomposition Tree / Treewidth

p(a,b) q(b,z0) r(a,b,t0) p(b,t0) q(t0,z1) r(b,t0,t1) p(t0,t1)





Decomposition tree

1) each node *(term)* appears in a bag

2) each hyperedge *(atom)* has all its nodes in a bag

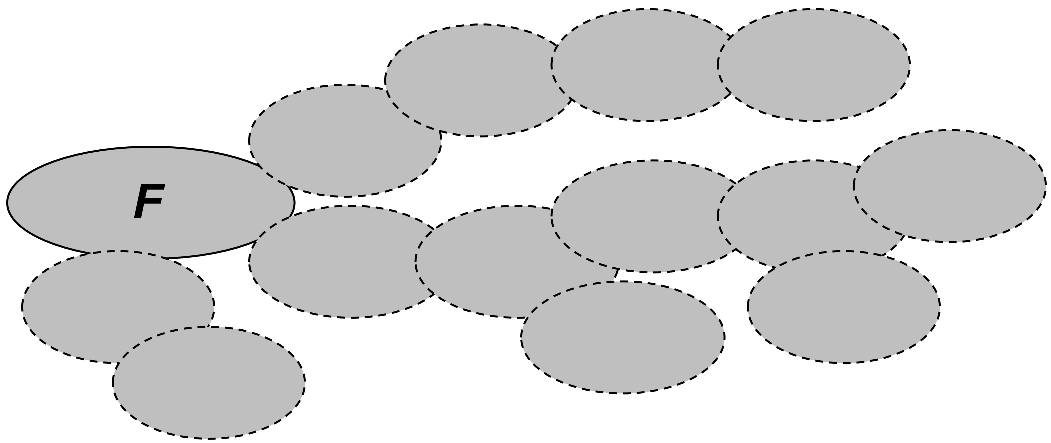
3) for each node *x*, the subgraph induced by the bags containing *x* is connected

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 the forward chaining with $\mathcal{R}$ generates facts with bounded treewidth:
i.e., for any factbase *F*, there is an integer *b* s.t.
any factbase $\mathcal{R}$-derived from *F* has treewidth bounded by *b*

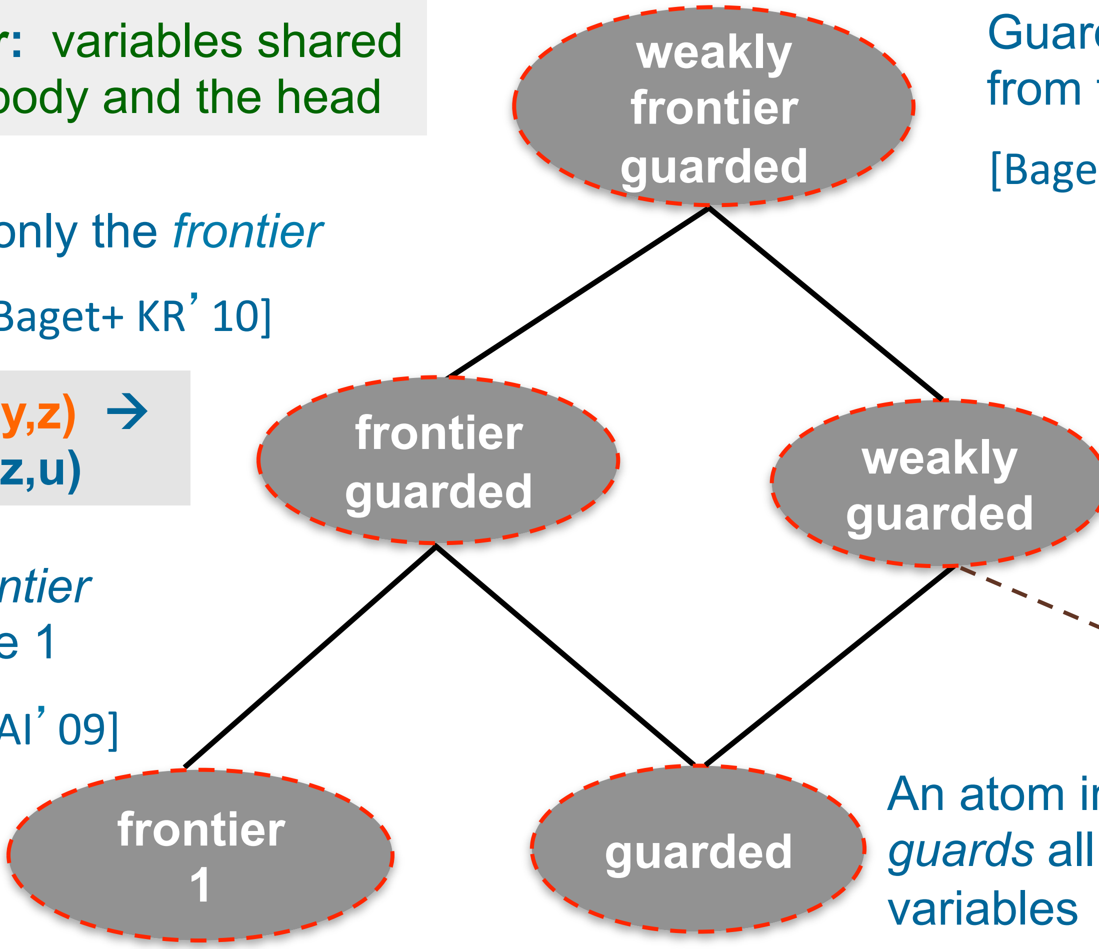


fes (finite saturation) is included in *bts*
(bound given by the number of terms in the finite saturation)

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

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) ∧ r(y,z) → r(y,u) ∧ r(z,u)

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

[Cali+ KR' 08]

datalog

The *frontier* has size 1

[Baget+ IJCAI' 09]

An atom in the body *guards* all the body variables [Cali+ KR' 08]

r(x,y) ∧ r(y,z) ∧ r(x,z) → ∃u r(z,u)

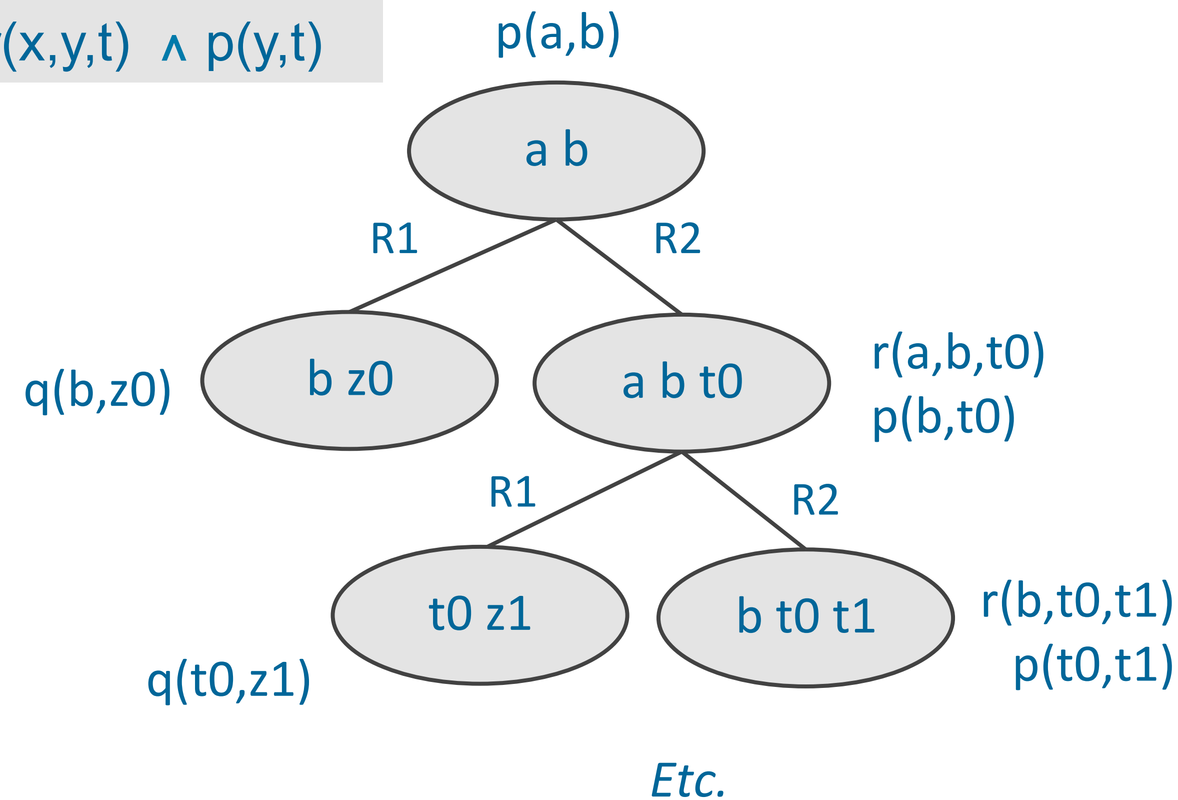
r(x,y) ∧ r(y,z) ∧ s(x,y,z) → ∃u r(y,u) ∧ r(z,u)

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

Greedy *bts*

$R_1 = p(x,y) \rightarrow \exists z\ p(y,z)$
$R_2 = p(x,y) \wedge q(x,z) \rightarrow \exists t\ r(x,y,t) \wedge p(y,t)$

$F = p(a,b)$



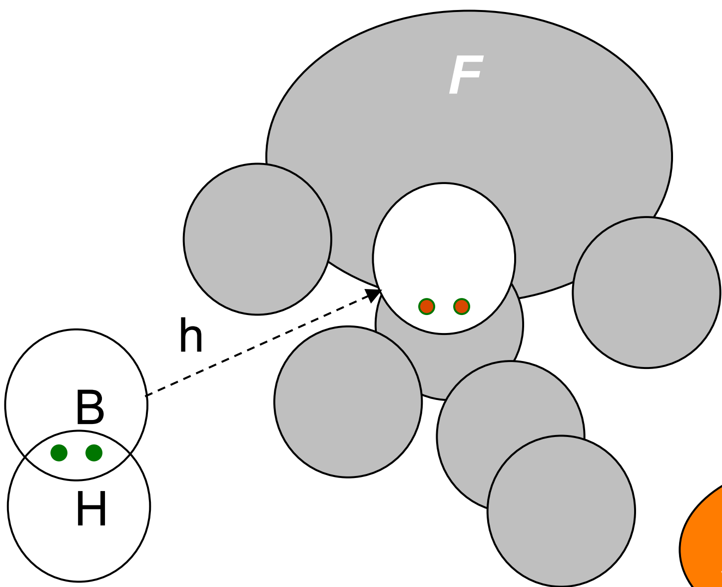
Greedy construction of a decomposition tree of derived facts

with bounded width

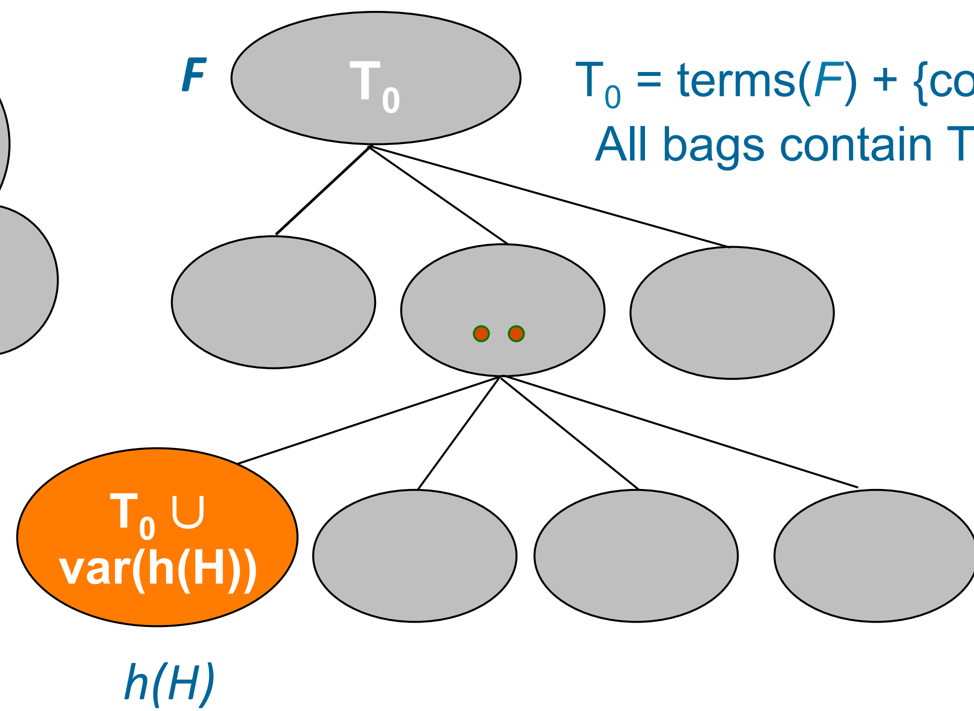
The « Greedy bts » Property [Baget+ IJCAI' 11]

For any factbase, 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

Derived facts



Decomposition tree



T_0 = terms(*F*) + {constants}
All bags contain T0

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 factbase *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* added as a rule « *q* → *match* »

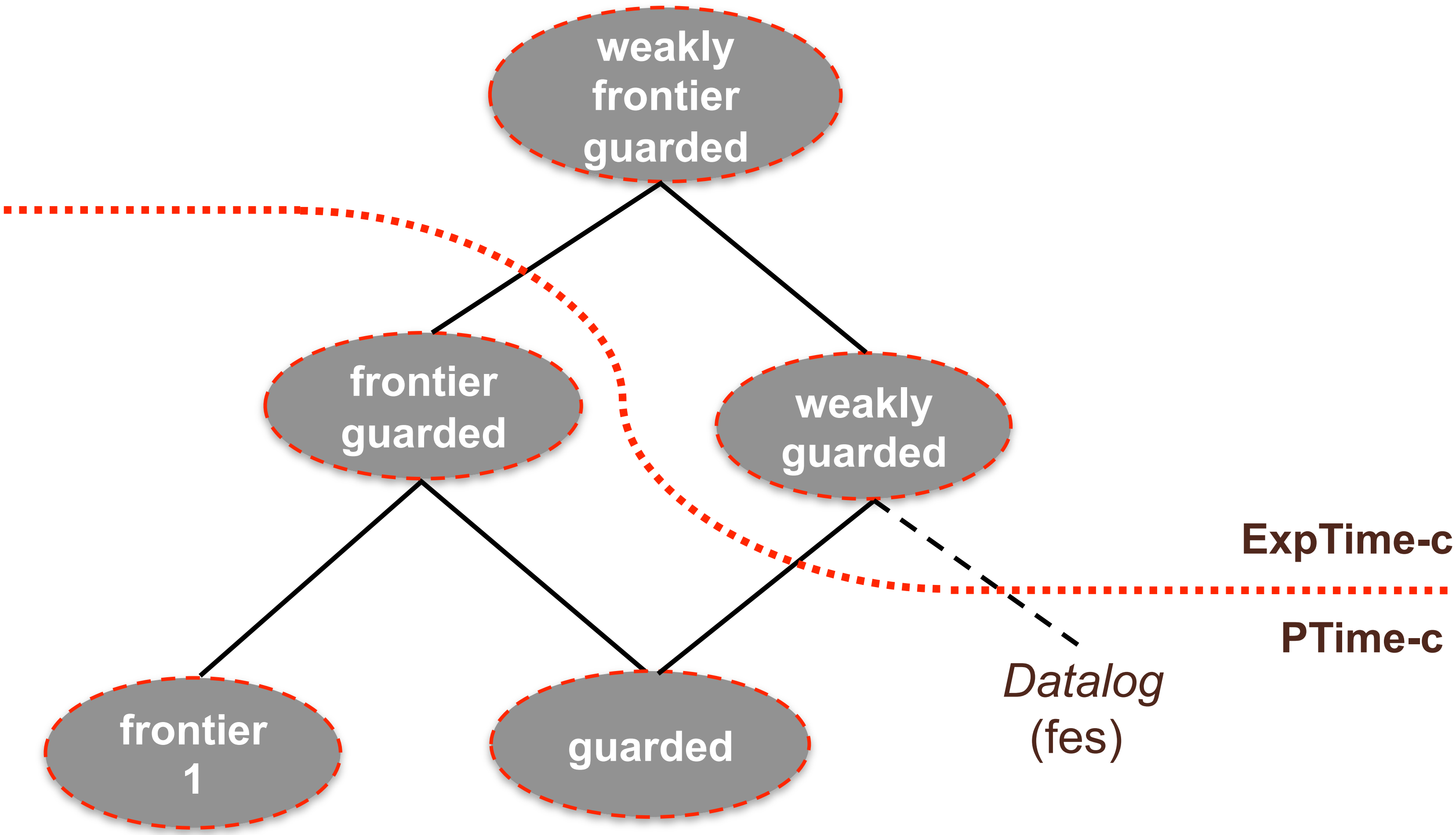
q is entailed iff *match* occurs in a bag pattern
i.e., q maps by homomorphism *to atoms(T*)*

[Thomazo+ KR 2012] offline /online separation

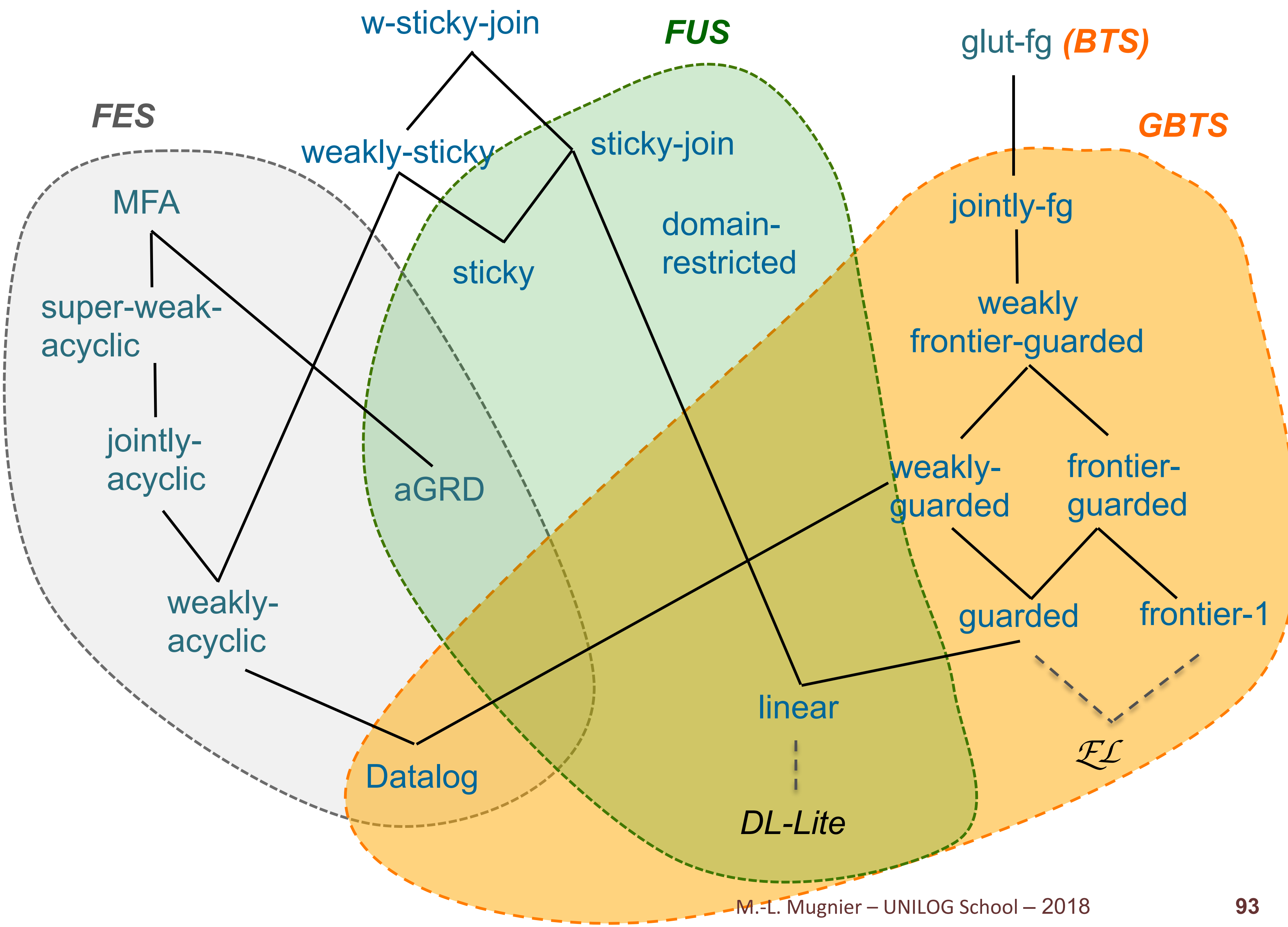
(1) compilation: tree T* built independently from *any* query

(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**

Data Complexity of gbts Classes



Previous algorithm is worst-case optimal on gbts
for data / combined complexity.
Can be specialized to be optimal on these gbts subclasses


w-sticky-join
FUS
glut-fg (BTS)
FES
GBTS
weakly-sticky
sticky-join
MFA
jointly-fg
domain-restricted
sticky
super-weak-acyclic
weakly frontier-guarded
jointly-acyclic
aGRD
weakly-guarded
frontier-guarded
weakly-acyclic
guarded
frontier-1
linear
Datalog
$\mathcal{EL}$
DL-Lite

CONCLUSION

- Reasoning with ontologies is becoming central in many data-centric applications

- Solid theoretical foundations with a range of ontological formalisms that offer various tradeoff expressivity/complexity

- Ongoing research
  - Go beyond (unions of) conjunctive queries, e.g. combine them with navigational queries like regular path queries
  - New query rewriting techniques that target more powerful langages, e.g. Datalog
  - New query answering techniques that combine materialisation and query rewriting
  - Study the interaction of the ontology with mappings, which is key to efficient query answering over heterogeneous data
  - Representing and reasoning with temporal and spatial data
  - Dealing with data inconsistencies ....

(Small) Bibliography

Bienvenu M., Leclère M., Mugnier, M.-L. and Rousset, M.-C., Reasoning with Ontologies, chapter 6, volume 1 in « A guided tour of artificial intelligence research », Springer, to appear.

Introductions to several aspects of ontology-mediated query answering with description logics or existential rules in the Reasoning Web summer school books:

in particular:

Bienvenu, M. and Ortiz, M. (2015). Ontology-mediated query answering with data tractable description logics. 11th International Reasoning Web Summer School , volume 9203 of LNCS , pages 218–307. Springer.

Mugnier, M. and Thomazo, M. (2014). An introduction to ontology-based query answering with existential rules. 10th International Reasoning Web Summer School, volume 8714 of LNCS, pages 245–278. Springer.

Gottlob, G., Orsi, G., Pieris, A., and Simkus, M. (2012). Datalog and its extensions for semantic web databases. 10th International Reasoning Web Summer School ,volume 7487 of LNCS, pages 54–77. Springer.

These syntheses provide further references

Appendix: further details

- Fundamental definitions and properties for the FOL($\exists,\wedge$) fragment
- Piece-unifiers

INTERPRETATIONS / MODELS (1)

- **Vocabulary** $\mathcal{V} = (\mathcal{P}, \mathcal{C})$, where $\mathcal{P}$ = finite set of predicates
  $\mathcal{C}$ = set of constants
- **Interpretation** $\mathcal{I} = (D_{\mathcal{I}}, .^{\mathcal{I}})$ of $\mathcal{V}$, where
  $D_{\mathcal{I}} \neq \emptyset$ (domain)
  for all *c* in C, $c^{\mathcal{I}}$ in $D_{\mathcal{I}}$
  for all *p* in $\mathcal{P}$ with arity *k*, $p^{\mathcal{I}} \subseteq D_{\mathcal{I}}^{k}$
- Furthermore, unique name assumption: for all *c* and *d* in C, $c^{\mathcal{I}} \neq d^{\mathcal{I}}$
- Simplifying assumption (in line with the unique name assumption):
  $\mathcal{C} \subseteq D_{\mathcal{I}}$ and for all c in $\mathcal{C}$, $c^{\mathcal{I}} = c$

$\mathcal{V} = (\{p_{/2}, r_{/3} \}, \{a, b\})$

$\mathcal{I}$: $D_{\mathcal{I}} = \{a, b, d_1\}$ $p^{\mathcal{I}} = \{ (b, a), (b, d_1), (d_1, b) \}$
$r^{\mathcal{I}} = \{ (d_1, d_1, a) \}$

- $\mathcal{I}$ is a **model** of *f* (built on $\mathcal{V}$) if *f* is true in $\mathcal{I}$

INTERPRETATIONS / MODELS (2)

- Let f in FOL($\exists$, $\wedge$). $\mathcal{I}$ is a **model** of f iff

  there is a mapping v from *terms(f)* to $D^{\mathcal{I}}$ such that

  for all $p(e_1, \ldots, e_k)$ in f, $(v(e_1), \ldots, v(e_k))$ in $p^{\mathcal{I}}$

$\mathcal{I}$: $D_{\mathcal{I}} = \{a, b, d_1\}$ $\quad p^{\mathcal{I}} = \{(b, a), (b, d_1), (d_1, b)\}$

$r^{\mathcal{I}} = \{(d_1, d_1, a)\}$

$f = \exists x \exists y \exists z \, (p(x, y) \wedge p(y, z) \wedge r(x, z, a))$

v: $x \mapsto d_1 \quad y \mapsto b \quad z \mapsto d_1$

- Interpretations can be seen as **sets of atoms**

  (with elements from $D \setminus \mathcal{C}$ seen as variables)

$p(b,a),\ p(b,x_1),\ p(x_1,b),\ r(x_1, x_1,a)$

- $\mathcal{I}$ is a **model** of f iff there is a **homomorphism** from f to $\mathcal{I}$

HOMOMORPHISMS AGAIN AND AGAIN

- One can define **homomorphisms** between **interpretations**
- We have:

  If I_1 maps I_2 then, for any f, I_1 model of $f \Rightarrow I_2$ model of f

- To a formula f in FOL($\exists$, $\wedge$), we assign its **isomorphic model *M(f)***
  (also called **canonical model**)

$f = \exists x \exists y \exists z \, (p(x,y) \wedge p(y,z) \wedge r(x,z,a))$

M(f): D = {dx, dy, dz, a}

$p^{M(f)}$ = { (dx,dy), (dy,dz) }

$r^{M(f)}$ = { (dx, dz, da) }

NICE SEMANTIC PROPERTIES OF FOL($\exists$, $\wedge$)

- The canonical model *M(f)* is **universal**, i.e., for all *M'* model of *f*, *M(f)* maps to *M'*

Proof: Let *M'* model of *f*. Then, *f* maps to *M'*. Since *M(f)* isomorphic to *f*, *M(f)* maps to *M'*

- ***g*** $\vDash$ ***f*** (i.e., every model of *g* is a model of *f*) iff

  f maps by homomorphism to ***M(g)*** iff

  f maps by homomorphism to ***g***

Proof:

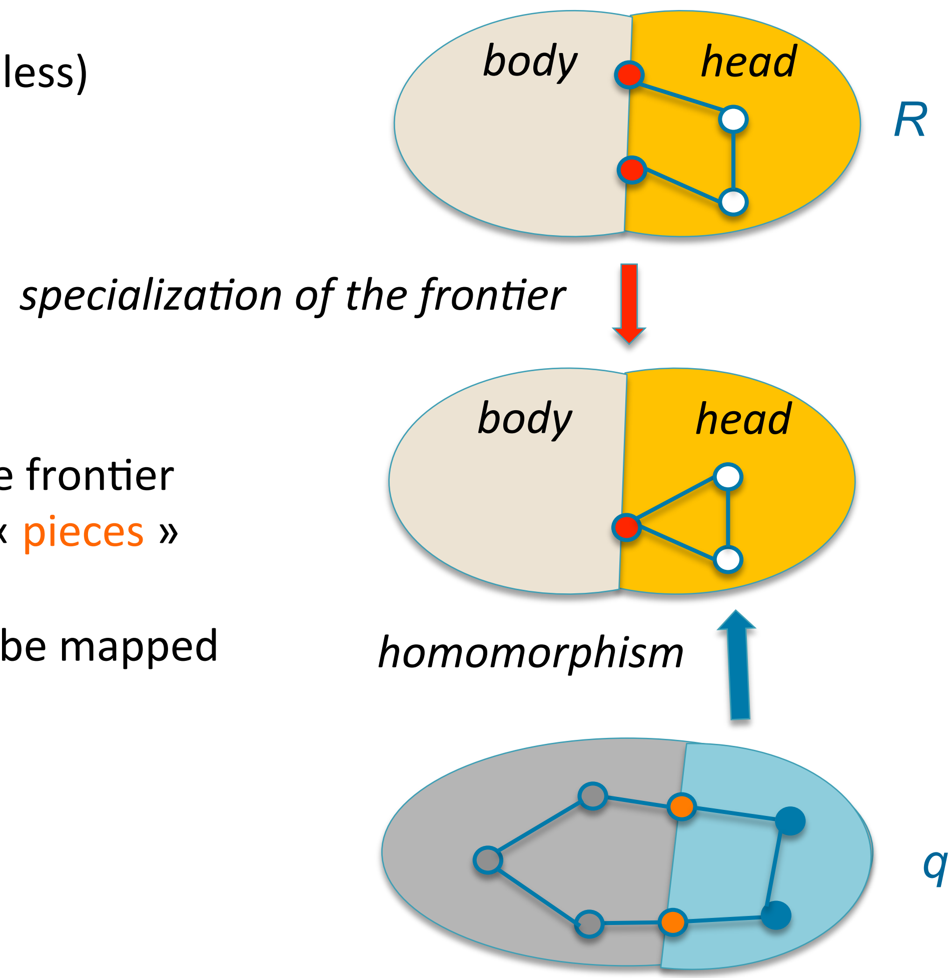
$\Rightarrow$ Assume *g* $\vDash$ *f*. In particular *M(g)* is a model of *f*, hence *f maps to M(g)*

$\Leftarrow$ Assume *f maps to M(g)*. Since *M(g)* is universal: for any *M'* model of *g*, *f maps to M'*, i.e., *M'* is a model of *f*, hence *g* $\vDash$ *f*

Why « pieces » ? (cont'd) – Piece-unifiers

- Unification must « map » parts of *q* according to « pieces » that can be provided by a rule application

  (otherwise it is unsound or useless)



The terms of *q* unified with the frontier (or with constants) cut *q* into « pieces »

⇒ entire « pieces » of *q* must be mapped to the pieces of the rule

PIECE-UNIFIER: ALTERNATIVE DEFINITION

Let $\boldsymbol{u_1}$: frontier(R) → frontier (R) ∪ constants
(u_1 is a specialization of the frontier of R)

Let $\boldsymbol{u_2}$ be a homomorphism from $q' \subseteq q$ to $u_1(head(R))$

Cutpoints: terms of q' mapped to $u_1(frontier(R))$
or to constants

The cutpoints cut q into « **pieces** »

u_1+u_2 is a piece-unifier if q' is composed of *pieces*