TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut Künstliche Intelligenz, Professur Computational Logic

Query Answering under Existential Rules

An Overview

Sebastian Rudolph
TU Dresden

ICCL Summer School 2015

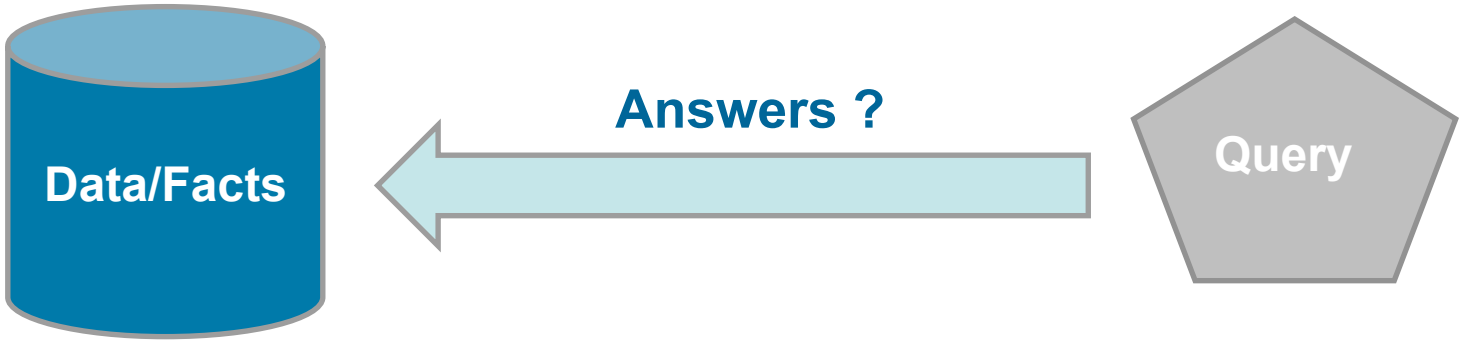
Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Classical Query Answering

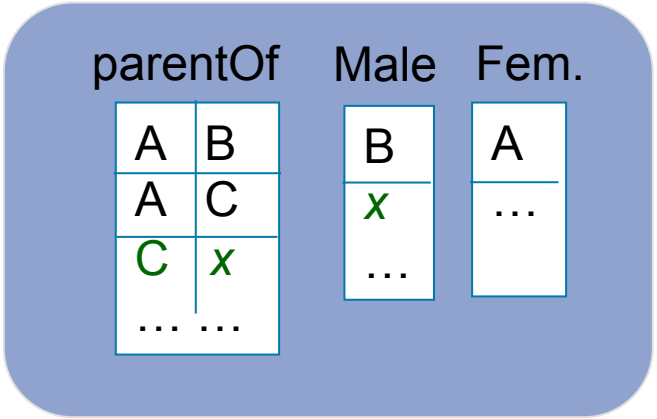


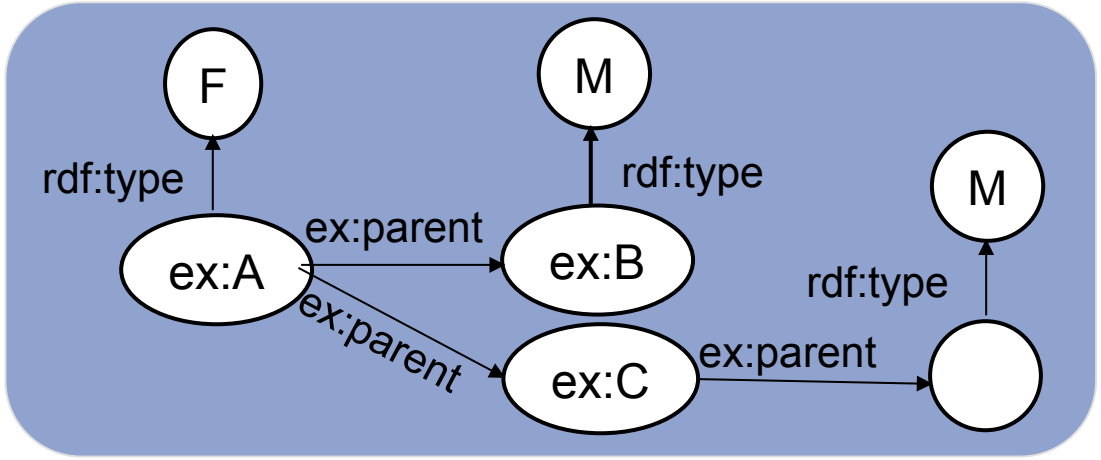
Data / Facts

Relational Database

RDF (Semantic Web)

Etc.

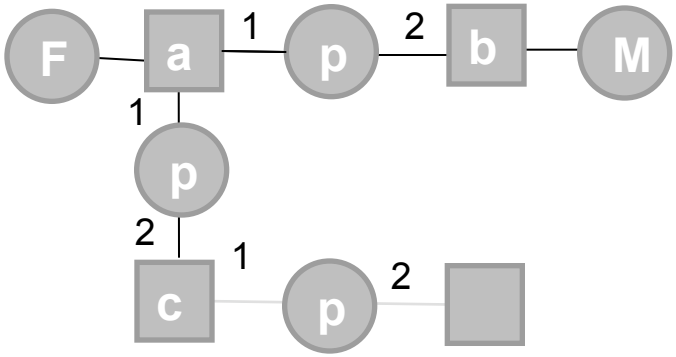




Abstraction in First-Order Logic

$$\exists x(\ \text{parentOf}(A,B) \wedge \text{parentOf}(A,C) \wedge \text{parentOf}(C,x) \wedge F(A) \wedge M(B) \wedge M(x)\ )$$

Or in **graphs / hypergraphs**



Queries

Typically expressed as formulae of some logic (the query language) with free variables.

A lot of options, tradeoff between expressivity and computational well-behavedness.

Conjunctive Queries

Example:

« Find all x *such that x is a female and has a child who is a female »*

$\exists y\ (\mathrm{Female}(x) \wedge \mathrm{childOf}(x, y) \wedge \mathrm{Female}(y))$	FOL formula
$Q(x) = \mathrm{Female}(x), \mathrm{childOf}(x, y), \mathrm{Female}(y)$	Common notation
$\mathrm{ans}(x) \leftarrow \mathrm{Female}(x), \mathrm{childOf}(x, y), \mathrm{Female}(y)$	Datalog notation
SELECT … FROM … WHERE …	SQL/SPARQL

Formally: A **CQ** *Q* has the form $\exists x_{k+1},\ldots,x_m\ A_1 \wedge \ldots \wedge A_p$
where $A_1,\ldots,A_p$ are atoms over the variables $x_1,\ldots,x_m$
and $x_1 \ldots x_k$ are free variables (defining the answer part)

If k = 0, *Q* is a **Boolean CQ** (existentially closed conjunctive formula)
then the answer can only be *yes* or *no.*

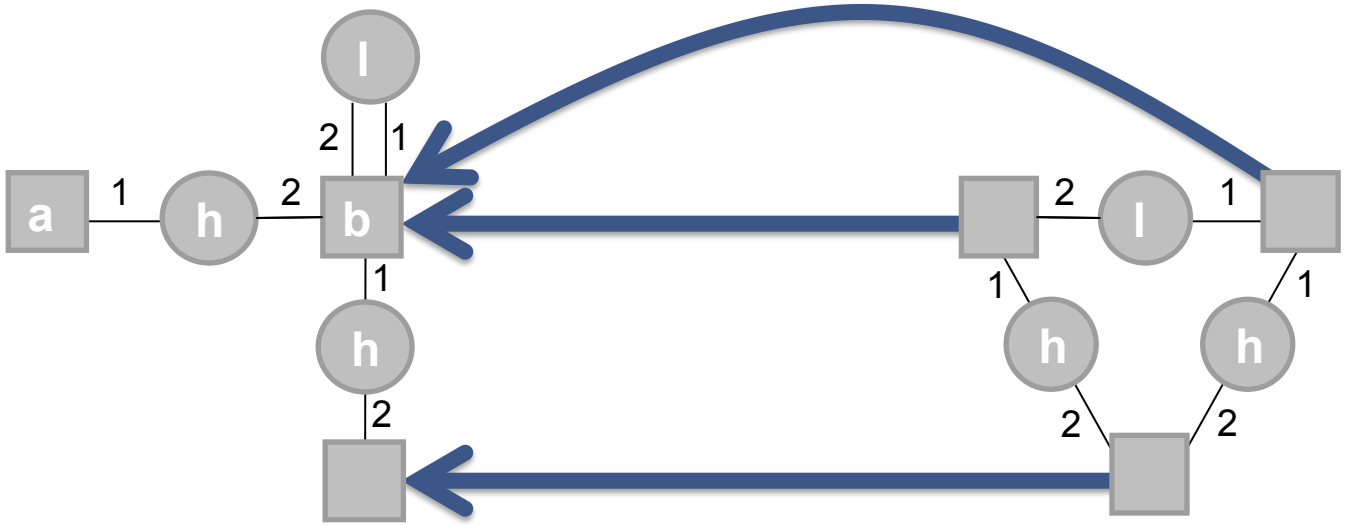
Evaluating Boolean CQs over Data

Data:

$\exists x\ (\mathrm{loves}(bob,bob)$
$\quad \wedge\ \mathrm{hates}(bob,x)$
$\quad \wedge\ \mathrm{hates}(alice,bob))$

Query:

$\exists xyz\ (\mathrm{loves}(x,y)$
$\quad \wedge\ \mathrm{hates}(x,z)$
$\quad \wedge\ \mathrm{hates}(y,z))$



Homomorphism

Some well-known query languages

TECHNISCHE UNIVERSITÄT DRESDEN

- **Conjunctive queries (CQ)**
  - … and their unions
- **First-order logic (FOL)**
  - Basis of SQL
- **Datalog**
  - Recursive, higher-order language
- **Second-order logic (SO)**

Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Why Ontological Query Answering?

- vocabulary of data and query may not coincide (→ information exchange)
- databases may be incomplete
- some information may only be obtained when factoring in background knowledge

Views on Ontological Query Answering

TECHNISCHE UNIVERSITÄT DRESDEN

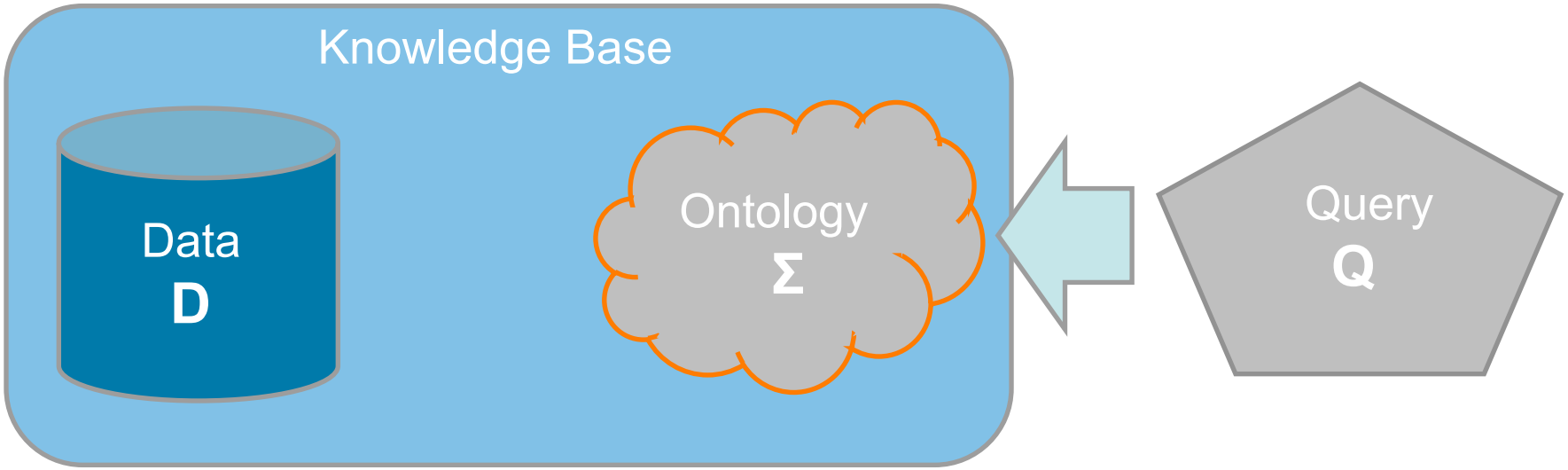
The Plain View:

$$D \models_{\Sigma} Q$$

Views on Ontological Query Answering

TECHNISCHE UNIVERSITÄT DRESDEN

The Knowledge Representation View:

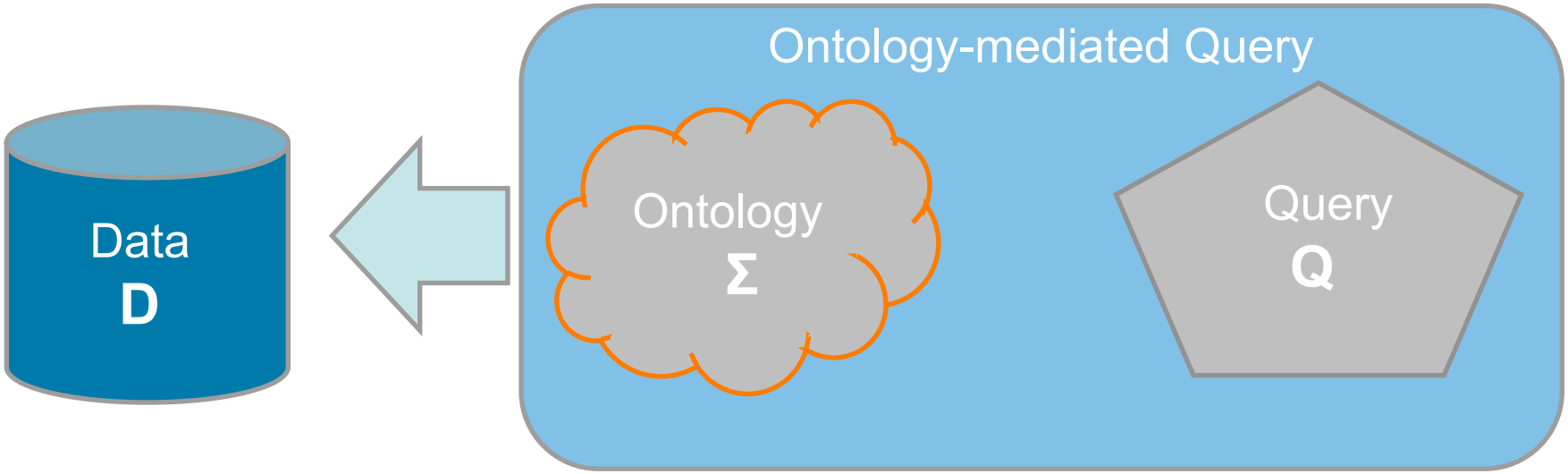


$$D \wedge \Sigma \models Q$$

Views on Ontological Query Answering

TECHNISCHE UNIVERSITÄT DRESDEN

The Database View:



$$D \models (\bigwedge \Sigma) \rightarrow Q$$

Our Main Showcase:

Our Main Showcase:

Existential Rules

TECHNISCHE UNIVERSITÄT DRESDEN

« Value Invention »

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

X, Y, Z :
tuples of variables

Any conjunction of atoms (on variables and constants)

∀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+/- [Cali Gottlob Lukasiewicz PODS 2009]
- Same as the logical translation of Conceptual Graph rules
- Generalize lightweight DLs used for OBQA

DL-Lite Family

DL-Lite: Popular family of DLs - at the basis of the OWL 2 QL profile of OWL

DL-Lite Axioms	First-order Representation
$A \sqsubseteq B$	$\forall x\ (A(x) \rightarrow B(x))$
$A \sqsubseteq \exists R$	$\forall x\ (A(x) \rightarrow \exists y\ R(x,y))$
$\exists R \sqsubseteq A$	$\forall x \forall y\ (R(x,y) \rightarrow A(x))$
$\exists R \sqsubseteq \exists P$	$\forall x \forall y\ (R(x,y) \rightarrow \exists z\ P(x,z))$
$A \sqsubseteq \exists R.B$	$\forall x\ (A(x) \rightarrow \exists y\ (R(x,y) \wedge B(y)))$
$R \sqsubseteq P$	$\forall x \forall y\ (R(x,y) \rightarrow P(x,y))$
$A \sqsubseteq \neg B$	$\forall x\ (A(x) \wedge B(x) \rightarrow \bot)$

The Description Logic EL

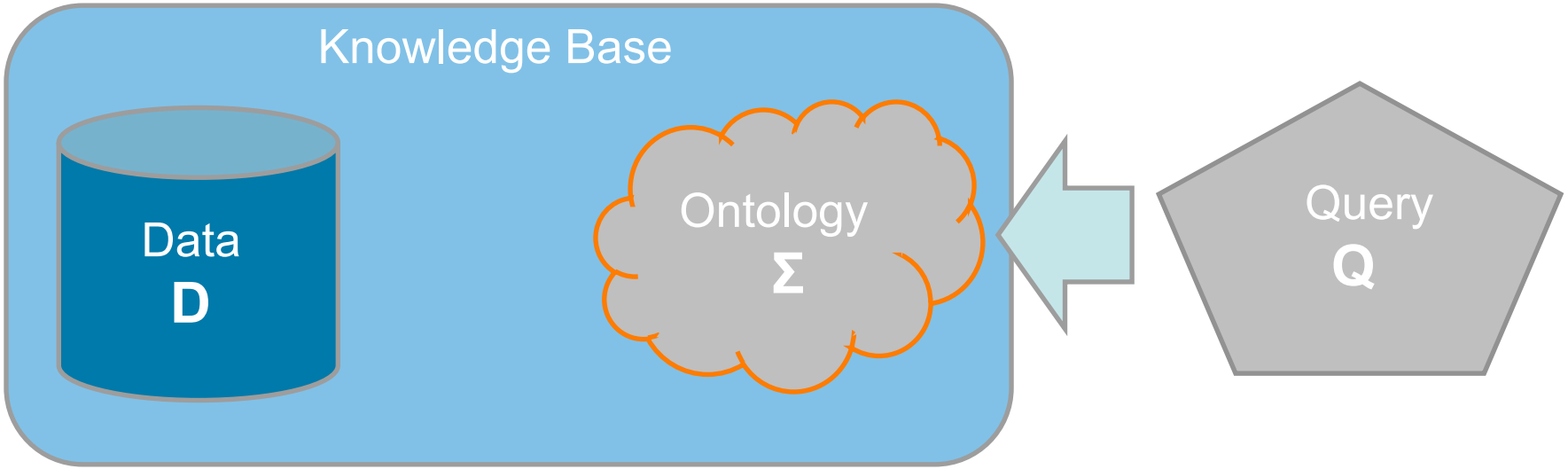
EL: Popular DL for biological applications - at the basis of OWL 2 EL profile

EL Axioms	First-order Representation
$A \sqsubseteq B$	$\forall x\ (A(x) \rightarrow B(x))$
$A \sqcap B \sqsubseteq C$	$\forall x\ (A(x) \wedge B(x) \rightarrow C(x))$
$A \sqsubseteq \exists R.B$	$\forall x\ (A(x) \rightarrow \exists y\ (R(x,y) \wedge B(y)))$
$\exists R.B \sqsubseteq A$	$\forall x \forall y\ (R(x,y) \wedge B(y) \rightarrow A(x))$

Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Computation Approaches Inspired by KR View



rewrite knowledge base into a representation that is easier to query, by making the structure of its models more explicit

- for FOL: semantic tableau (also used for Description Logics)
- for Existential Rules: the chase

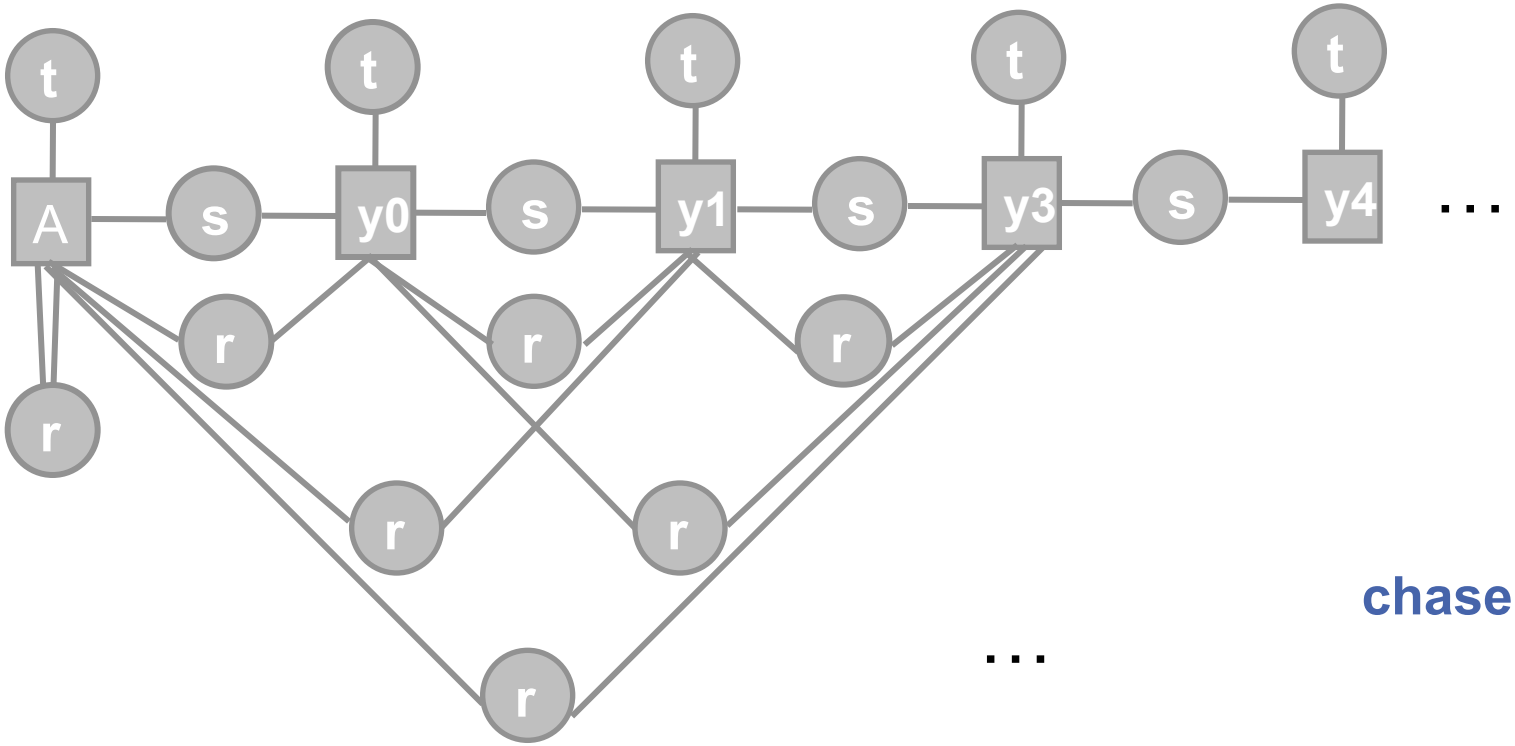
The Chase

Σ

$t(x) \rightarrow s(x,y) \wedge t(y)$

$t(x) \wedge t(y) \rightarrow r(x,y)$

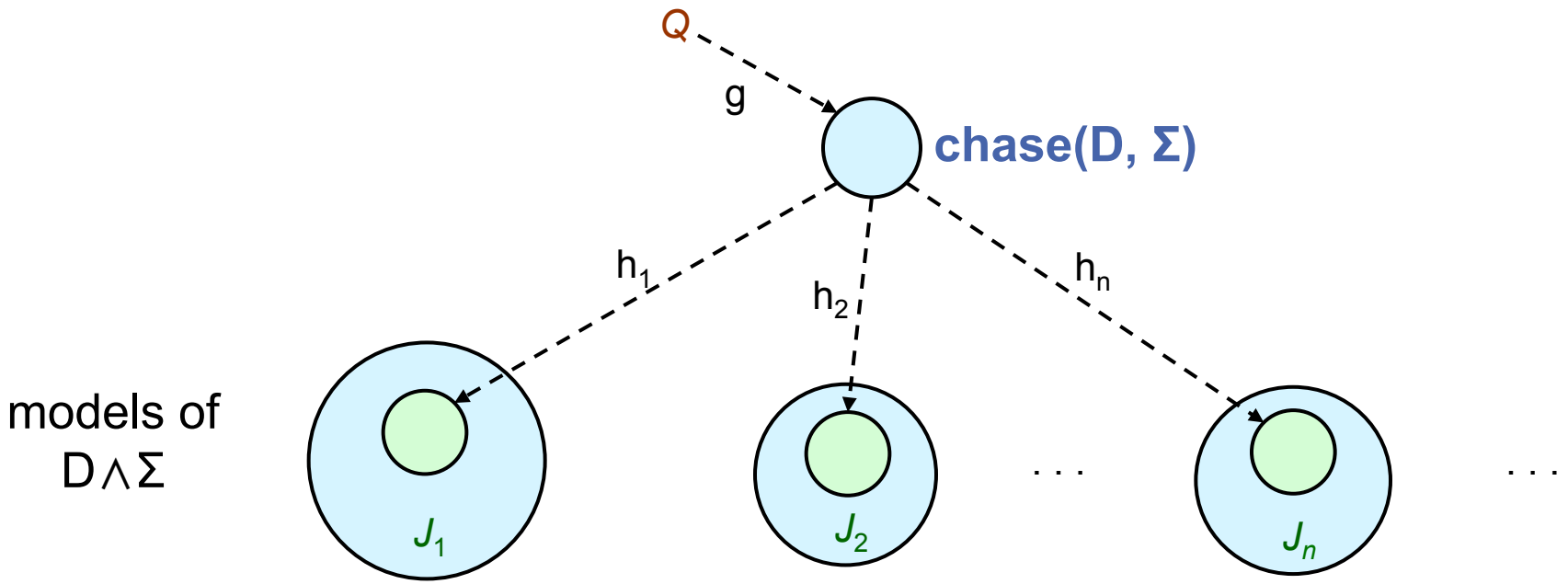
D= {t(A)}



chase(D, Σ)

The Chase

- $D \models_{\Sigma} Q$ iff chase(D,Σ) satisfies Q
- works for all query languages where satisfaction is preserved under homomorphism (not just for CQs)
- produces a universal model



The Chase

- $D \models_{\Sigma} Q$ iff chase(D,Σ) satisfies Q
- works for all query languages where satisfaction is preserved under homomorphism (not just for CQs)
- produces a universal model
- but: not sufficient for decidability, only in cases where the chase is known to have further properties:
  - finiteness
  - bounded tree-width
- checking these properties is again undecidable
- there exist sufficient (groups of) criteria
  - acyclicity
  - guardedness

Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Acyclicity → Finite Universal Models

TECHNISCHE UNIVERSITÄT DRESDEN

- **Goal:** find a simple syntactic way to guarantee that bottom-up rule applications lead to only finitely many invented values

  (such a set of rules is also called *fes* – *finite extension set*)

- **Idea:**
  - Trace the predicate positions that an invented value can "move" to
  - Use this to estimate if the invented value could (indirectly) trigger the rule that it was generated from

Example

$$\text{mother}(x) \rightarrow \text{person}(x)$$

$$\text{person}(x) \wedge \text{mother}(x) \rightarrow \exists v.\text{parent}(v,x) \wedge \text{person}(v)$$

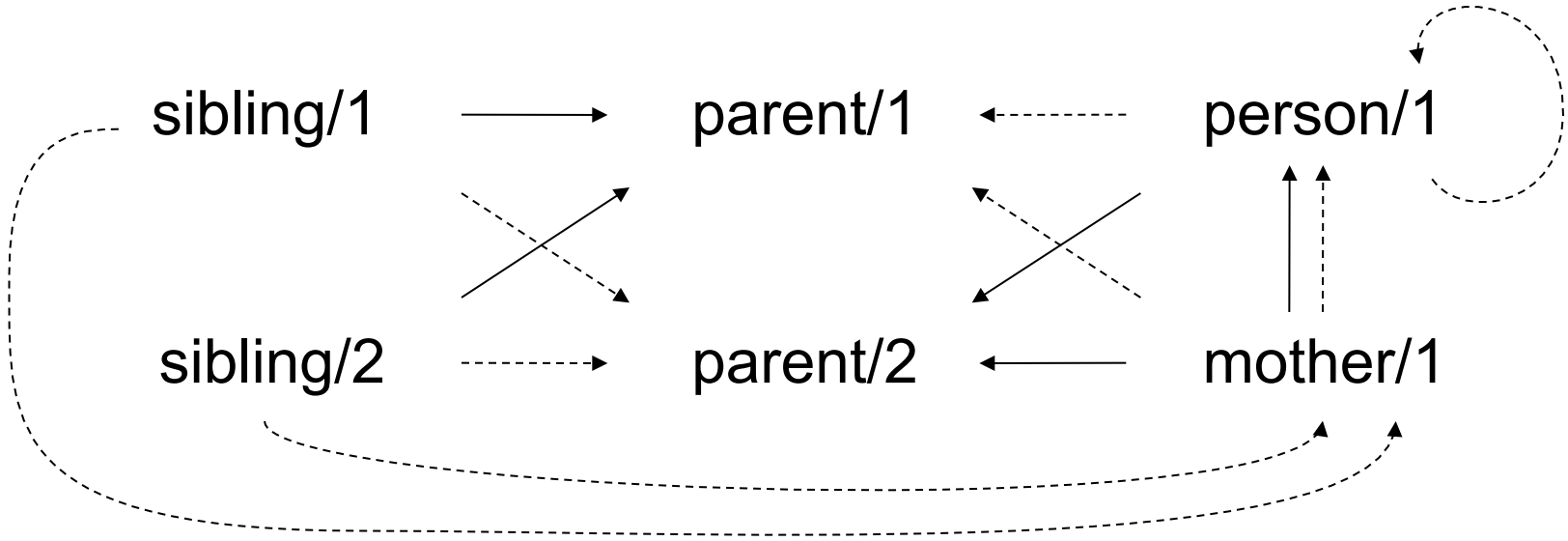
$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

Weak Acyclicity

$$\text{mother}(x) \rightarrow \text{person}(x)$$

$$\text{person}(x) \wedge \text{mother}(x) \rightarrow \exists v.\text{parent}(v,x) \wedge \text{person}(v)$$

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$



Joint Acyclicity

$$\text{mother}(x) \rightarrow \text{person}(x)$$

$$\text{person}(x) \wedge \text{mother}(x) \rightarrow \exists v.\text{parent}(v,x) \wedge \text{person}(v)$$

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

Joint Acyclicity

$$\text{mother}(x) \rightarrow \text{person}(x)$$

$$\text{person}(x) \wedge \text{mother}(x) \rightarrow \exists v.\text{parent}(v,x) \wedge \text{person}(v)$$

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

Ω_v: parent/1

Joint Acyclicity

$$\text{mother}(x) \rightarrow \text{person}(x)$$

$$\text{person}(x) \wedge \text{mother}(x) \rightarrow \exists v.\text{parent}(v,x) \wedge \text{person}(v)$$

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

Ω_v: parent/1, person/1

Joint Acyclicity

$$\text{mother}(x) \rightarrow \text{person}(x)$$

$$\text{person}(x) \wedge \text{mother}(x) \rightarrow \exists v.\text{parent}(v,x) \wedge \text{person}(v)$$

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

Ω_v: parent/1, person/1

Ω_w: parent/2, mother/1, person/1

Joint Acyclicity

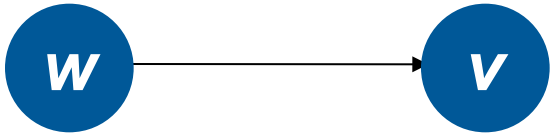
$$\text{mother}(x) \rightarrow \text{person}(x)$$

$$\text{person}(x) \wedge \text{mother}(x) \rightarrow \exists v.\text{parent}(v,x) \wedge \text{person}(v)$$

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

Ω_v: parent/1, person/1

Ω_w: parent/2, mother/1, person/1



Eliminating Jointly Acyclic Existential Quantifiers

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

Eliminating Jointly Acyclic Existential Quantifiers

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

- **Skolemize:**

$$\text{sibling}(x,y) \rightarrow \text{parent}(x,f(x,y)) \wedge \text{parent}(y,f(x,y)) \wedge \text{mother}(f(x,y))$$

Eliminating Jointly Acyclic Existential Quantifiers

TECHNISCHE UNIVERSITÄT DRESDEN

$$\text{sibling}(x,y) \rightarrow \exists w.\text{parent}(x,w) \wedge \text{parent}(y,w) \wedge \text{mother}(w)$$

- **Skolemize:**

$$\text{sibling}(x,y) \rightarrow \text{parent}(x,f(x,y)) \wedge \text{parent}(y,f(x,y)) \wedge \text{mother}(f(x,y))$$

- But **function symbols** difficult to handle computationally
- Get rid of functions by **flattening:**

$$\text{sibling}(x,y) \rightarrow \text{parent}(x,f,x,y) \wedge \text{parent}(y,f,x,y) \wedge \text{mother}(f,x,y)$$

Flattened Example

Arity of predicates increases everywhere:

$$\text{mother}(x,x_1,x_2) \rightarrow \text{person}(x,x_1,x_2)$$

$$\text{person}(x,x_1,x_2) \wedge \text{mother}(x,x_1,x_2) \rightarrow \exists v.\text{parent}(v,x,x_1,x_2) \wedge \text{person}(v,\square,\square)$$

$$\text{sibling}(x,y) \rightarrow \text{parent}(x,f,x,y) \wedge \text{parent}(y,f,x,y) \wedge \text{mother}(f,x,y)$$

Can this Actually Work?

TECHNISCHE UNIVERSITÄT DRESDEN

- Flattening can only represent terms of depth up to 1
- **Acyclicity controls nesting of Skolem functions symbols**
- Flattening based on topological order of variables in existential dependency graph
- Transformation can increase predicate arity **exponentially**

Today, a plethora of more acyclicity notions exist. Typically there is a trade-off between coverage and cost of checking.

Outline

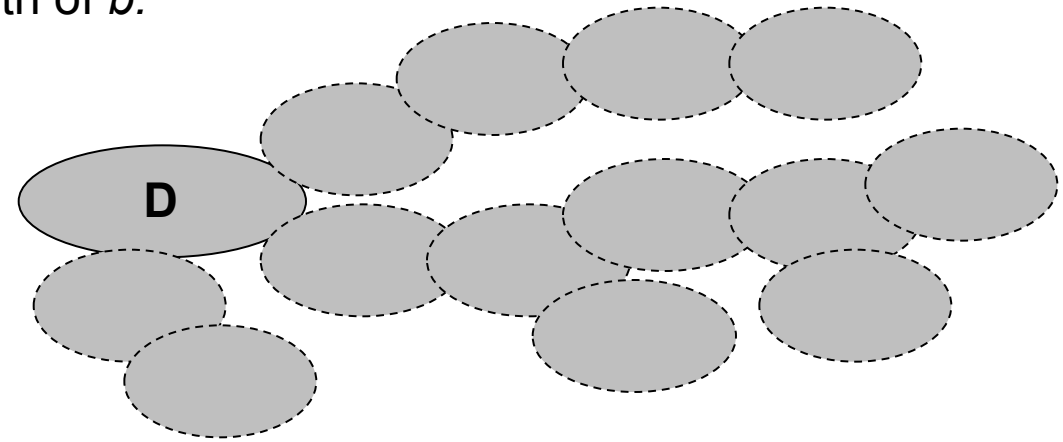
- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Bounded Treewidth Sets of Rules

Essentially [Cali Gottlob Kifer KR'08]

Σ is *bts* (bounded treewidth set) if for any database D, the chase with Σ generates a structure with finite treewidth

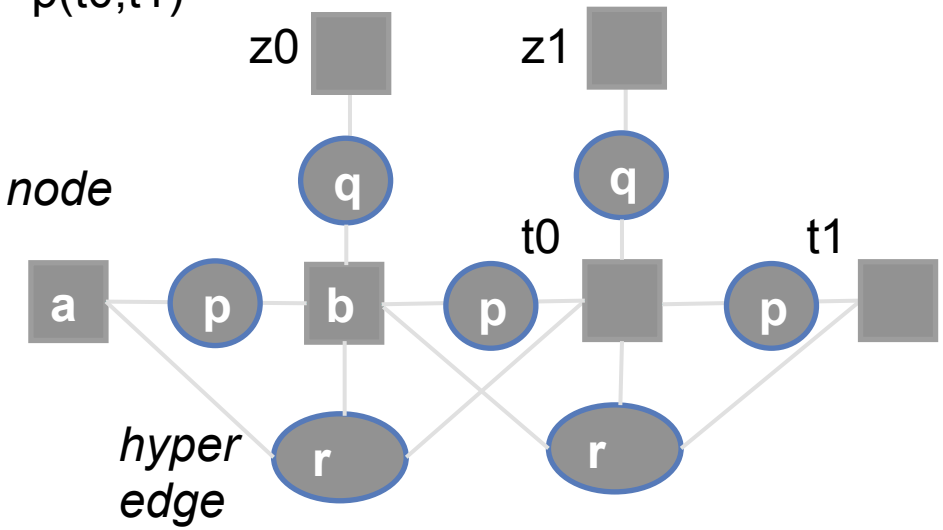
i.e., for any database D, there is an integer *b* such that chase(D,Σ) has treewidth of *b.*

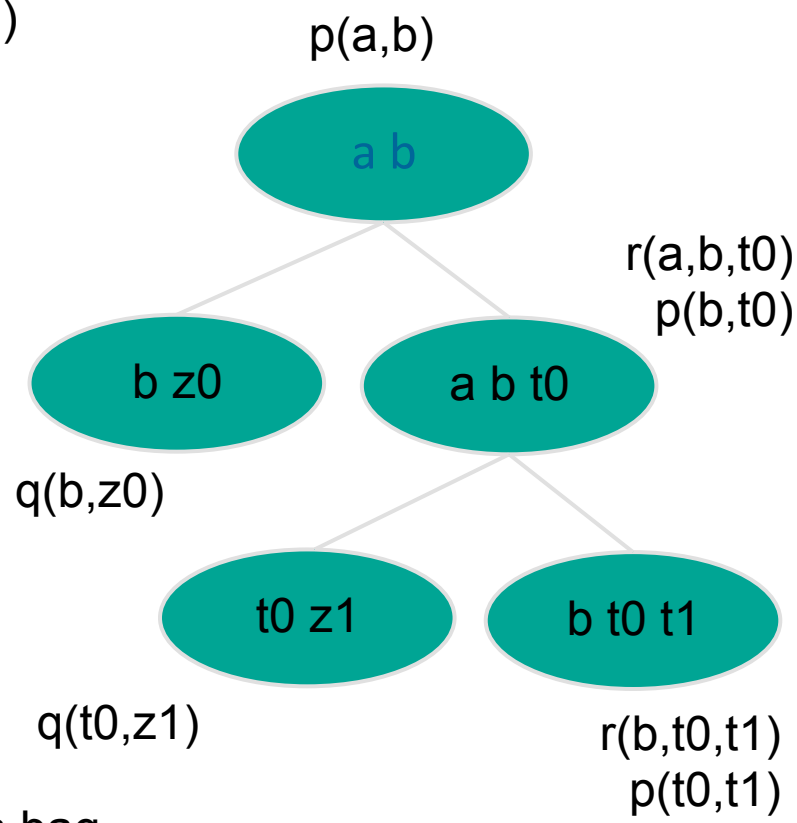


All Σ with finite chase are included in *bts*
(bound given by the number of terms in the finite chase)

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

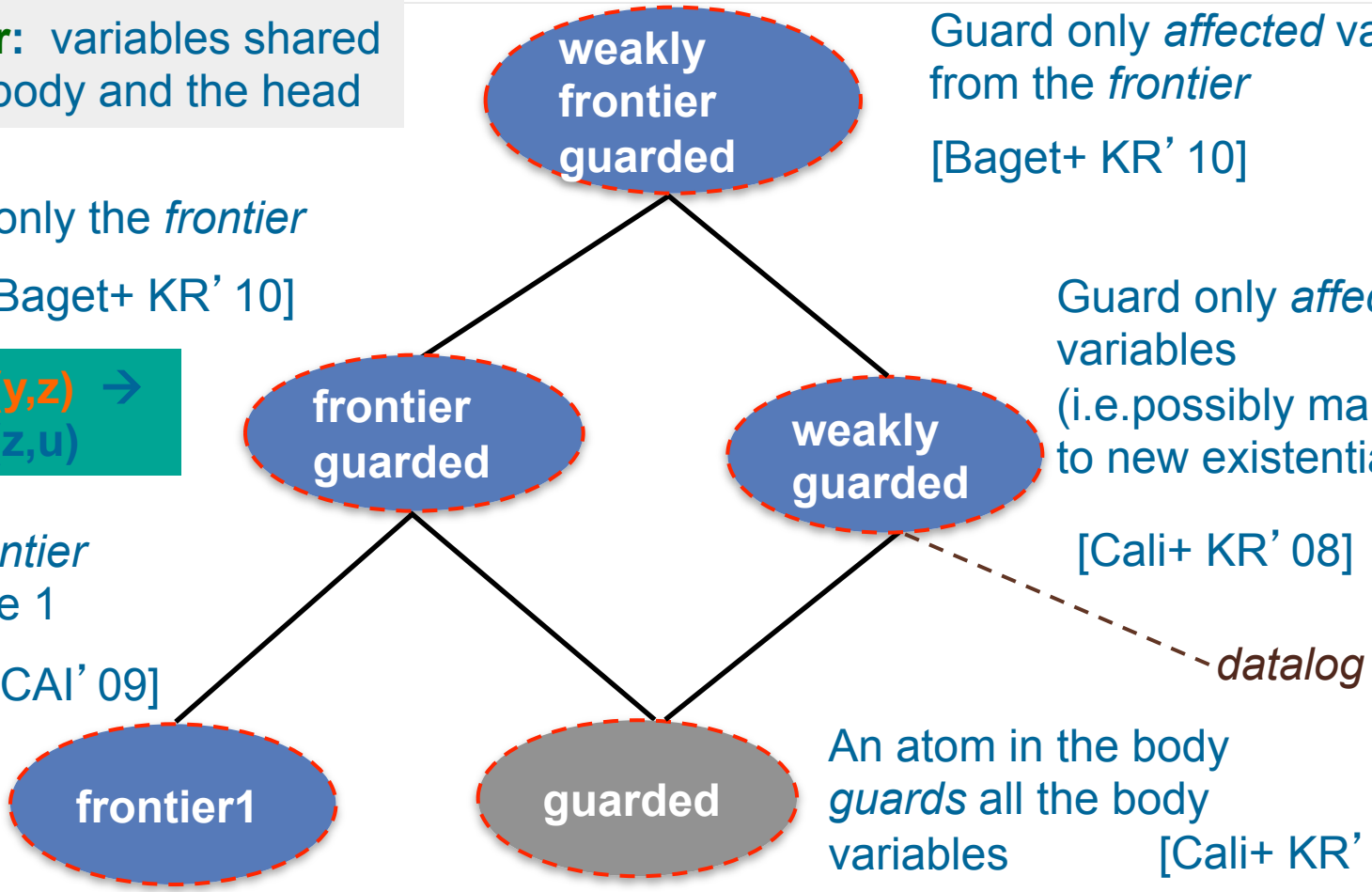
Guardedness → Universal Models of Bounded Treewidth

General Idea: variables of a rule that might represent invented values must occur together in *one* body atom

The Family of Guarded Rules (Selection)

TECHNISCHE UNIVERSITÄT DRESDEN

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]

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

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

[Cali+ KR' 08]

The *frontier* has size 1

[Baget+ IJCAI' 09]

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

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

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

Example

$$c(x) \land ancestor(x,y) \land ancestor(y,z) \rightarrow ancestor(x,z)$$

$$parent(x,y) \rightarrow ancestor(x,y)$$

$$c(x) \rightarrow person(x)$$

$$person(x) \rightarrow \exists w.parent(x,w) \land person(w)$$

$$sibling(x,y) \rightarrow \exists v.parent(x,v) \land parent(y,v) \land c(v)$$

$$parent(x,y) \land sibling(y,z) \rightarrow uncle(x,z)$$

Guardedness

$$c(\boldsymbol{x}) \land \text{ancestor}(\boldsymbol{x},\boldsymbol{y}) \land \text{ancestor}(\boldsymbol{y},\boldsymbol{z}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{z})$$

$$\text{parent}(\boldsymbol{x},\boldsymbol{y}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{y})$$

$$c(\boldsymbol{x}) \rightarrow \text{person}(\boldsymbol{x})$$

$$\text{person}(\boldsymbol{x}) \rightarrow \exists\, w.\text{parent}(\boldsymbol{x},w) \land \text{person}(w)$$

$$\text{sibling}(\boldsymbol{x},\boldsymbol{y}) \rightarrow \exists\, v.\text{parent}(\boldsymbol{x},v) \land \text{parent}(\boldsymbol{y},v) \land c(v)$$

$$\text{parent}(\boldsymbol{x},\boldsymbol{y}) \land \text{sibling}(\boldsymbol{y},\boldsymbol{z}) \rightarrow \text{uncle}(\boldsymbol{x},\boldsymbol{z})$$

Frontier-Guardedness

$$c(\boldsymbol{x}) \land \text{ancestor}(\boldsymbol{x},y) \land \text{ancestor}(y,\boldsymbol{z}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{z})$$

$$\text{parent}(\boldsymbol{x},\boldsymbol{y}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{y})$$

$$c(\boldsymbol{x}) \rightarrow \text{person}(\boldsymbol{x})$$

$$\text{person}(\boldsymbol{x}) \rightarrow \exists\, w.\text{parent}(\boldsymbol{x},w) \land \text{person}(w)$$

$$\text{sibling}(\boldsymbol{x},\boldsymbol{y}) \rightarrow \exists\, v.\text{parent}(\boldsymbol{x},v) \land \text{parent}(\boldsymbol{y},v) \land c(v)$$

$$\text{parent}(\boldsymbol{x},y) \land \text{sibling}(y,\boldsymbol{z}) \rightarrow \text{uncle}(\boldsymbol{x},\boldsymbol{z})$$

Weak Guardedness

$$c(\boldsymbol{x}) \wedge \text{ancestor}(\boldsymbol{x},\boldsymbol{y}) \wedge \text{ancestor}(\boldsymbol{y},\boldsymbol{z}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{z})$$

$$\text{parent}(\boldsymbol{x},\boldsymbol{y}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{y})$$

$$c(\boldsymbol{x}) \rightarrow \text{person}(\boldsymbol{x})$$

$$\text{person}(\boldsymbol{x}) \rightarrow \exists\, w.\text{parent}(\boldsymbol{x},w) \wedge \text{person}(w)$$

$$\text{sibling}(x,y) \rightarrow \exists\, v.\text{parent}(x,v) \wedge \text{parent}(y,v) \wedge c(v)$$

$$\text{parent}(\boldsymbol{x},\boldsymbol{y}) \wedge \text{sibling}(y,z) \rightarrow \text{uncle}(\boldsymbol{x},z)$$

Weak Frontier-Guardedness

$$c(\boldsymbol{x}) \wedge \text{ancestor}(\boldsymbol{x},y) \wedge \text{ancestor}(y,\boldsymbol{z}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{z})$$

$$\text{parent}(\boldsymbol{x},\boldsymbol{y}) \rightarrow \text{ancestor}(\boldsymbol{x},\boldsymbol{y})$$

$$c(\boldsymbol{x}) \rightarrow \text{person}(\boldsymbol{x})$$

$$\text{person}(\boldsymbol{x}) \rightarrow \exists\, w.\text{parent}(\boldsymbol{x},w) \wedge \text{person}(w)$$

$$\text{sibling}(x,y) \rightarrow \exists\, v.\text{parent}(x,v) \wedge \text{parent}(y,v) \wedge c(v)$$

$$\text{parent}(\boldsymbol{x},y) \wedge \text{sibling}(y,z) \rightarrow \text{uncle}(\boldsymbol{x},z)$$

Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Reconciling Acyclicity and Guardedness

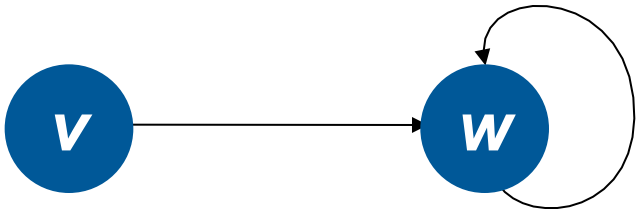
TECHNISCHE UNIVERSITÄT DRESDEN

- **Idea:** No need to guard invented values that don't occur in existential dependency cycles

Approach for reasoning:

- Eliminate all jointly acyclic existential variables by flattening
- Process remaining (guarded) existential variables as usual

Example



$$c(x) \land ancestor(x,y) \land ancestor(y,z) \rightarrow ancestor(x,z)$$

$$parent(x,y) \rightarrow ancestor(x,y)$$

$$c(x) \rightarrow person(x)$$

$$person(x) \rightarrow \exists\, w.parent(x,w) \land person(w)$$

$$sibling(x,y) \rightarrow \exists\, v.parent(x,v) \land parent(y,v) \land c(v)$$

$$parent(x,y) \land sibling(y,z) \rightarrow uncle(x,z)$$

Glut-Guardedness

$$c(x) \wedge ancestor(x,\boldsymbol{y}) \wedge ancestor(\boldsymbol{y},\boldsymbol{z}) \rightarrow ancestor(x,\boldsymbol{z})$$

$$parent(\boldsymbol{x},\boldsymbol{y}) \rightarrow ancestor(\boldsymbol{x},\boldsymbol{y})$$

$$c(x) \rightarrow person(x)$$

$$person(\boldsymbol{x}) \rightarrow \exists\, w.parent(\boldsymbol{x},w) \wedge person(w)$$

$$sibling(x,y) \rightarrow \exists\, v.parent(x,v) \wedge parent(y,v) \wedge c(v)$$

$$parent(\boldsymbol{x},y) \wedge sibling(y,z) \rightarrow uncle(\boldsymbol{x},z)$$

Glut-Frontier-Guardedness

$$c(x) \wedge ancestor(x,y) \wedge ancestor(y,\boldsymbol{z}) \rightarrow ancestor(x,\boldsymbol{z})$$

$$parent(\boldsymbol{x},\boldsymbol{y}) \rightarrow ancestor(\boldsymbol{x},\boldsymbol{y})$$

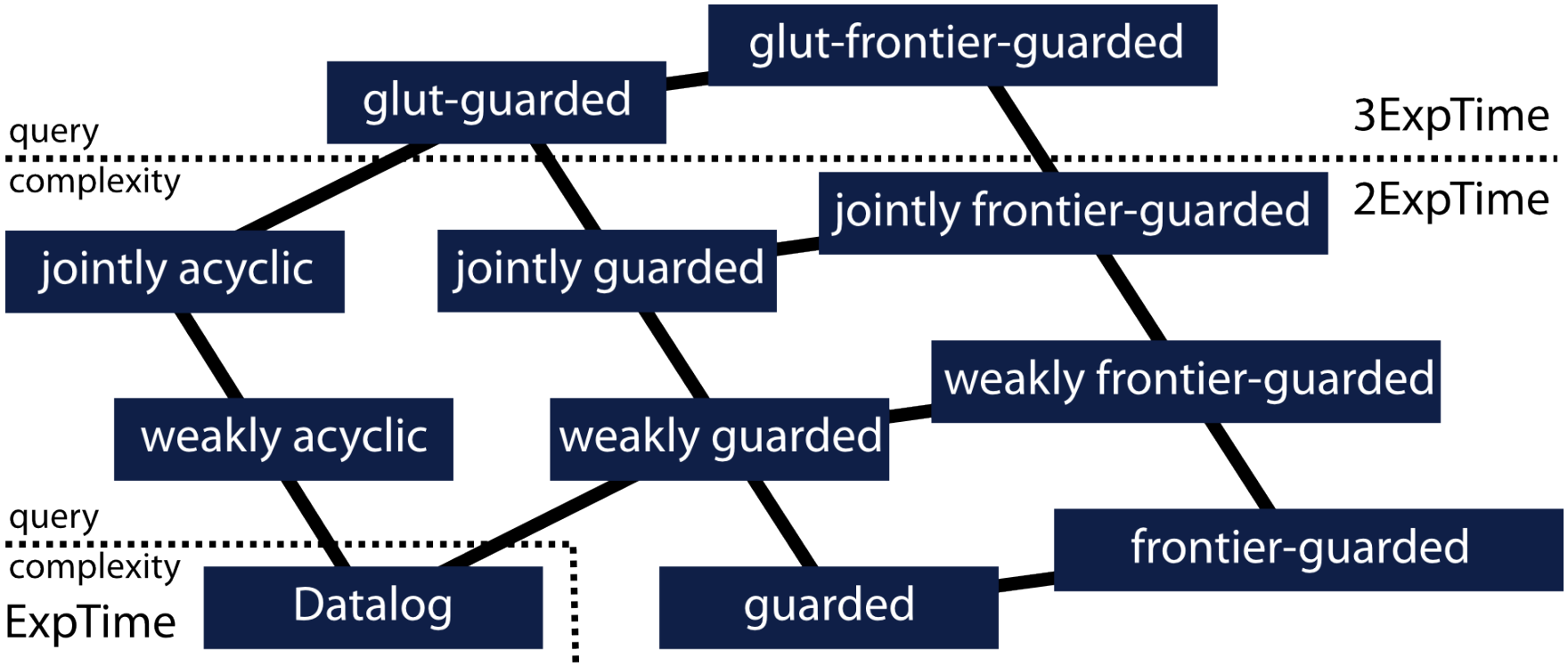
$$c(x) \rightarrow person(x)$$

$$person(\boldsymbol{x}) \rightarrow \exists\, w.parent(\boldsymbol{x},w) \wedge person(w)$$

$$sibling(x,y) \rightarrow \exists\, v.parent(x,v) \wedge parent(y,v) \wedge c(v)$$

$$parent(\boldsymbol{x},y) \wedge sibling(y,z) \rightarrow uncle(\boldsymbol{x},z)$$

Expressivities and Complexities



Main insight:

Acyclicity can be combined with any other decidability principle in a modular way

and the result is more expressive than the sum of its parts

Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Worst-Case Optimal Reasoning with Guarded Existential Rules

TECHNISCHE UNIVERSITÄT DRESDEN

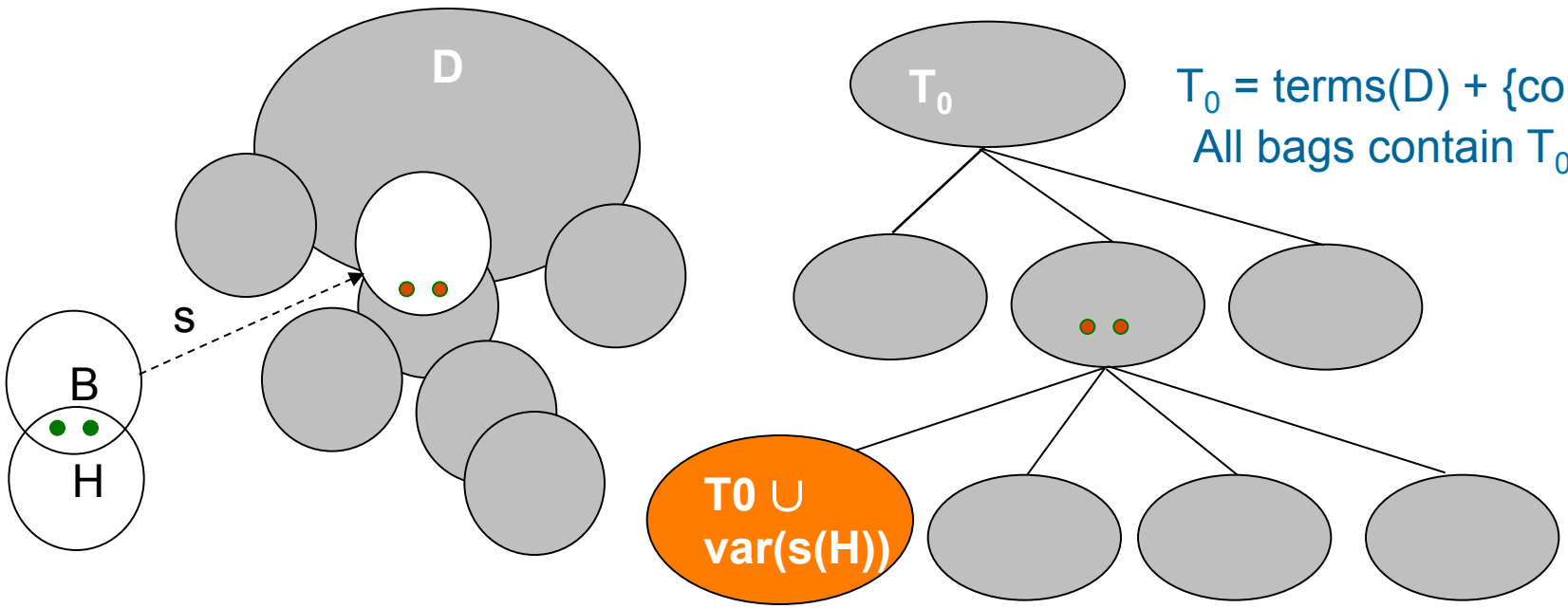
- decidability of bounded-treewidth sets comes with a suboptimal algorithm
- desirable: worst-case optimal algorithm(s) for known versions of guardedness
- to this end:
  - introduce new class: greedy bounded-treewith sets of rules (gbts) subsuming all presented versions of guarded rules
  - provide algorithm for this class

Greedy Bounded-Treewidth Sets of Rules (gbts)

TECHNISCHE UNIVERSITÄT DRESDEN

Allow for a greedy construction of a decomposition tree in the course of the chase

chase under construction *decomposition tree*



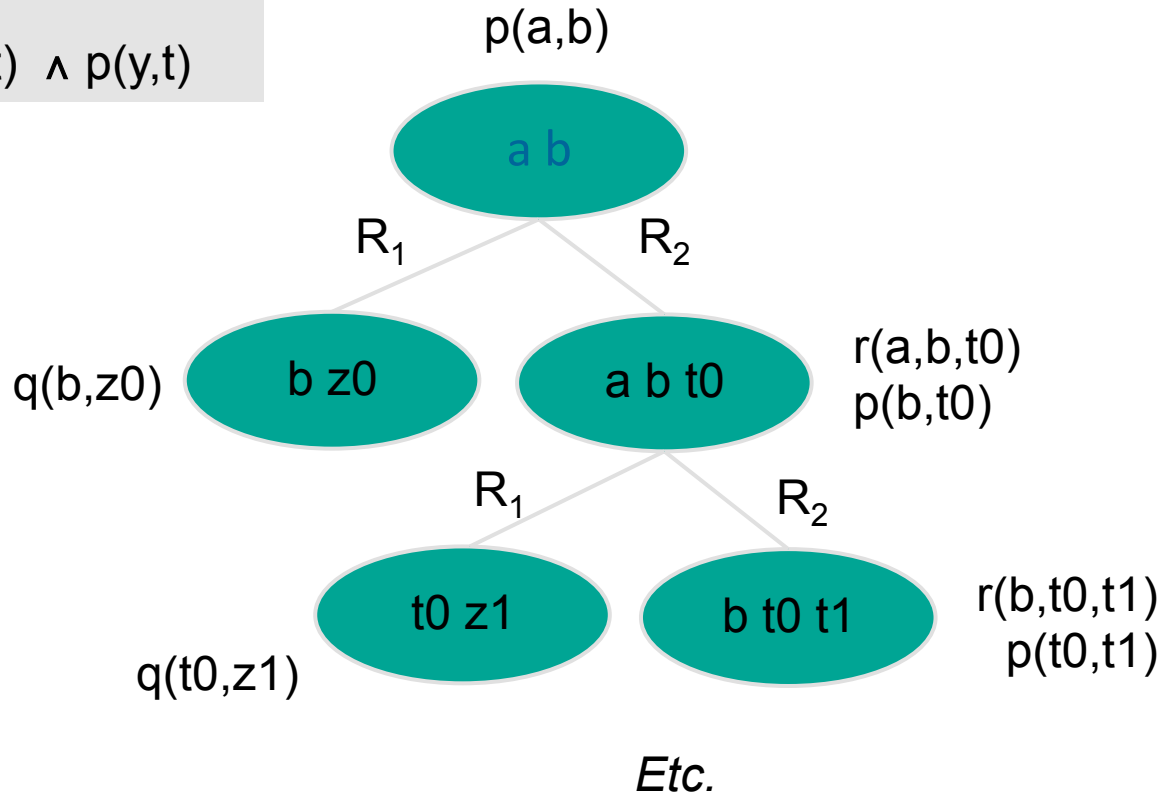
T_0 = terms(D) + {constants}
All bags contain T_0

- Gbts condition:
  Frontier variables that are not being mapped to terms from D are *jointly* mapped to variables occurring in atoms added by a single rule application

Greedy *bts*

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

D = p(a,b)



Greedy construction of a decomposition tree of the chase

with bounded width

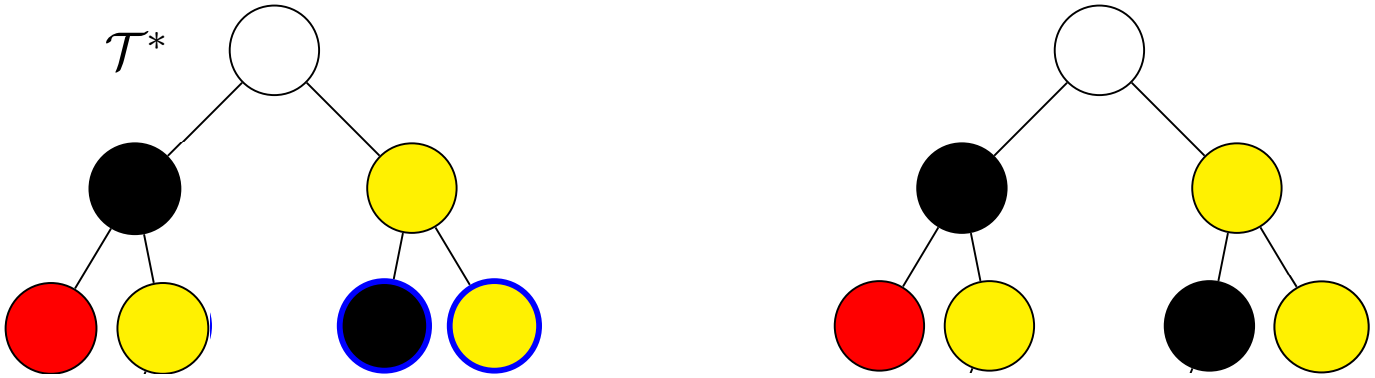
Worst-Case Optimal Reasoning with Guarded Existential Rules

TECHNISCHE UNIVERSITÄT DRESDEN

- for gbts, chase can be “lifted” to the decomposition tree
- introduce *patterns* describing the (finitely many) ways one bag of the tree decomposition may look like (similar to “types” used in logics)
- apply a consequence-based technique to construct the initial part of the greedy tree decomposition, terminate as soon as type-repetition is detected (similar to *blocking* in DL tableaux)

Worst-Case Optimal Reasoning with Guarded Existential Rules

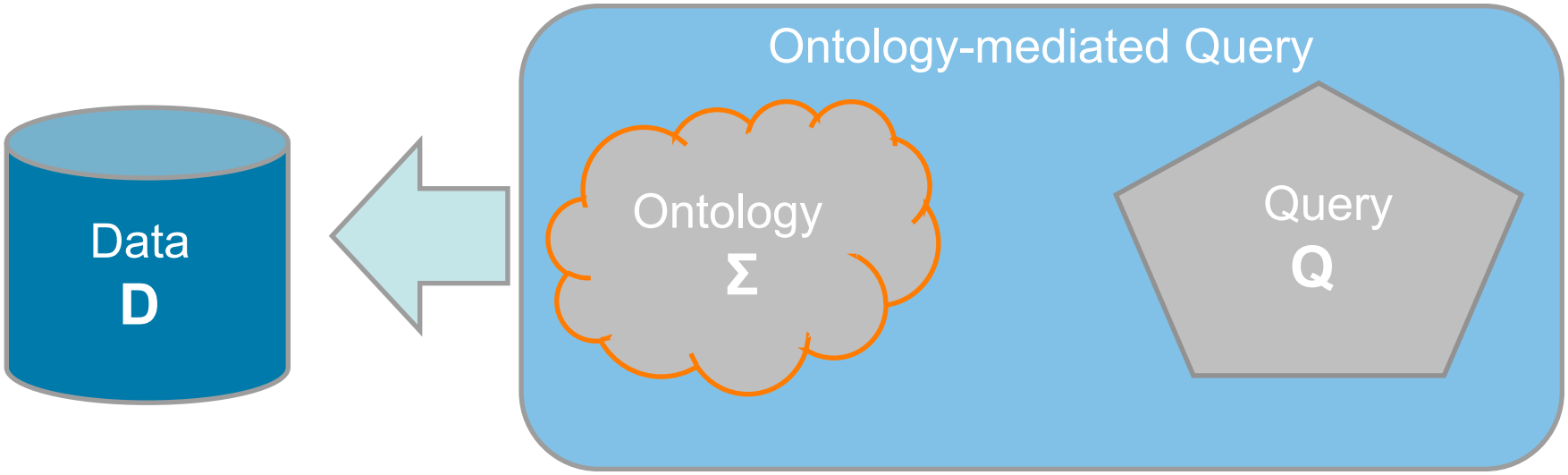
“full blocked tree” represents the full chase and can be used for CQ answering



Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Computation Approaches Inspired by DB View

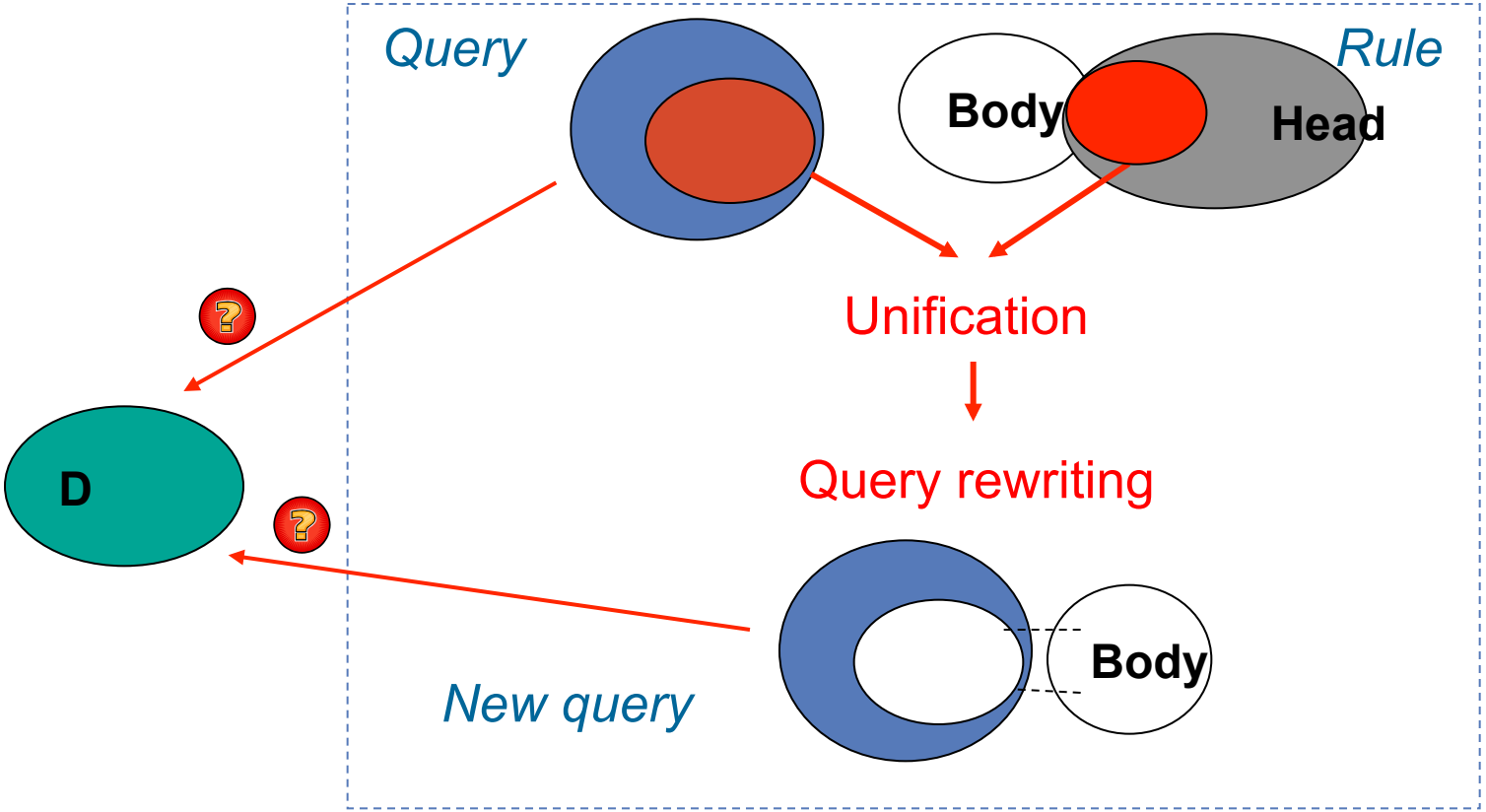


rewrite ontology-mediated query into a representation that is easier to execute against the data

- frequent choice:
  Unions of Conjunctive Queries
- other options: (fragments of) Datalog or disjunctive Datalog

Backward Chaining Scheme

Basic step:



Finite Unification Sets of Rules

- Iterated backward chaining creates a union of CQs (UCQ).
- Terminates if after one rewriting step (w.r.t. all rules) the obtained UCQ is subsumed by the previous one.
- If, given a Σ, termination of backward chaining is guaranteed for all possible Q, we call Σ *finite unification set* (fus) or *first-order rewritable.*
- This property is not decidable, but again, syntactic criteria have been identified:
  DL-Lite, sticky rules, sticky-join rules,…

Outline

- Classical Query Answering
  - Some Remarks about Query Languages
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Views on Ontological Query Answering

TECHNISCHE UNIVERSITÄT DRESDEN

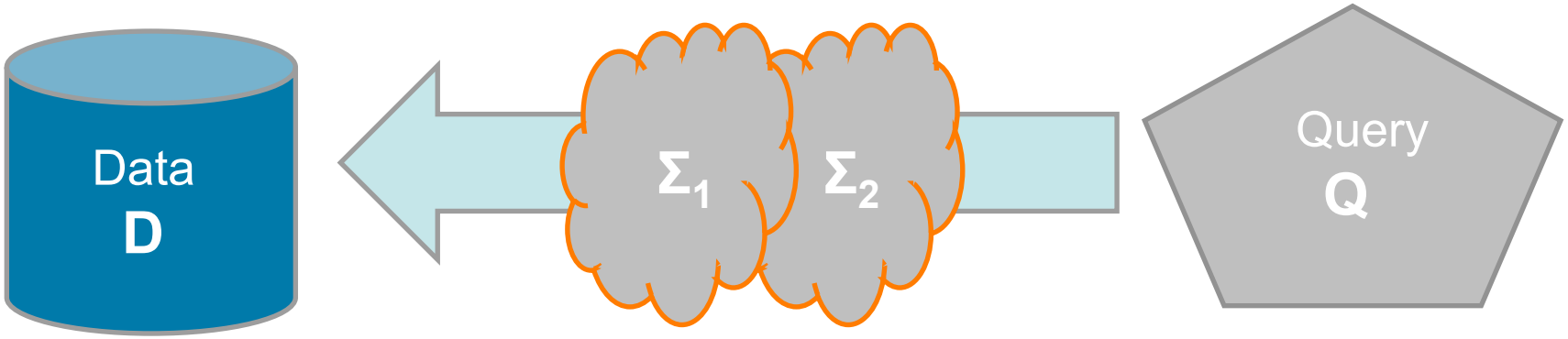
The Mixed View:

$$D \models_{\Sigma} Q$$

Views on Ontological Query Answering

TECHNISCHE UNIVERSITÄT DRESDEN

The Mixed View:

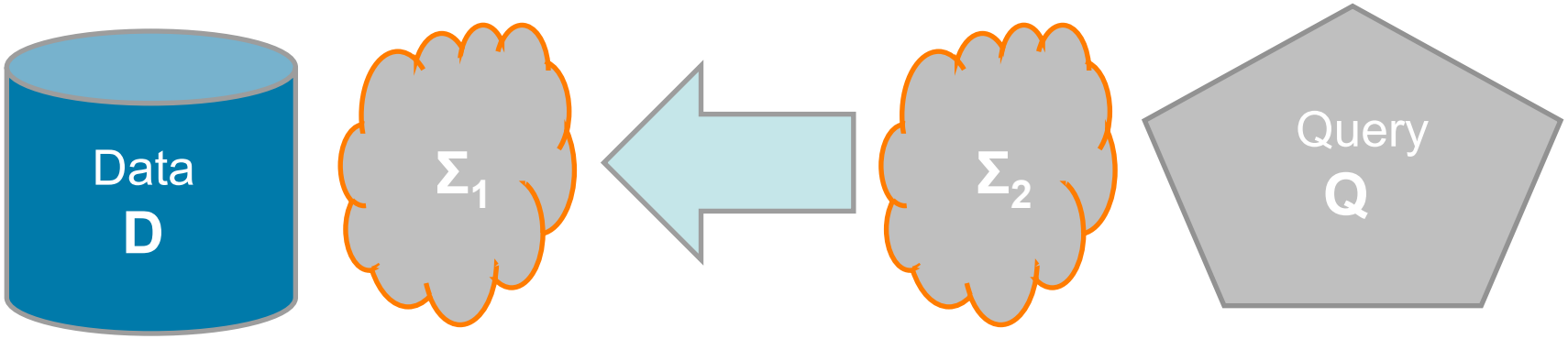


$$D \models_{\Sigma_1+\Sigma_2} Q$$

Views on Ontological Query Answering

TECHNISCHE UNIVERSITÄT DRESDEN

The Mixed View:

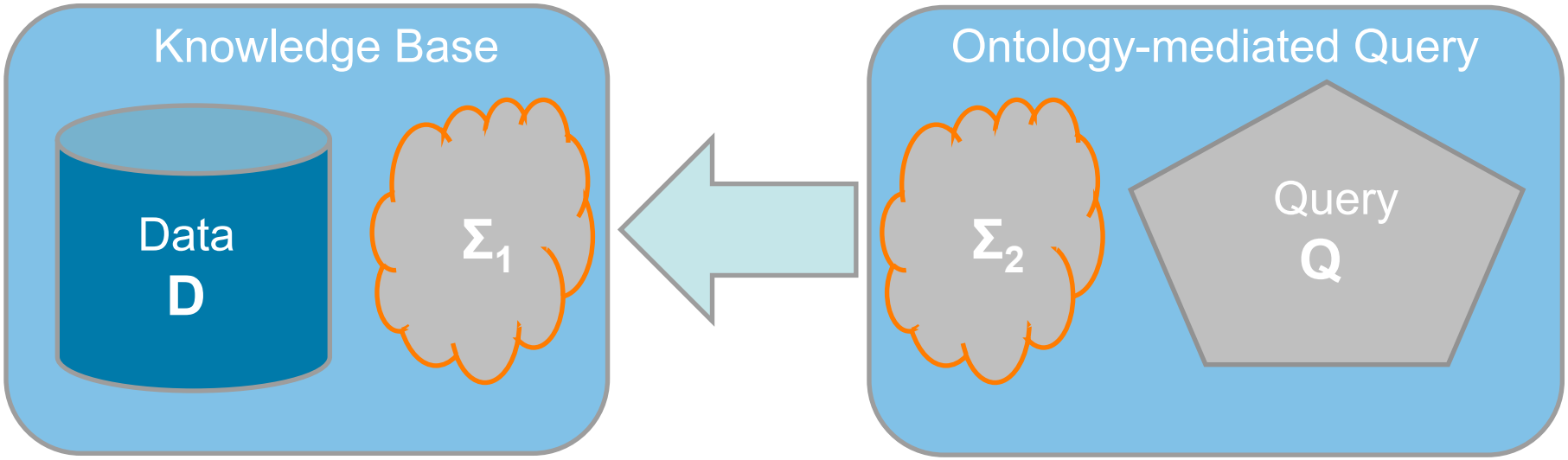


$$(D,\Sigma_1) \models (\Sigma_2,Q)$$

Views on Ontological Query Answering

TECHNISCHE UNIVERSITÄT DRESDEN

The Mixed View:



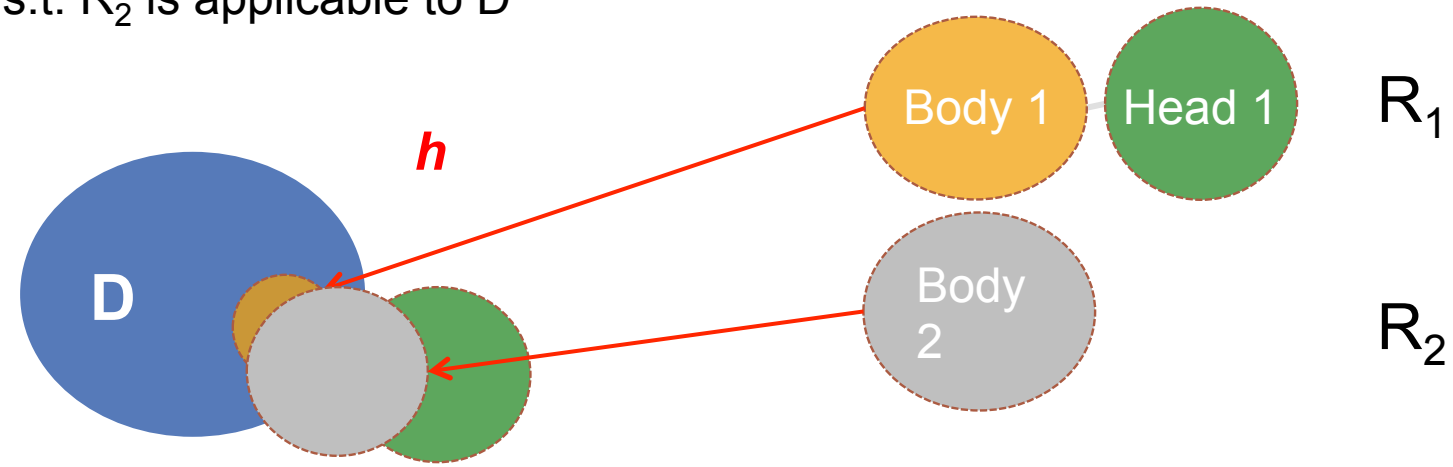
$$(D,\Sigma_1) \models (\Sigma_2,Q)$$

A tool : the Graph of Rule Dependencies

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

R_2 depends on R_1 if applying R_1 may trigger a *new* application of R_2

i.e., there exists a database D s.t. R_1 is applicable to D but R_2 is not
and there is an application of R_1 to D leading to D'
s.t. R_2 is applicable to D'



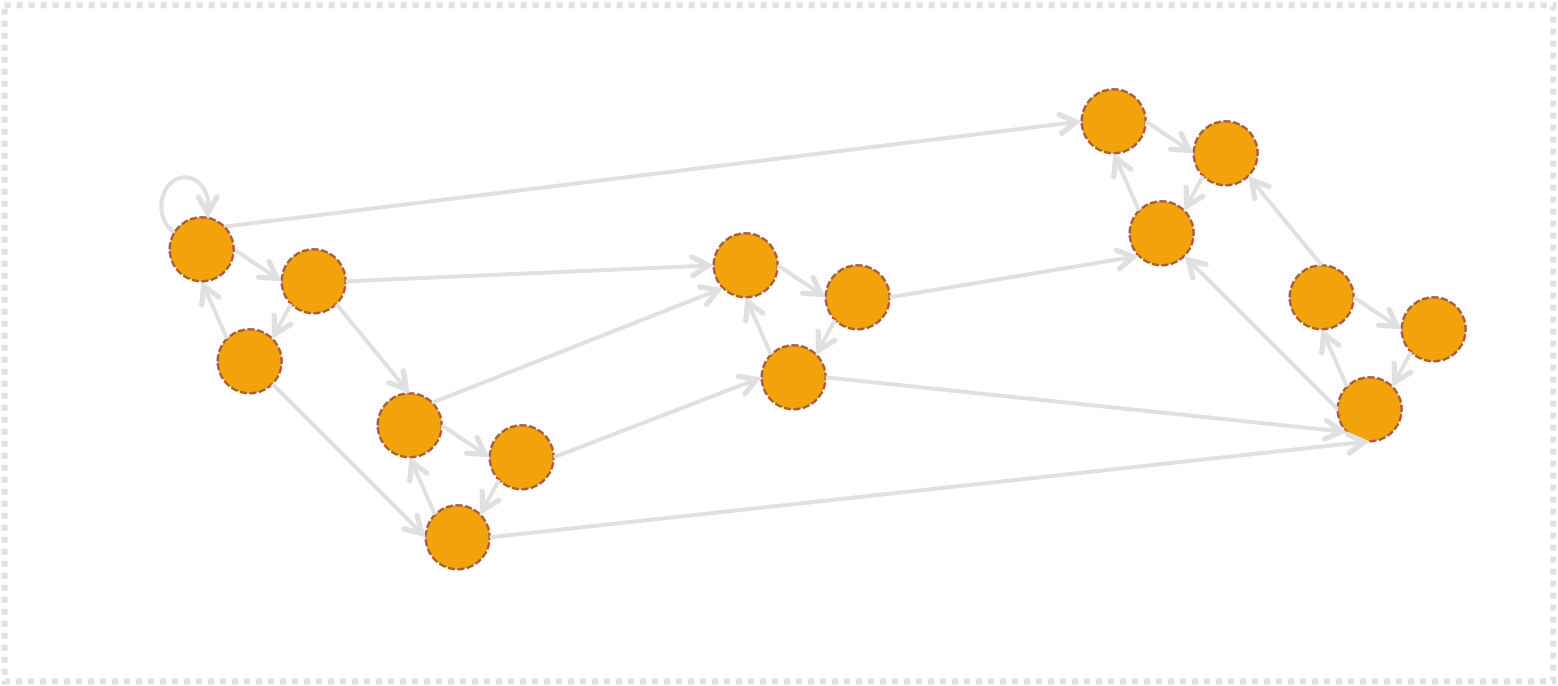
Effective computation of dependencies with a piece-unification test:

R_2 depends on R_1 iff there is a *piece-unifier* of body(R_2) with head(R_1)

Combining Decidable Classes with the Graph of Rule Dependencies

Rules

$R_1 \longrightarrow R_2$: R_1 «may trigger» R_2 (R_2 depends on R_1)



Combining Decidable Classes with the Graph of Rule Dependencies

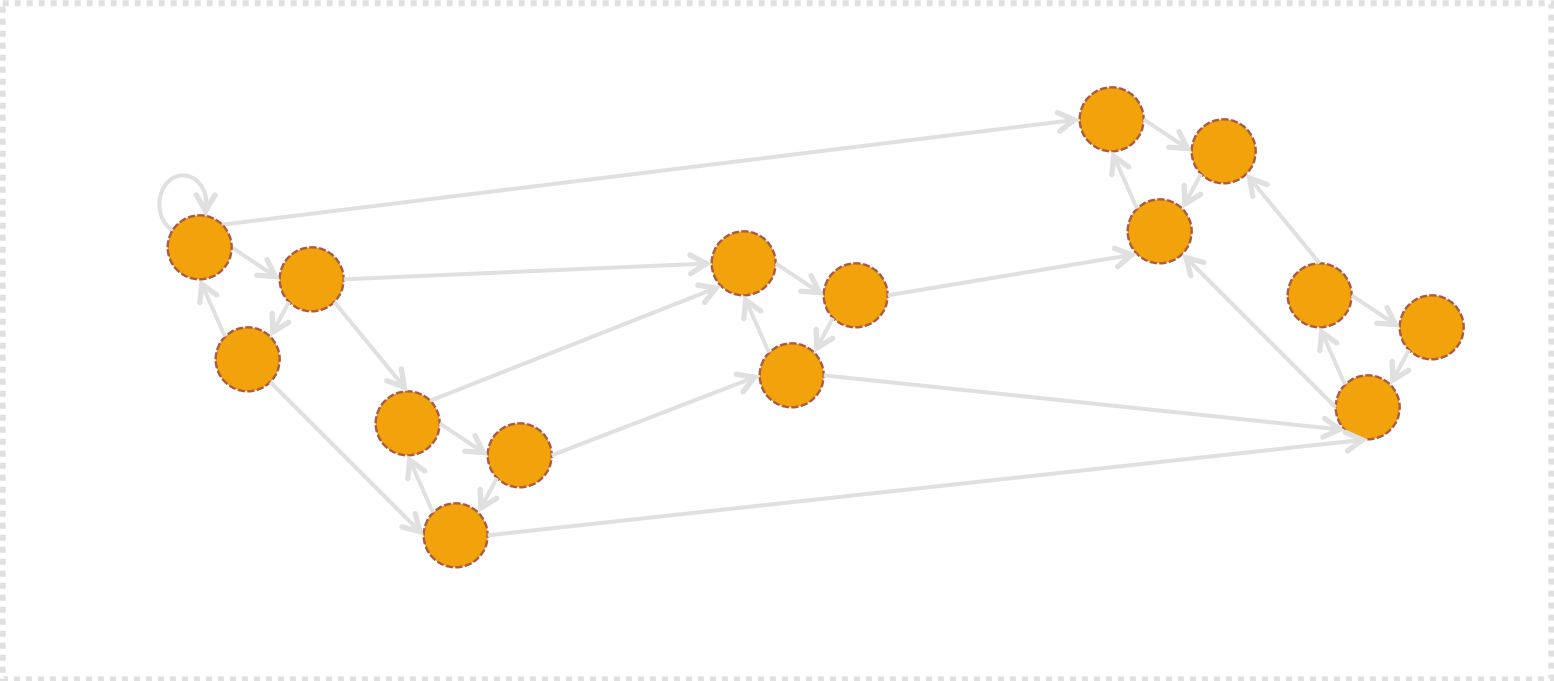
TECHNISCHE UNIVERSITÄT DRESDEN

If GRD(Σ) is without cycle then Σ is both *fes* (thus *bts*) and *fus*

fes = finite chase

fus = finite conjunctive query rewriting

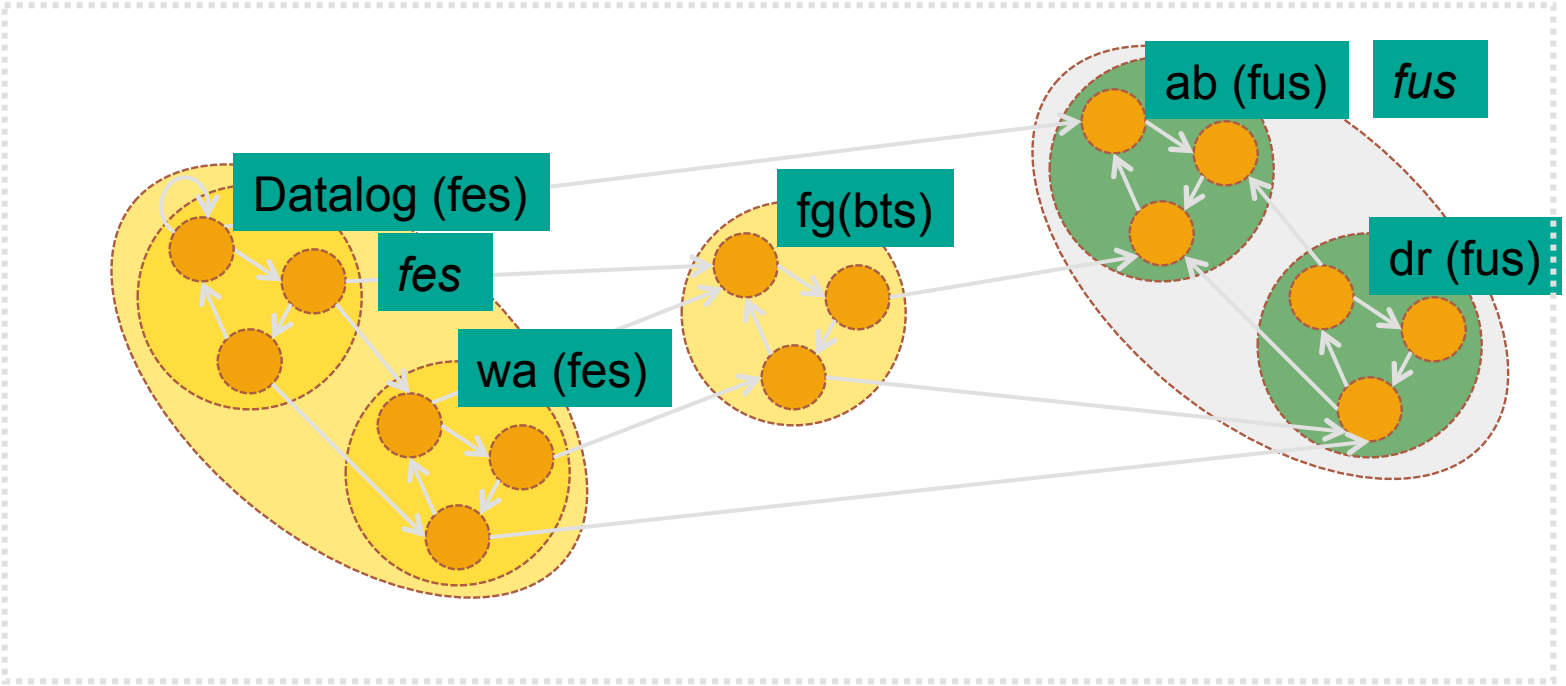
bts = (possibly infinite) bounded-treewidth chase



Combining Decidable Classes with the Graph of Rule Dependencies

If all strongly connected components of GRD(Σ) are *fes* then Σ is *fes* [Baget 2004]

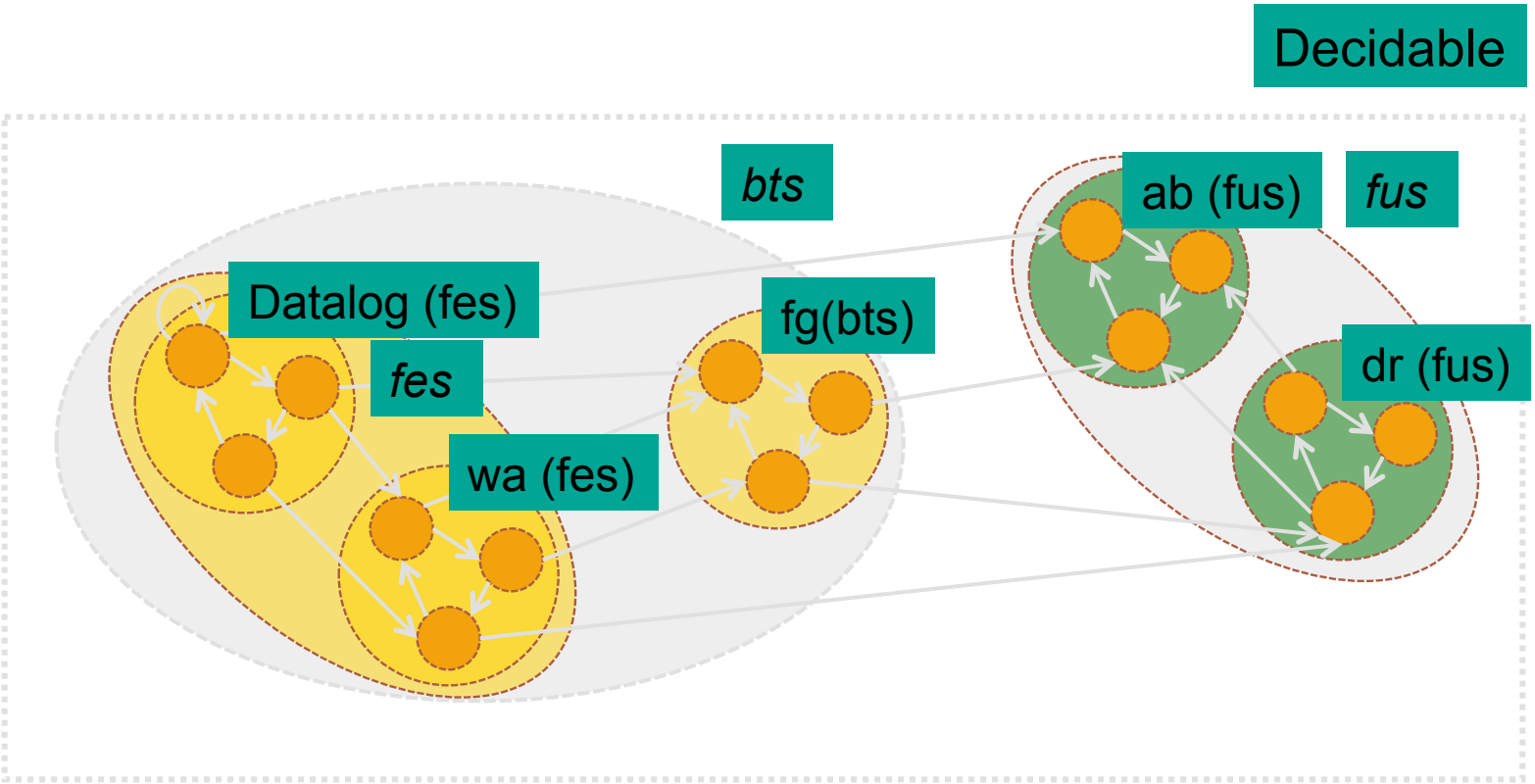
The same holds for *fus (but not for* bts*)*



Combining Decidable Classes with the Graph of Rule Dependencies

Let $\Sigma_1 \rangle \Sigma_2$ be a partition of Σ s.t. no rule of Σ_1 depends on a rule of Σ_2

- If Σ_1 is *fes* and Σ_2 is *bts*, then Σ is *bts*
- If Σ_1 is *bts* and Σ_2 is *fus*, then Σ is decidable



Conclusion

- currently there are two antagonistic computational approaches to ontological query answering:
  - forward chaining starting from data
  - backward chaining starting from query
- both approaches come with (abstract) criteria and concrete syntactic conditions that ensure decidability
- approaches can be “loosely coupled” if ontology allows for adequate separation

- desirable: a common criterion subsuming both
  → future work

Special Thanks & Credits

TECHNISCHE UNIVERSITÄT DRESDEN

Presented contributions joint work with a lot of colleagues (and special thanks to Markus, Marie-Laure and Michaël for letting me use their slides)