TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut Künstliche Intelligenz, Professur Computational Logic

Query Answering over Existential Rules

Sebastian Rudolph
TU Dresden

Tutorial @ ICCL summer school 2016, Rayong

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Learning Outcomes and Prerequisites

A good understanding of:

- the fundamentals of query answering under existential rules
- the main concepts and techniques
- possible research directions

Basic knowledge of:

- first-order logic (syntax and semantics)
- databases (relational model)

Computational Logic Group
Institute for Artificial Intelligence

Outline

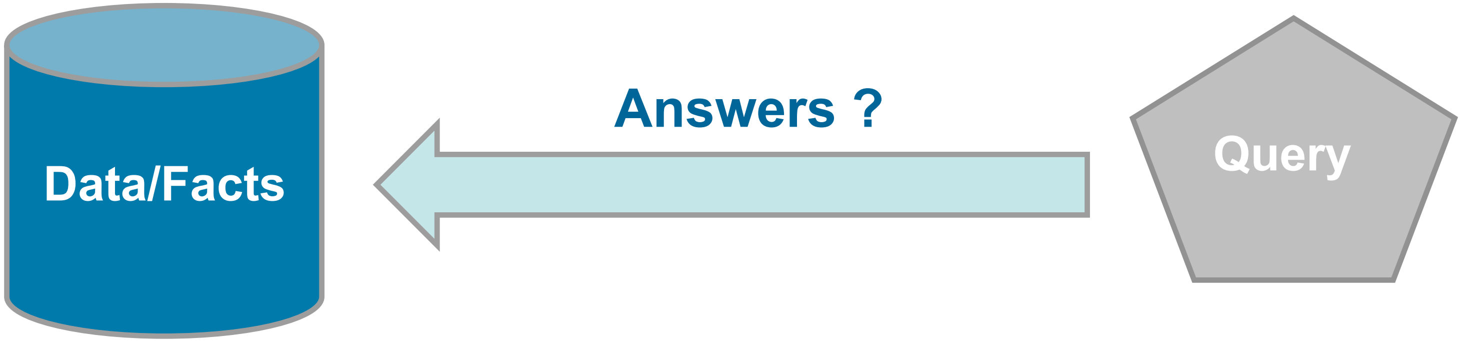
- Classical Query Answering
- Ontological Query Answering: Two Views
- KR View: KB Rewriting into Nice Models
  - Finite models through Acyclicity
  - Bounded-treewidth Models through Guardedness
  - Joining Acyclicity and Guardedness
  - Algorithmic Aspects
- DB View: Query rewriting
- Mixing the Views

Computational Logic Group
Institute for Artificial Intelligence

Outline

- Classical Query Answering
- 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

parentOf

A	B
A	C
C	*x*
…	…

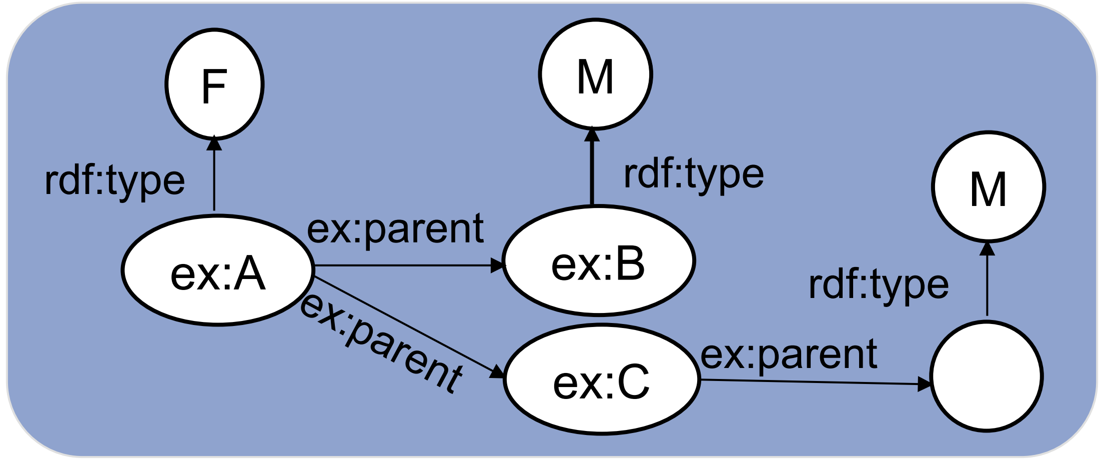
Male

B
x
…

Fem.

A
…

RDF (Semantic Web)

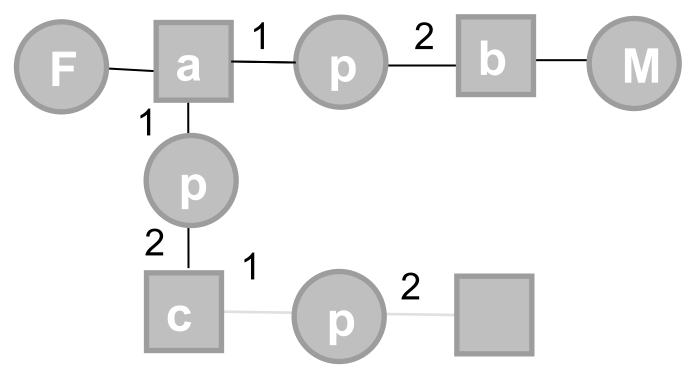


Etc.

Abstraction in First-Order Logic

∃x(parentOf(A,B) ∧ parentOf(A,C) ∧ parentOf(C,x) ∧ F(A) ∧ M(B) ∧ M(x))

Or in **graphs / hypergraphs**



Computational Logic Group
Institute for Artificial Intelligence

Some Notation

TECHNISCHE UNIVERSITÄT DRESDEN

- Our basic vocabulary:
    - A countable set C of constants - domain of a database
    - A countable set N of (labeled) nulls - globally ∃-quantified variables
    - A countable set V of (regular) variables - used in rule and queries

- A term is a constant, null or variable

- An atom has the form $P(t_1, \ldots, t_n)$ - P is an n-ary predicate and t_i's are terms

Computational Logic Group
Institute for Artificial Intelligence

Queries

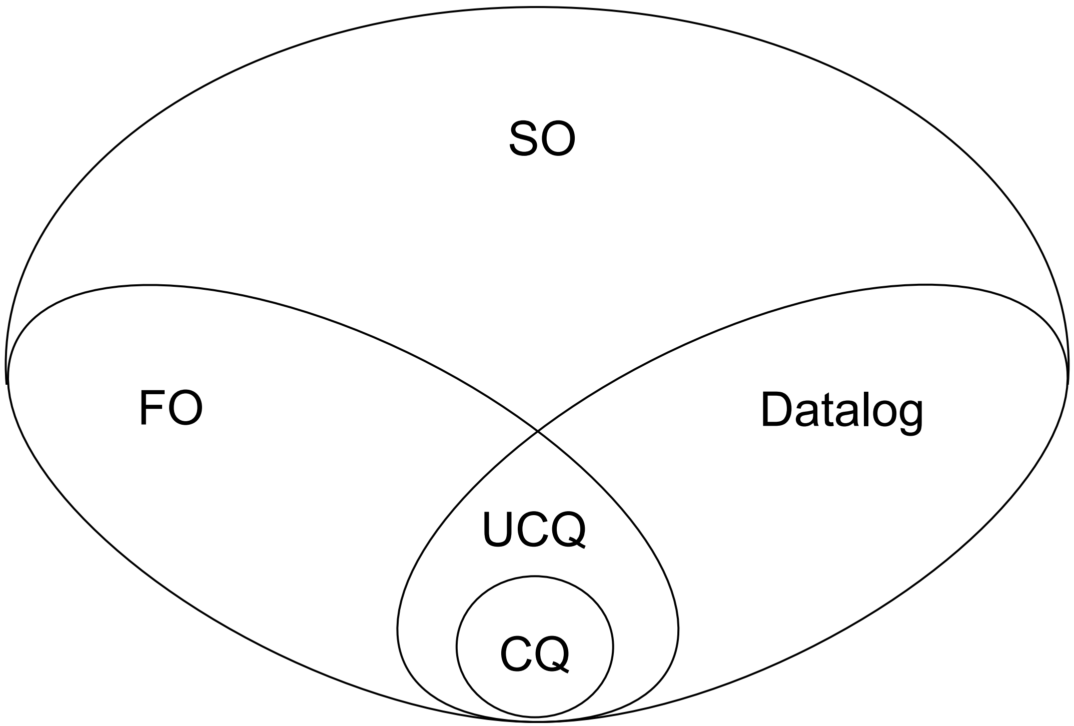
TECHNISCHE UNIVERSITÄT DRESDEN

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

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

Popular Query Languages:

- **conjunctive queries (CQ)**
  … and their unions (UCQ)
- **first-order logic (FOL)**
  Basis of SQL
- **Datalog**
  …and its fragments
- **second-order logic (SOL)**



Conjunctive Queries

TECHNISCHE UNIVERSITÄT DRESDEN

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

	$\exists y\, (\text{Female}(x) \wedge \text{childOf}(x, y) \wedge \text{Female}(y))$	FOL formula
Q(x) =	Female(x), childOf(x, y), Female(y)	Common notation
ans(x) ←	Female(x), childOf(x, y), Female(y)	Datalog notation
SELECT x	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.* CQ-Answering $\equiv_{\text{LOGSPACE}}$ Boolean-CQ-Answering

Computational Logic Group
Institute for Artificial Intelligence

Evaluating Boolean CQs over Data

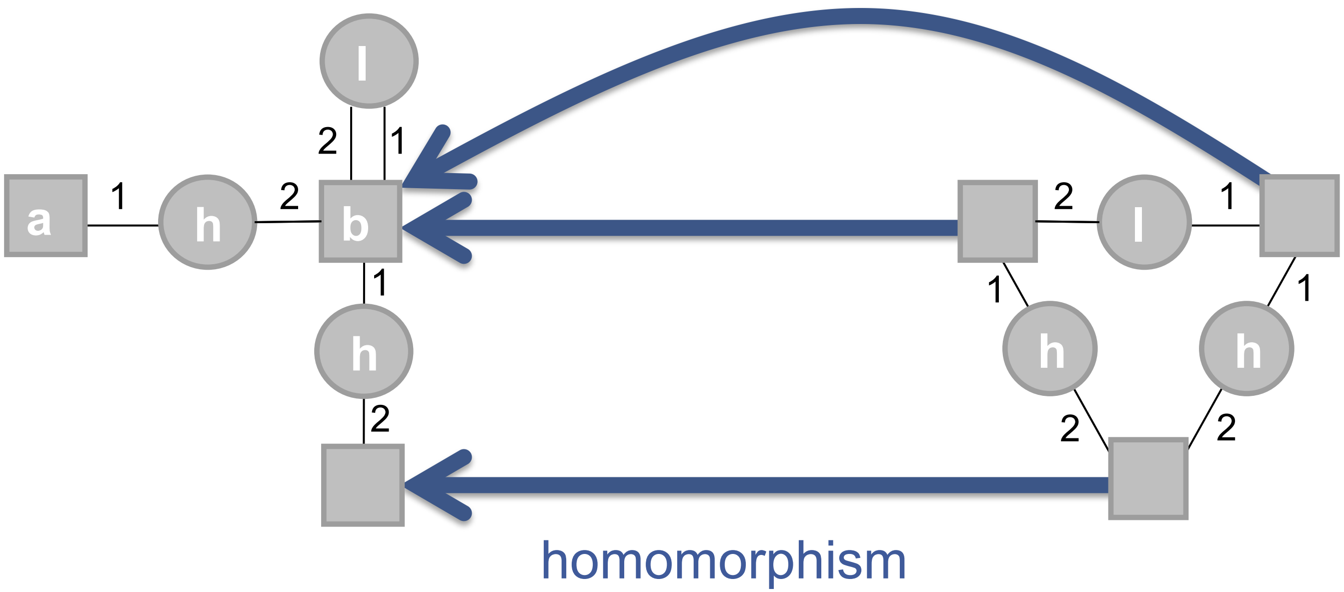
TECHNISCHE UNIVERSITÄT DRESDEN

Data:

∃x (loves(bob,bob)
∧ hates(bob,x)
∧ hates(alice,bob))

Query:

∃xyz (loves(x,y)
∧ hates(x,z)
∧ hates(y,z))



Homomorphism

- Semantics of queries and existential rules definable via the key notion of homomorphism

- A substitution from a set of symbols S to a set of symbols T is a function $h : S \to T$ - h is a set of mappings of the form $s \to t$, where $s \in S$ and $t \in T$

- A homomorphism from a set of atoms A to a set of atoms B is a substitution $h : C \cup N \cup V \to C \cup N \cup V$ such that:

  (i) $t \in C \Rightarrow h(t) = t$ - unique name assumption

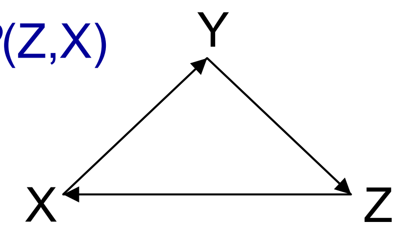
  (ii) $P(t_1, \ldots, t_n) \in A \Rightarrow h(P(t_1, \ldots, t_n)) = P(h(t_1), \ldots, h(t_n)) \in B$

- Can be naturally extended to conjunctions of atoms

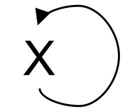
Computational Logic Group
Institute for Artificial Intelligence

Exercise: Find the Homomorphisms

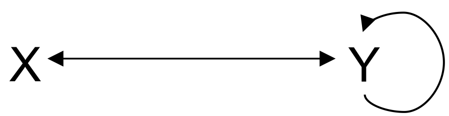
$\varphi_1 \quad = \quad P(X,Y) \wedge P(Y,Z) \wedge P(Z,X)$



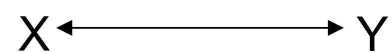
$\varphi_2 \quad = \quad P(X,X)$



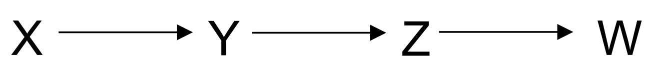
$\varphi_3 \quad = \quad P(X,Y) \wedge P(Y,X) \wedge P(Y,Y)$



$\varphi_4 \quad = \quad P(X,Y) \wedge P(Y,X)$

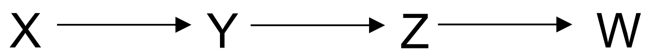


$\varphi_5 \quad = \quad P(X,Y) \wedge P(Y,Z) \wedge P(Z,W)$

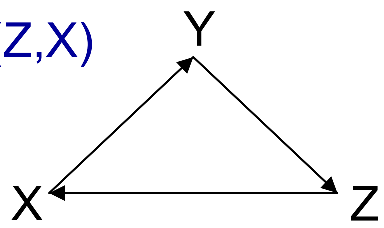


Exercise: Find the Homomorphisms

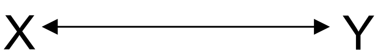
$\varphi_5 = P(X,Y) \wedge P(Y,Z) \wedge P(Z,W)$



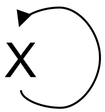
$\varphi_1 = P(X,Y) \wedge P(Y,Z) \wedge P(Z,X)$



$\varphi_4 = P(X,Y) \wedge P(Y,X)$



$\varphi_2 = P(X,X)$



$\varphi_3 = P(X,Y) \wedge P(Y,X) \wedge P(Y,Y)$

Outline

- Classical Query Answering
- 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

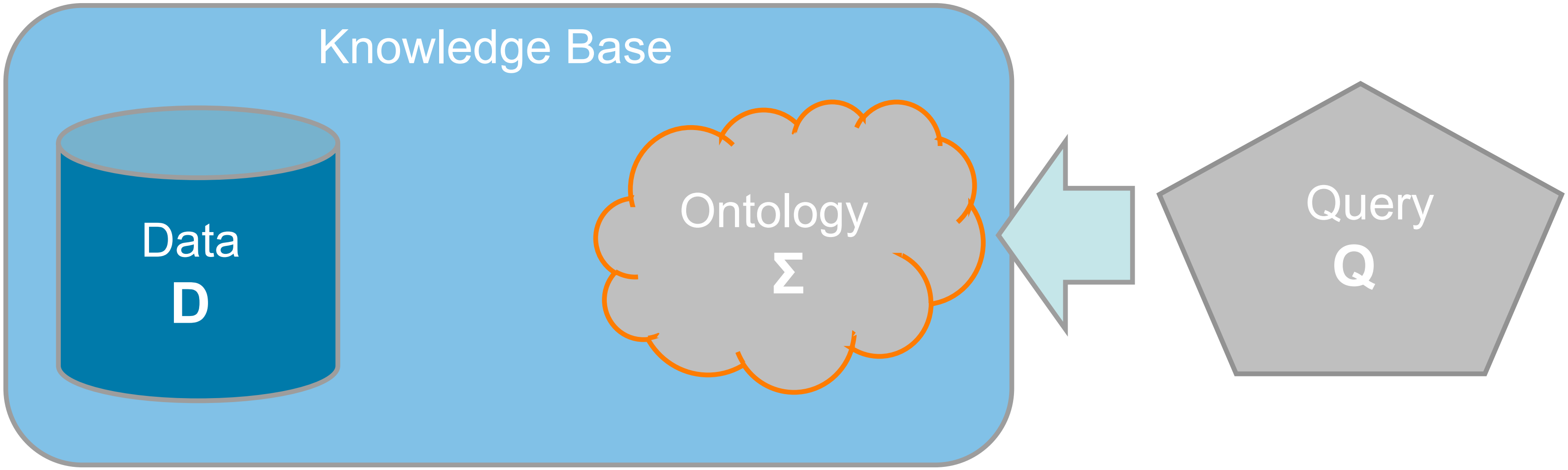
TECHNISCHE UNIVERSITÄT DRESDEN

The Plain View:

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

Views on Ontological Query Answering

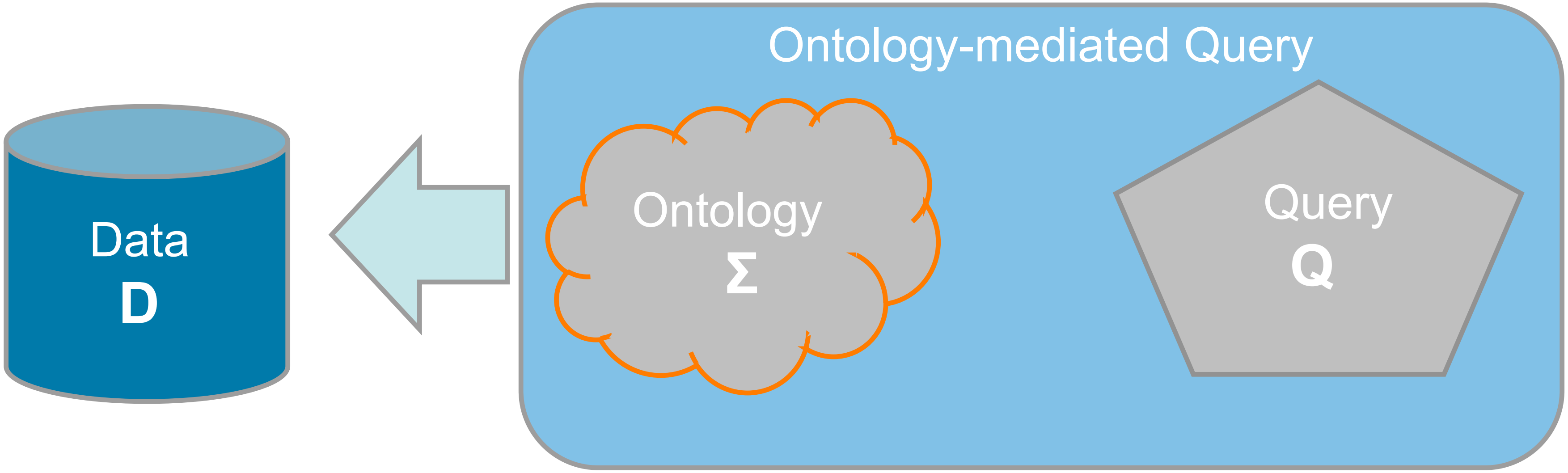
The Knowledge Representation View:



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

Views on Ontological Query Answering

The Database View:



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

Our Main Showcase:

Our Main Showcase:

TECHNISCHE UNIVERSITÄT DRESDEN

Existential Rules

« Value Invention »

$$\forall X\ \forall Y\ (\ B[X, Y] \rightarrow \exists Z\ H[X, Z]\ )$$

Body (B[X, Y]) — Head (H[X, Z])

X, Y, Z : *tuples of variables*

Any conjunction of atoms (on variables and constants)

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

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

- Same as Tuple Generating Dependencies (TGDs)
- See also Datalog+/-
- Same as the logical translation of Conceptual Graph rules
- Generalize lightweight DLs used for OBQA

Semantics of Existential Rules

- A database D is a model of the rule

  $$\sigma = \forall X \forall Y\, (B[X,Y] \rightarrow \exists Z\, H[X,Z])$$

  written as $D \models \sigma$, if the following holds:

  whenever there exists a homomorphism h such that $h(B[X,Y]) \subseteq D$,

  then there exists $g \supseteq h_{|X}$ such that $g(H[X,Z]) \subseteq D$

  $\{t \rightarrow h(t) \mid t \in X\}$ – the restriction of h to X

- Given a set Σ of existential rules, D is a model of Σ, written as $D \models \Sigma$, if the following holds: for each $\sigma \in \Sigma$, $D \models \sigma$

- $D \models \Sigma$ iff D is a model of the first-order theory $\bigwedge_{\sigma \in \Sigma} \sigma$

Existential Rules vs. DLs

Existential rules and DLs rely on first-order semantics - comparable formalisms

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

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

Existential Rules vs. DLs

Existential rules and DLs rely on first-order semantics - comparable formalisms

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

EL Axioms	Existential Rules
$A \sqsubseteq B$	$\forall x\ (A(x) \rightarrow B(x))$
$A \sqcap B \sqsubseteq C$	$\forall x\ (A(x) \land B(x) \rightarrow C(x))$
$A \sqsubseteq \exists R.B$	$\forall x\ (A(x) \rightarrow \exists y\ (R(x,y) \land B(y)))$
$\exists R.B \sqsubseteq A$	$\forall x \forall y\ (R(x,y) \land B(y) \rightarrow A(x))$

Existential Rules vs. DLs

- Several Horn DLs (without disjunction) can be expressed via existential rules

- But, existential rules can express more

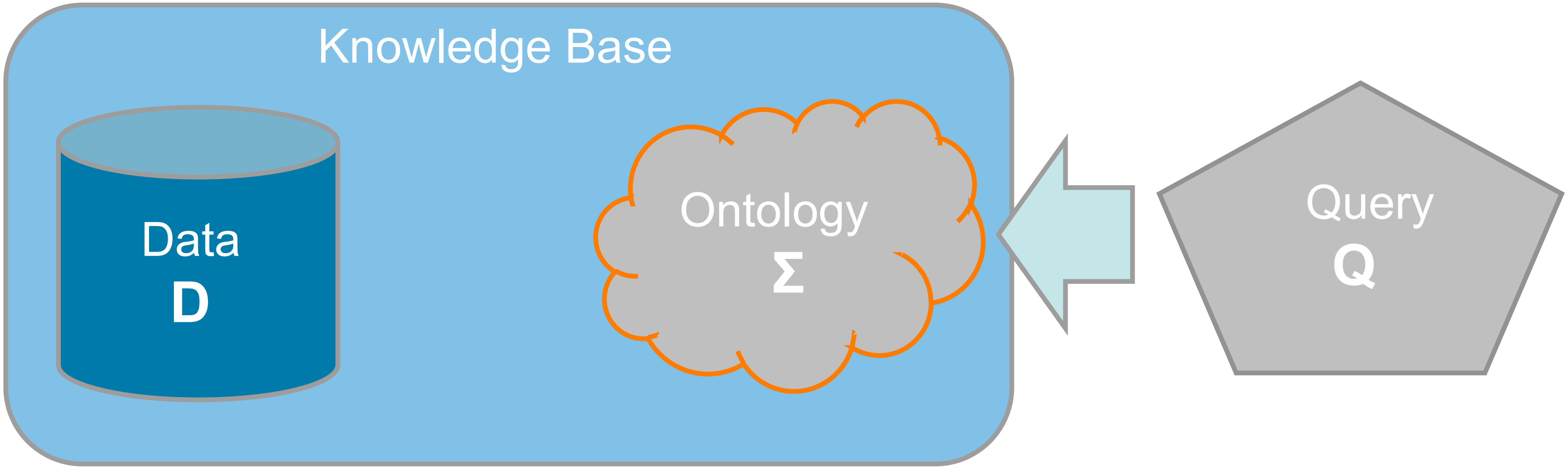
$$\forall x \forall y\, (\mathit{siblingOf}(x,y) \rightarrow \exists z\, (\mathit{parentOf}(z,x) \land \mathit{parentOf}(z,y)))$$

- Higher arity predicates allow for more flexibility
  - Direct translation of database relations
  - Adding contextual information is easy (provenance, trust, etc.)

Outline

- Classical Query Answering
- 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



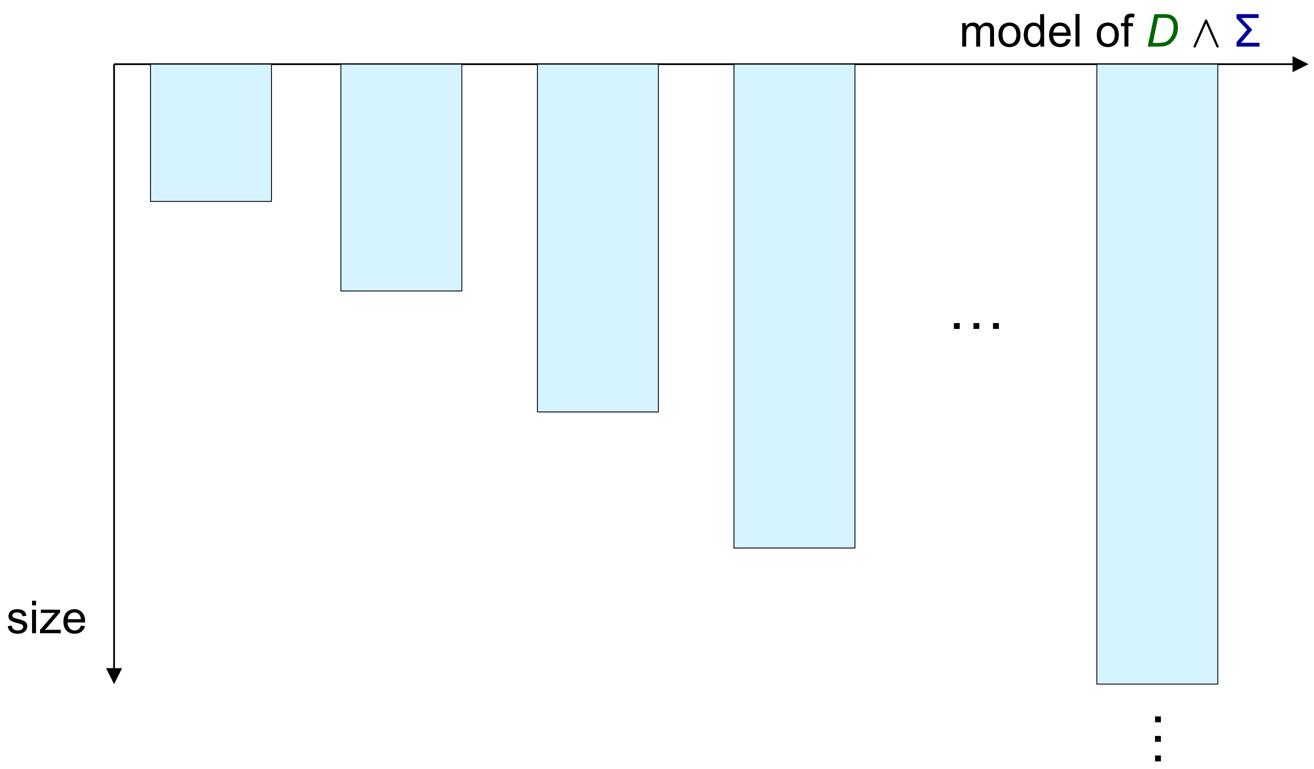
$D \wedge \Sigma \models Q$

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 Two Dimensions of Infinity

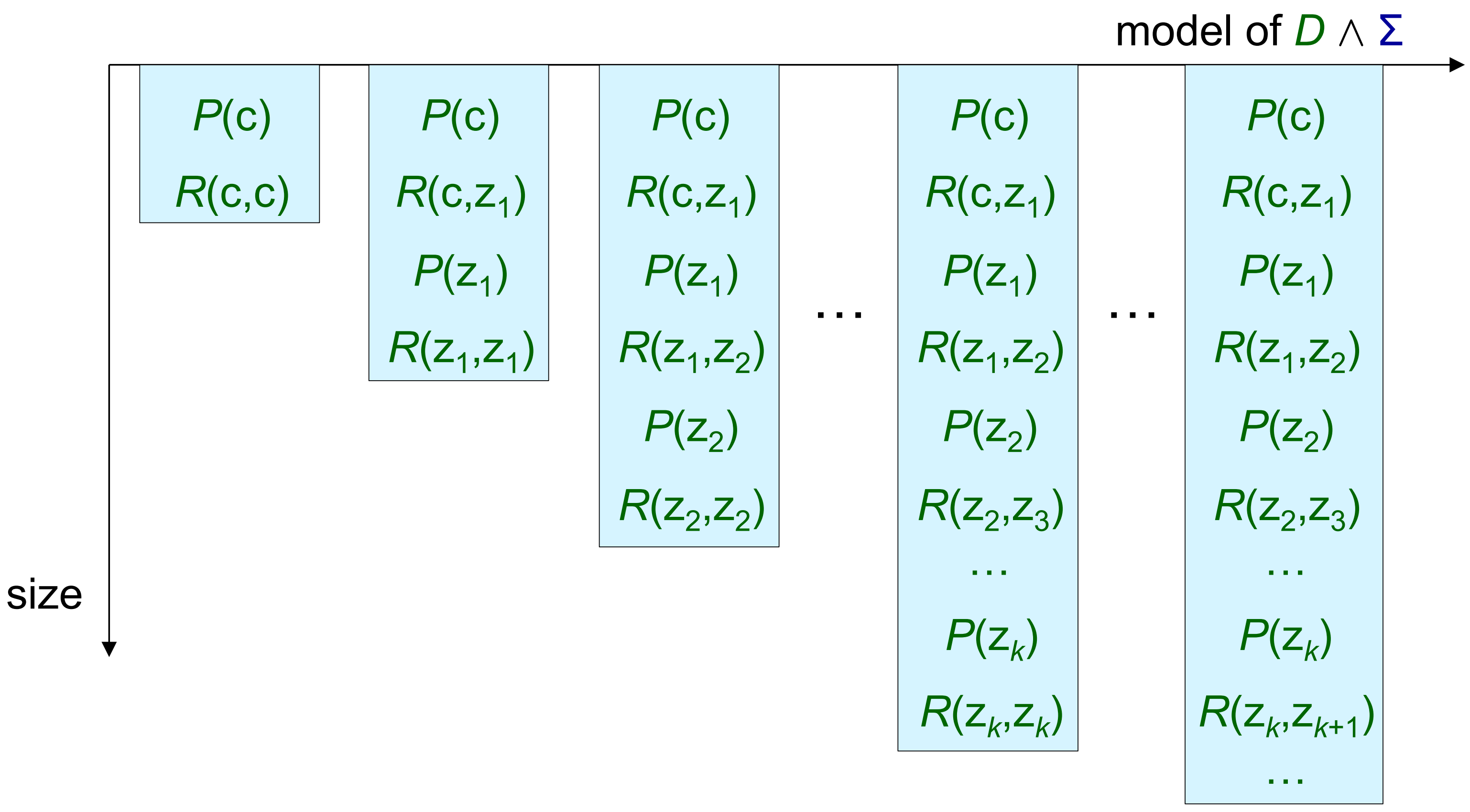
Consider the database *D*, and the set of existential rules Σ



D ∧ Σ admits infinitely many models, and each one may be of infinite size

The Two Dimensions of Infinity

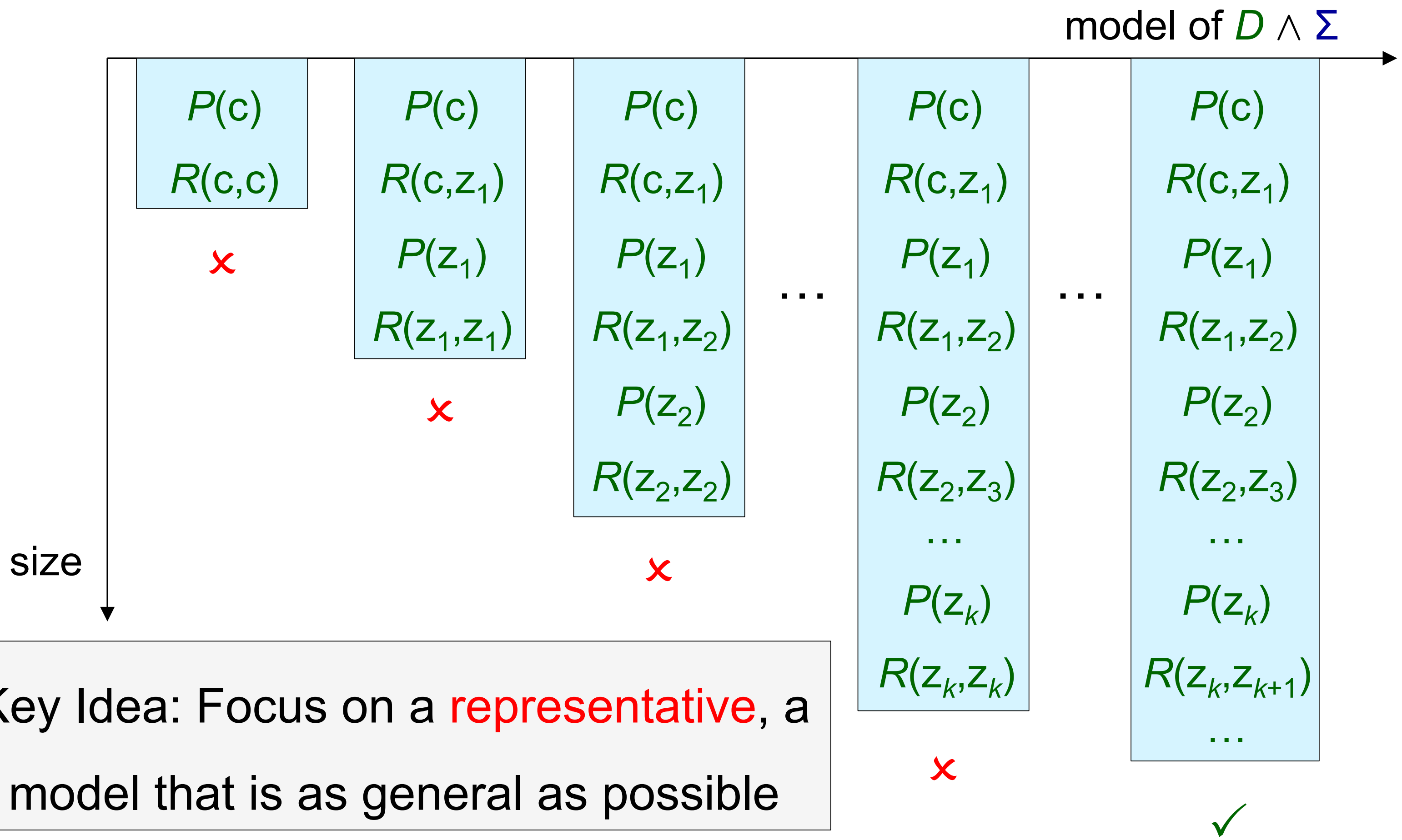
$D = \{P(c)\}$ $\Sigma = \{\forall x\, (P(x) \rightarrow \exists y\, (R(x,y) \land P(y)))\}$



$z_1, z_2, z_3, \ldots$ are nulls of N

Taming the First Dimension of Infinity

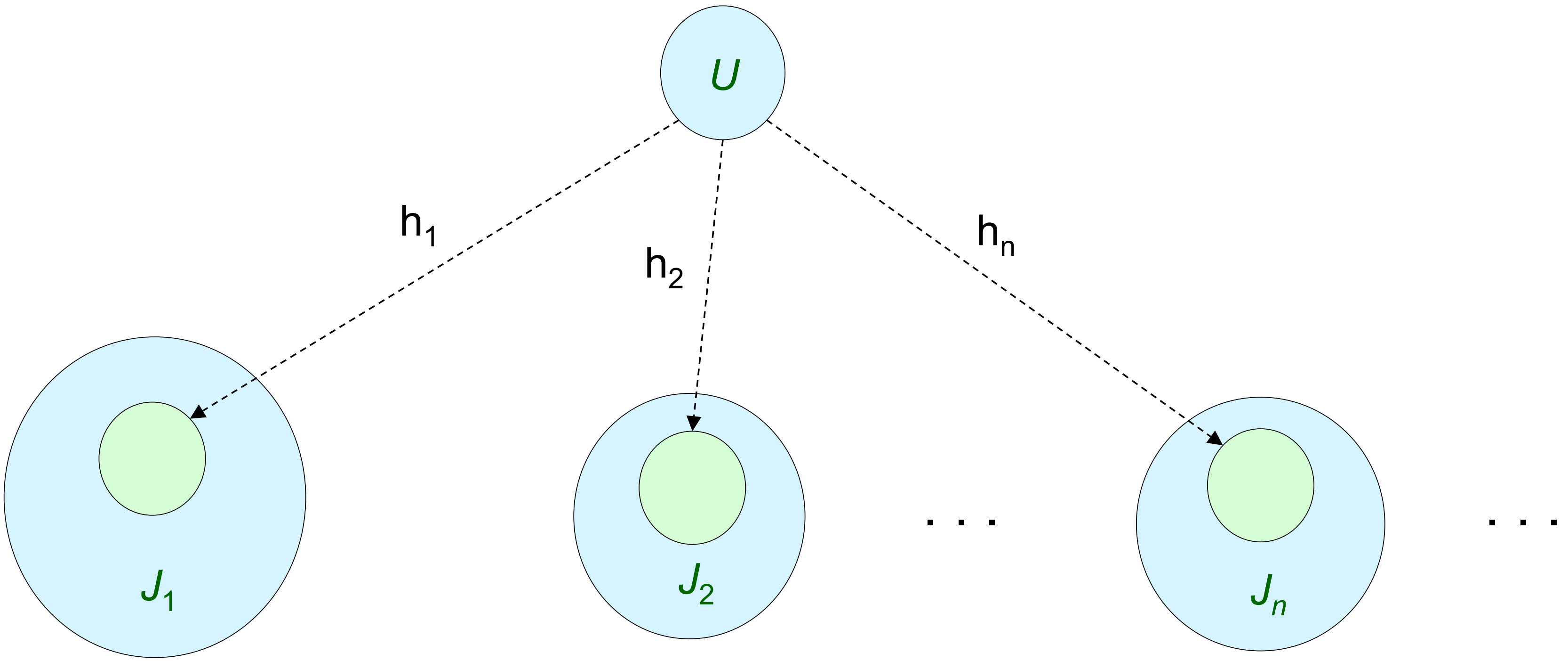
$D = \{P(c)\}$ $\Sigma = \{\forall x\, (P(x) \rightarrow \exists y\, (R(x,y) \land P(y)))\}$



Key Idea: Focus on a representative, a model that is as general as possible

Universal (also: Canonical) Models

TECHNISCHE UNIVERSITÄT DRESDEN



An instance U is a universal model of $D \wedge \Sigma$ if the following holds:

1. U is a model of $D \wedge \Sigma$

2. $\forall J \in$ models($D \wedge \Sigma$), there exists a homomorphism h_J such that $h_J(U) \subseteq J$

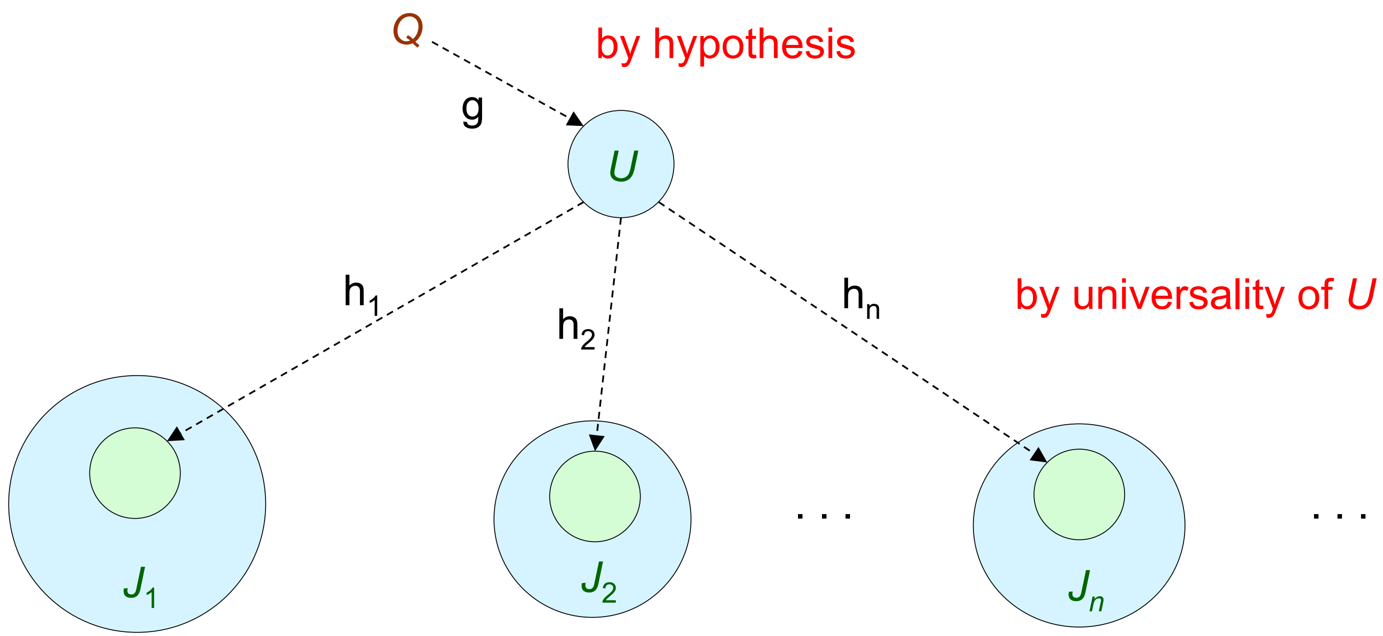
Query Answering via Universal Models

TECHNISCHE UNIVERSITÄT DRESDEN

Theorem: $D \wedge \Sigma \models Q$ iff $U \models Q$, where U is a universal model of $D \wedge \Sigma$

Proof: (⇒) Trivial since, for every $J \in \text{models}(D \wedge \Sigma)$, $J \models Q$

(⇐) By exploiting the universality of U



$\forall J \in \text{models}(D \wedge \Sigma), \exists h_J$ such that $h_J(g(Q)) \subseteq J \quad \Rightarrow \quad \forall J \in \text{models}(D \wedge \Sigma), J \models Q$

$\Rightarrow \quad D \wedge \Sigma \models Q$

Computational Logic Group
Institute for Artificial Intelligence

The Chase Procedure

- Fundamental algorithmic tool used in databases

- It has been applied to a wide range of problems:
  - Checking containment of queries under constraints
  - Computing data exchange solutions
  - Computing certain answers in data integration settings
  - …

… what's the reason for the ubiquity of the chase in databases?

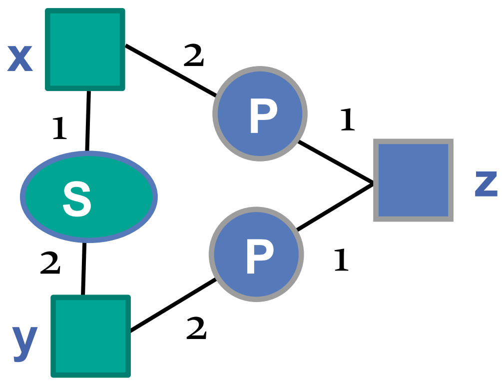
it constructs universal models

Computational Logic Group
Institute for Artificial Intelligence

Value Invention (Generation of Fresh Existentials)

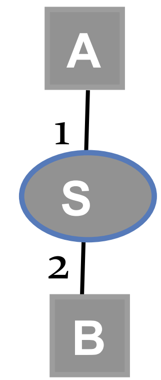
R = ∀x ∀y (siblingOf(x,y) → ∃ z (parentOf(z,x) ∧ parentOf(z,y)))

D = siblingOf(A,B)



h: body → F

h =*{(x,A), (y,B)}*



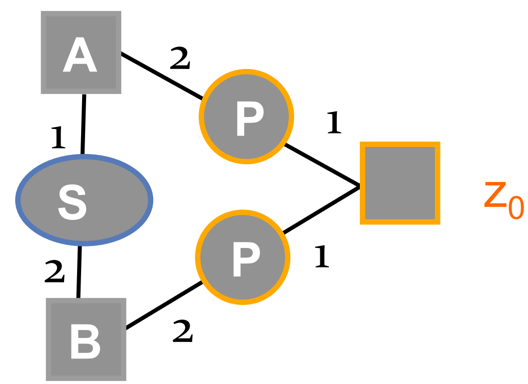
A rule *body* → *head* is applicable to a database D if

there is a homomorphism *h* from *body* to D

The resulting database is D' = D ∪ *h*(*head*)

[with new names for existential variables of *head*]

D' = ∃ z_0 (siblingOf(A,B)
∧ parentOf(z_0,A) ∧ parentOf(z_0,B))



The Chase

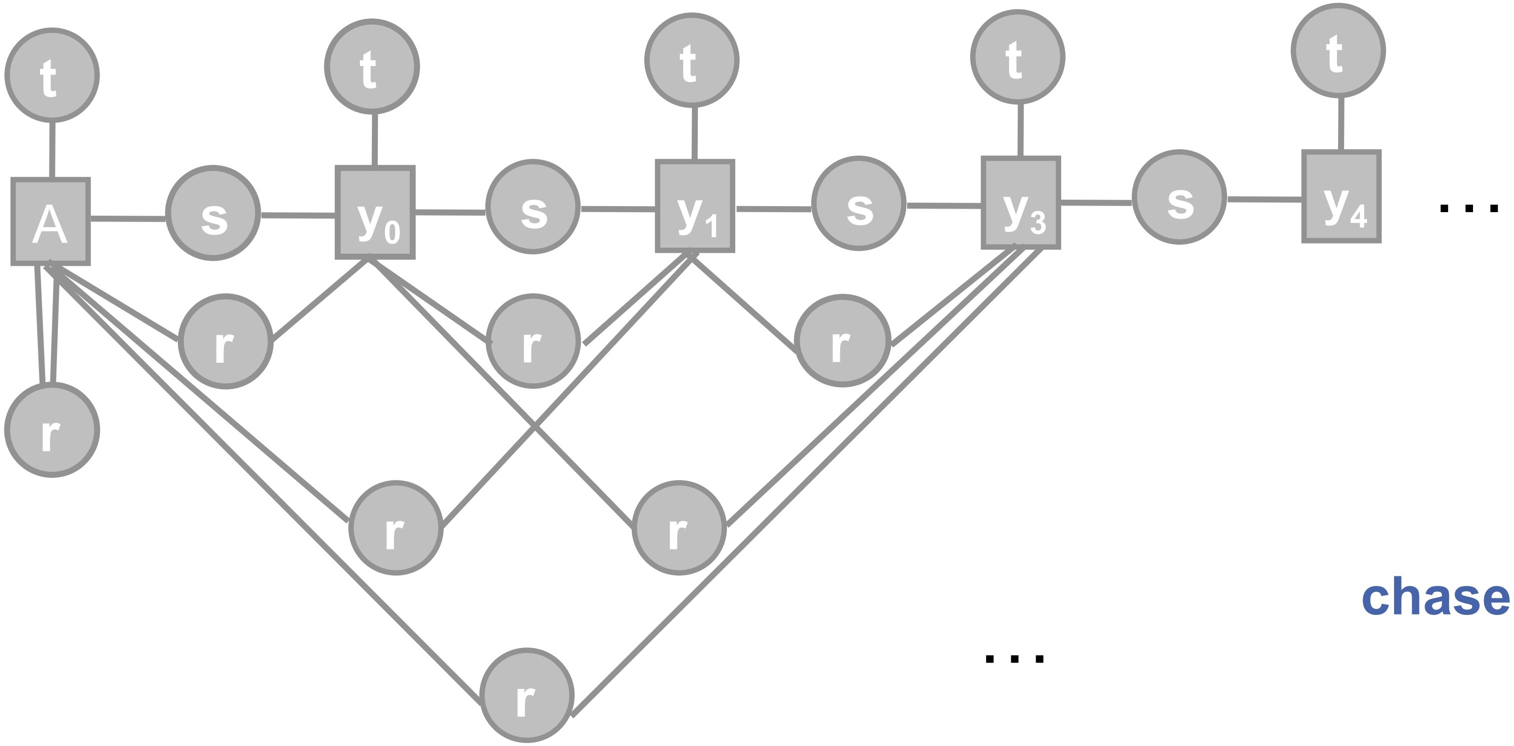
TECHNISCHE UNIVERSITÄT DRESDEN

Σ

t(x) → s(x,y) ∧ t(y)

t(x) ∧ t(y) → r(x,y)

D= {t(A)}



chase(D, Σ)

Computational Logic Group
Institute for Artificial Intelligence