Ontology-Based Data Access

Roman Kontchakov

Dept. of Computer Science and Inf. Systems, Birkbeck, University of London

http://www.dcs.bbk.ac.uk/~roman

acknowledgements:

Alessandro Artale, Diego Calvanese, Carsten Lutz, Mariano Rodríguez-Muro, David Toman, Frank Wolter and Michael Zakharyaschev

Data Management: New Challenges

- Statoil (Norway)

  many databases, e.g., EPDS (Exploration and Production Data Store

  over 1500 tables

  historical exploration data (e.g., layers of rocks, porosity),

  production logs, maps, etc.

  business information such as license areas and companies

direct data access by engineers (and geologists in particular) is often **challenging**

Data Management: New Challenges

- **Statoil** (Norway)

  many databases, e.g., EPDS (Exploration and Production Data Store

  over 1500 tables

  historical exploration data (e.g., layers of rocks, porosity),
  production logs, maps, etc.

  business information such as license areas and companies

direct data access by engineers (and geologists in particular) is often **challenging**

- **Siemens Energy** (Germany)

  power generation facilities (gas and steam turbines)

  50 service centres linked to a common database

  each turbine

  2000 sensors

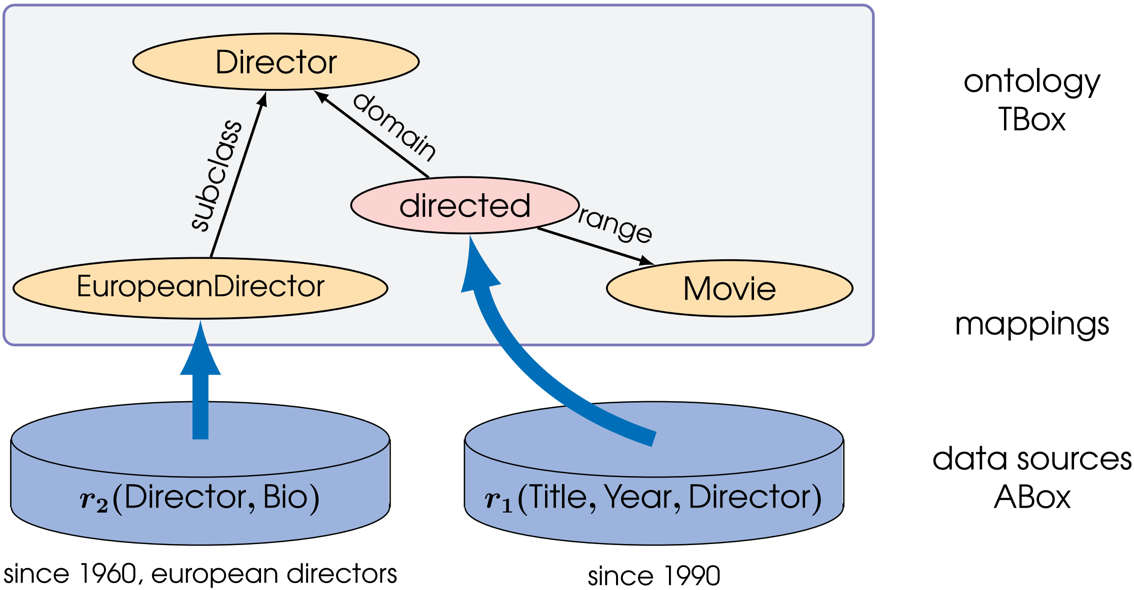
  150 tables

  30 GB of data is generated daily (hundreds of terabytes in total)

Ontology-Based Data Access

Aim: to achieve **logical transparency** in accessing data

- hide from the user where and how data is stored
- present only a **conceptual view** of the data
- **query** the data sources through the **conceptual model** using **RDBMSs**



Issues in OBDA

- what is the right ontology language?
  - there is a wide spectrum of languages that differ in
    expressive power and **complexity** of inference
  - scalability to very large amounts of data is key

Issues in OBDA

- what is the right ontology language?
  - there is a wide spectrum of languages that differ in
    expressive power and **complexity** of inference
  - scalability to very large amounts of data is key

- what is the query language?

Issues in OBDA

- what is the right ontology language?
  - there is a wide spectrum of languages that differ in **expressive power** and **complexity** of inference
  - scalability to very large amounts of data is key
- what is the query language?
- how do we connect ontologies to data sources?
  - multiple data sources and ontologies

Issues in OBDA

- what is the right ontology language?
  - there is a wide spectrum of languages that differ in
    expressive power and **complexity** of inference
  - scalability to very large amounts of data is key

- what is the query language?

- how do we connect ontologies to data sources?
  - multiple data sources and ontologies

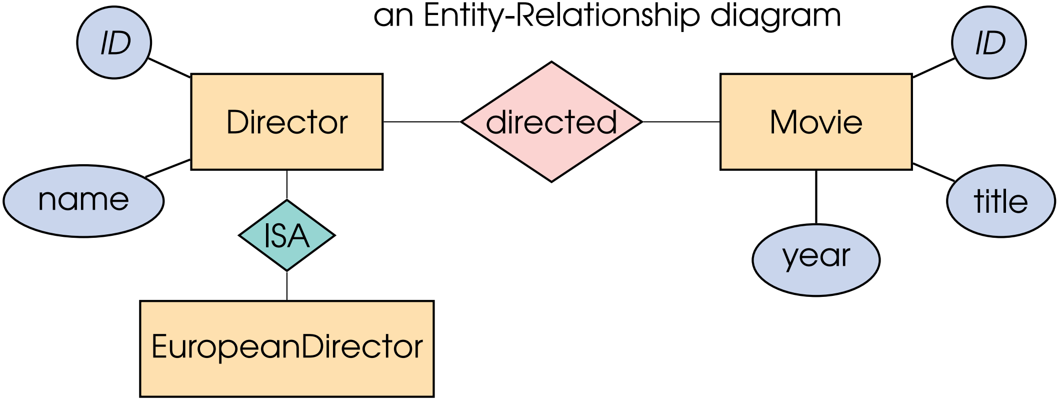
- available tools?
  - sound and complete reasoning
  - practical scalability

Part 1

Databases and Logic

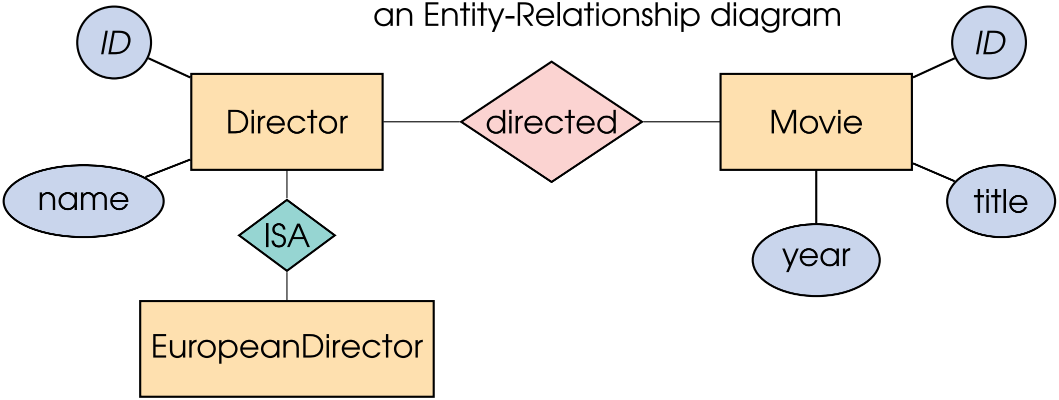
Databases: Specifying Schema

an Entity-Relationship diagram



Databases: Specifying Schema

an Entity-Relationship diagram

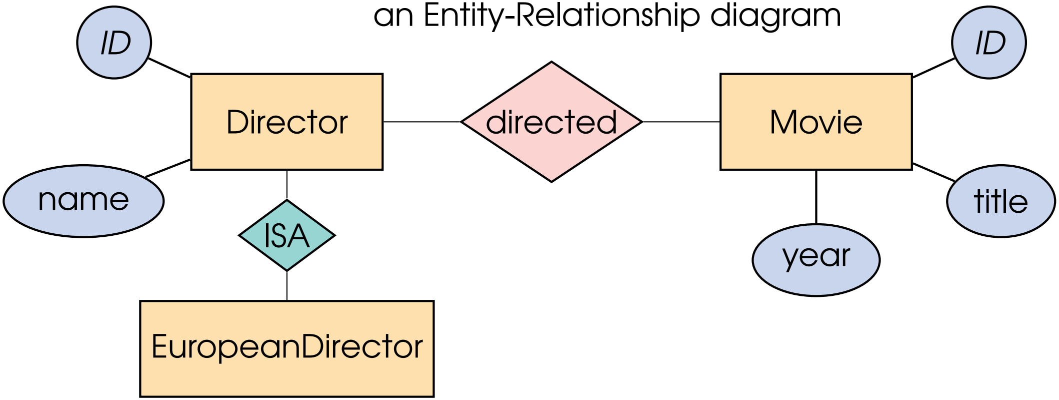


integrity constraints or dependencies (in the language of FO):

$\forall d\ (\exists m\ \text{directed}(d, m) \to \exists n\ \text{Director}(d, n))$ (foreign keys, inclusion or
$\forall m\ (\exists d\ \text{directed}(d, m) \to \exists ty\ \text{Movie}(m, t, y))$ tuple-generating dependencies,
$\forall dn\ (\text{EuropeanDirector}(d, n) \to \text{Director}(d, n))$ **TGDs**)

Databases: Specifying Schema

an Entity-Relationship diagram



integrity constraints or dependencies (in the language of FO):

$\forall d\,(\exists m\,\text{directed}(d,m) \to \exists n\,\text{Director}(d,n))$
$\forall m\,(\exists d\,\text{directed}(d,m) \to \exists ty\,\text{Movie}(m,t,y))$
$\forall dn\,(\text{EuropeanDirector}(d,n) \to \text{Director}(d,n))$

(foreign keys, inclusion or tuple-generating dependencies, **TGDs**)

$\forall d n_1 n_2\,(\text{Director}(d,n_1) \wedge \text{Director}(d,n_2) \to (n_1 = n_2))$
$\forall m t_1 t_2 y_1 y_2\,(\text{Movie}(m,t_1,y_1) \wedge \text{Movie}(m,t_2,y_2) \to (t_1 = t_2))$

(keys, functional or equality-generating dependencies, **EGDs**)

Databases: Data and the Closed World Assumption

data is **completely** specified (**closed world assumption**) and is typically **large**

what is specified is true, everything else is false

Databases: Data and the Closed World Assumption

data is **completely** specified (**closed world assumption**) and is typically **large**

what is specified is true, everything else is false

data:

$$\text{Director} = \{\ (\mathbf{0}, \text{“peter”}),\ \ (\mathbf{1}, \text{“quentin”}),\ \ (\mathbf{2}, \text{“danny”})\ \}$$

$$\text{EuropeanDirector} = \{\ (\mathbf{0}, \text{“peter”}),\ \ (\mathbf{2}, \text{“danny”})\ \}$$

$$\text{Movie} = \{\ (\mathbf{10}, \text{“DC”}),\ \ (\mathbf{11}, \text{“TS”})\ \}$$

$$\text{directed} = \{\ (\mathbf{0}, \mathbf{10}),\ \ (\mathbf{2}, \mathbf{11})\ \}$$

query: $\boldsymbol{q}(\boldsymbol{n}) = \exists \boldsymbol{d}\, \text{Director}(\boldsymbol{d}, \boldsymbol{n})$

Databases: Data and the Closed World Assumption

data is **completely** specified (**closed world assumption**) and is typically **large**

what is specified is true, everything else is false

data:

$$\text{Director} = \{\ (\mathbf{0}, \text{"peter"}),\ \ (\mathbf{1}, \text{"quentin"}),\ \ (\mathbf{2}, \text{"danny"})\ \}$$
$$\text{EuropeanDirector} = \{\ (\mathbf{0}, \text{"peter"}),\ \ (\mathbf{2}, \text{"danny"})\ \}$$
$$\text{Movie} = \{\ (\mathbf{10}, \text{"DC"}),\ \ (\mathbf{11}, \text{"TS"})\ \}$$
$$\text{directed} = \{\ (\mathbf{0}, \mathbf{10}),\ \ (\mathbf{2}, \mathbf{11})\ \}$$

query: $\boldsymbol{q}(\boldsymbol{n}) = \exists \boldsymbol{d}\, \text{Director}(\boldsymbol{d}, \boldsymbol{n})$

answer: $\{\ \text{"peter"},\ \ \text{"quentin"},\ \ \text{"danny"}\ \}$

NB: not having $(\mathbf{2}, \text{"danny"})$ in Director would violate the integrity constraint

$$\forall \boldsymbol{dn}\ (\text{EuropeanDirector}(\boldsymbol{d}, \boldsymbol{n}) \rightarrow \text{Director}(\boldsymbol{d}, \boldsymbol{n}))$$

Databases: Query Languages

SQL $\approx$ domain-independent **FO queries**:

database predicates + logical connectives $\lor$, $\land$, $\neg$ + quantifiers $\forall$, $\exists$

Databases: Query Languages

SQL $\approx$ domain-independent **FO queries**:

database predicates + logical connectives $\lor$, $\land$, $\neg$ + quantifiers $\forall$, $\exists$

data $\mathcal{D}$ = FO interpretation $\mathcal{I}_{\mathcal{D}}$ (CWA)

$\vec{a}$ is an **answer** to $q(\vec{x})$ iff $\mathcal{I}_{\mathcal{D}} \models q(\vec{a})$

Databases: Query Languages

SQL $\approx$ domain-independent **FO queries**:

database predicates + logical connectives $\lor$, $\land$, $\neg$ + quantifiers $\forall$, $\exists$

data $\mathcal{D}$ = FO interpretation $\mathcal{I}_{\mathcal{D}}$ (CWA)

$\vec{a}$ is an **answer** to $q(\vec{x})$ iff $\mathcal{I}_{\mathcal{D}} \models q(\vec{a})$

Select-Project-Join (SPJ) = **conjunctive queries** (CQs):

database predicates + $\land$ + $\exists$

database engines are optimised for CQs

Example: SELECT M.title, D.name
FROM Movie M, Directed MD, Director D
WHERE M.id = MD.movieId AND MD.directorId = D.id AND M.Year = 1982

Databases: Query Languages

SQL $\approx$ domain-independent **FO queries**:

database predicates + logical connectives $\vee, \wedge, \neg$ + quantifiers $\forall, \exists$

data $\mathcal{D}$ = FO interpretation $\mathcal{I}_{\mathcal{D}}$ (CWA)

$\vec{a}$ is an **answer** to $q(\vec{x})$ iff $\mathcal{I}_{\mathcal{D}} \models q(\vec{a})$

Select-Project-Join (SPJ) = **conjunctive queries** (CQs):

database predicates + $\wedge$ + $\exists$

database engines are optimised for CQs

Example: SELECT M.title, D.name
FROM Movie M, Directed MD, Director D
WHERE M.id = MD.movieId AND MD.directorId = D.id AND M.Year = 1982

Datalog notation:

$$\underbrace{q(\vec{x})}_{\text{head}} \leftarrow \underbrace{P_1(\vec{z}_1), \dots, P_k(\vec{z}_k)}_{\text{body}}$$

where each $\vec{z}_i$ is a vector, which may contain **answer variables** $\vec{x}$ and existentially quantified variables $\vec{y}$ (implicit)

Example: $q(t, n) \leftarrow$ Movie$(m, t, 1982)$, directed(m, d), director(d, n)

Why do Databases Work?

query answering problem (as a recognition problem):

given a finite data $\mathcal{D}$, a query $q(\vec{x})$ and a tuple $\vec{a}$,
decide whether $\mathcal{I}_{\mathcal{D}} \models q(\vec{a})$

$\mathcal{I}_{\mathcal{D}}$ makes the facts in $\mathcal{D}$ true (and only them)

what is the complexity of CQ answering?

Why do Databases Work?

query answering problem (as a recognition problem):

> given a finite data $\mathcal{D}$, a query $q(\vec{x})$ and a tuple $\vec{a}$,
> decide whether $\mathcal{I}_{\mathcal{D}} \models q(\vec{a})$

$\mathcal{I}_{\mathcal{D}}$ makes the facts in $\mathcal{D}$ true (and only them)

what is the complexity of CQ answering?

naive algorithm:

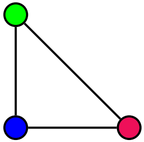
guess values for all existential variables and then
evaluate the query in polynomial time — in **NP**

can it be done better?

Why Do Databases Work? (2)

no, by reduction of the graph 3-colourability problem, which is **NP-complete**:

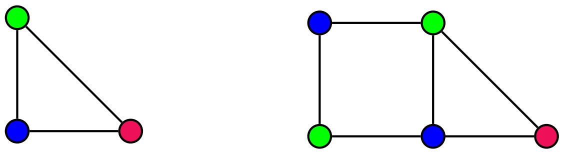
'given an undirected graph $G = (V, E)$,
decide whether it possible to colour it (using r, g, b)
so that no edge has the same colour at both ends?'



Why Do Databases Work? (2)

no, by reduction of the graph 3-colourability problem, which is **NP-complete**:

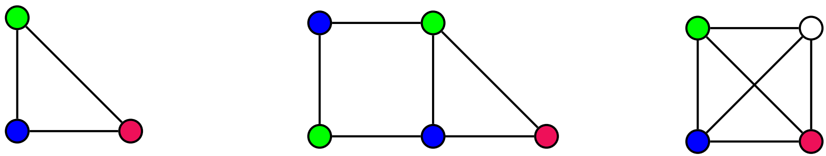
'given an undirected graph $G = (V, E)$,
decide whether it possible to colour it (using r, g, b)
so that no edge has the same colour at both ends?'



Why Do Databases Work? (2)

no, by reduction of the graph 3-colourability problem, which is **NP-complete**:

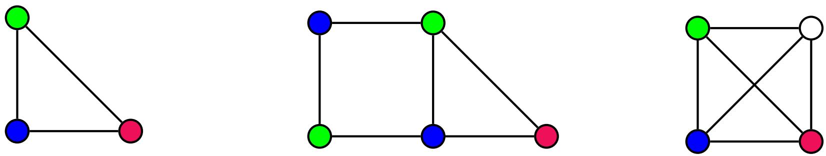
‘given an undirected graph $G = (V, E)$,
decide whether it possible to colour it (using r, g, b)
so that no edge has the same colour at both ends?’



Why Do Databases Work? (2)

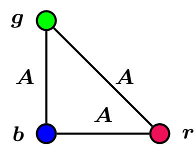
no, by reduction of the graph 3-colourability problem, which is **NP-complete**:

'given an undirected graph $G = (V, E)$,
decide whether it possible to colour it (using r, g, b)
so that no edge has the same colour at both ends?'



$$\mathcal{D} = \{A(r,g), A(g,b), A(b,r), A(g,r), A(r,b), A(b,g)\}$$

$$q_G = \exists v_1, \dots, v_n \bigwedge_{(v_i,v_j)\in E} A(v_i, v_j)$$

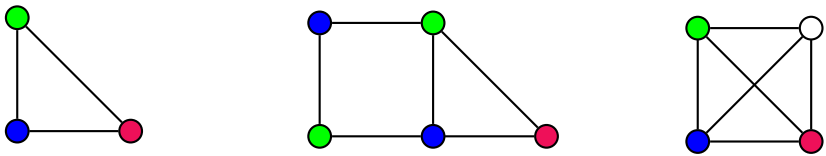


$\mathcal{D} \models q_G$ iff G is 3-colourable

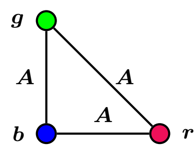
Why Do Databases Work? (2)

no, by reduction of the graph 3-colourability problem, which is **NP-complete**:

'given an undirected graph $G = (V, E)$,
decide whether it possible to colour it (using r, g, b)
so that no edge has the same colour at both ends?'



$$\mathcal{D} = \{A(r,g), A(g,b), A(b,r), A(g,r), A(r,b), A(b,g)\}$$
$$q_G = \exists v_1, \dots, v_n \bigwedge_{(v_i,v_j)\in E} A(v_i, v_j)$$



$\mathcal{D} \models q_G$ iff G is 3-colourable

in fact, the query answering algorithm runs in $O(|\mathcal{D}|^{|q|})$ **data is large, query is short**

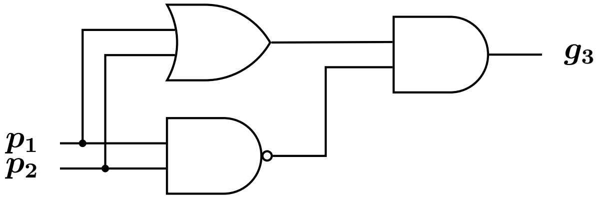
data complexity: only data $\mathcal{D}$ are counted as **input** (q is constant)

(Vardi, 1982): query answering is in AC^0 for data complexity

Circuits and AC^0

a **circuit** is an acyclic graph of AND-, OR- and NOT-gates

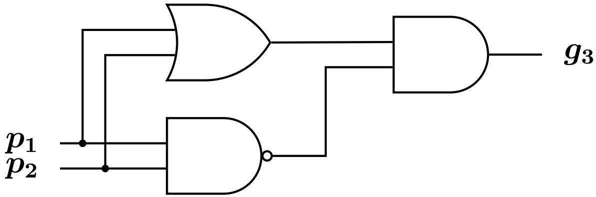
(with n inputs and a single output, **sink**)



Circuits and AC^0

a **circuit** is an acyclic graph of AND-, OR- and NOT-gates

(with n inputs and a single output, **sink**)



database instances $\mathcal{D}$ can be encoded on inputs

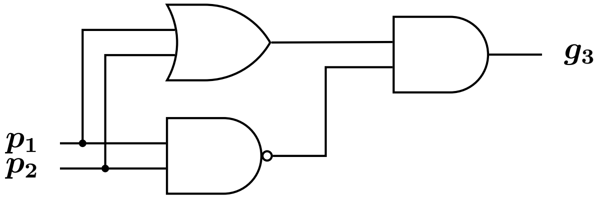
(one input for each possible ground atom)

FO-query is a circuit: $\wedge$, $\vee$ and $\neg$ are AND-, OR- and NOT-gates, respectively

Circuits and AC^0

a **circuit** is an acyclic graph of AND-, OR- and NOT-gates

(with n inputs and a single output, **sink**)



database instances $\mathcal{D}$ can be encoded on inputs

(one input for each possible ground atom)

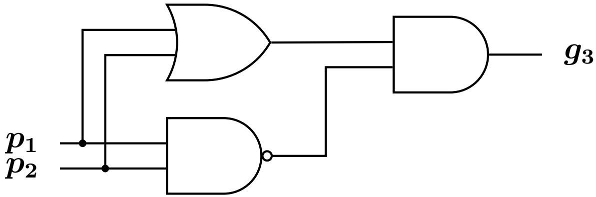
FO-query is a circuit: $\wedge$, $\vee$ and $\neg$ are AND-, OR- and NOT-gates, respectively

$\forall$ and $\exists$ are AND- and OR-gates with unbounded fan-in

Circuits and AC^0

a **circuit** is an acyclic graph of AND-, OR- and NOT-gates

(with n inputs and a single output, **sink**)



database instances $\mathcal{D}$ can be encoded on inputs

(one input for each possible ground atom)

FO-query is a circuit: $\wedge$, $\vee$ and $\neg$ are AND-, OR- and NOT-gates, respectively

$\forall$ and $\exists$ are AND- and OR-gates with unbounded fan-in

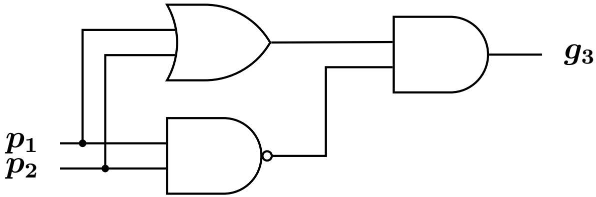
AC^0 = circuits of constant depth with AND- and OR-nodes of unbounded fan-in

constant time by a polynomial number of processors (high degree of parallelism)

Circuits and AC^0

a **circuit** is an acyclic graph of AND-, OR- and NOT-gates

(with n inputs and a single output, **sink**)



database instances $\mathcal{D}$ can be encoded on inputs

(one input for each possible ground atom)

FO-query is a circuit: $\wedge$, $\vee$ and $\neg$ are AND-, OR- and NOT-gates, respectively

$\forall$ and $\exists$ are AND- and OR-gates with unbounded fan-in

AC^0 = circuits of constant depth with AND- and OR-nodes of unbounded fan-in

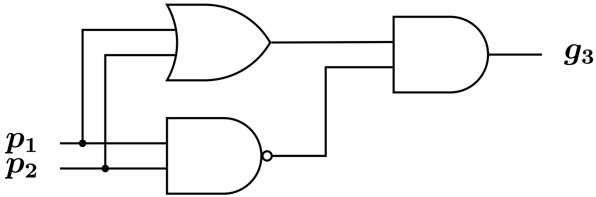
constant time by a polynomial number of processors (high degree of parallelism)

the depth of this circuit does not depend on $\mathcal{D}$ → Vardi's theorem

Circuits and AC^0

a **circuit** is an acyclic graph of AND-, OR- and NOT-gates

(with n inputs and a single output, **sink**)



database instances $\mathcal{D}$ can be encoded on inputs

(one input for each possible ground atom)

FO-query is a circuit: $\land$, $\lor$ and $\neg$ are AND-, OR- and NOT-gates, respectively

$\forall$ and $\exists$ are AND- and OR-gates with unbounded fan-in

AC^0 = circuits of constant depth with AND- and OR-nodes of unbounded fan-in

constant time by a polynomial number of processors (high degree of parallelism)

the depth of this circuit does not depend on $\mathcal{D}$ → Vardi's theorem

NB: AC^0 is a **proper** subclass of LogSpace $\subseteq$ P (Parity does not belong to AC^0)

given a word w, decide whether its length is even

Part 2

Basics of Ontology Languages

DLs and OWL: Syntax

concepts (classes, sets of elements)

$$C \quad ::= \quad \underbrace{A_i}_{\text{concept name}} \quad | \quad \underbrace{\top}_{\text{owl:Thing}} \quad | \quad \underbrace{\bot}_{\text{owl:Nothing}} \quad |$$

$$\underbrace{\neg C}_{\text{ObjectComplementOf}(C)} \quad | \quad \underbrace{C_1 \sqcap C_2}_{\text{ObjectIntersectionOf}(C_1,C_2)} \quad | \quad \underbrace{C_1 \sqcup C_2}_{\text{ObjectUnionOf}(C_1,C_2)} \quad |$$

$$\underbrace{\exists R.C}_{\text{ObjectSomeValuesFrom}(R,C)} \quad | \quad \underbrace{\forall R.C}_{\text{ObjectAllValuesFrom}(R,C)}$$

DLs and OWL: Syntax

concepts (classes, sets of elements)

$$C \quad ::= \quad \underbrace{A_i}_{\text{concept name}} \quad | \quad \underbrace{\top}_{\text{owl:Thing}} \quad | \quad \underbrace{\bot}_{\text{owl:Nothing}} \quad |$$

$$\underbrace{\neg C}_{\text{ObjectComplementOf}(C)} \quad | \quad \underbrace{C_1 \sqcap C_2}_{\text{ObjectIntersectionOf}(C_1,C_2)} \quad | \quad \underbrace{C_1 \sqcup C_2}_{\text{ObjectUnionOf}(C_1,C_2)} \quad |$$

$$\underbrace{\exists R.C}_{\text{ObjectSomeValuesFrom}(R,C)} \quad | \quad \underbrace{\forall R.C}_{\text{ObjectAllValuesFrom}(R,C)}$$

roles (object properties, binary relations)

$$R \quad ::= \quad \underbrace{P_i}_{\text{role name}} \quad | \quad P_i^-$$

DLs and OWL: Syntax

concepts (classes, sets of elements)

$$C \quad ::= \quad \underbrace{A_i}_{\text{concept name}} \mid \underbrace{\top}_{\text{owl:Thing}} \mid \underbrace{\bot}_{\text{owl:Nothing}} \mid$$

$$\underbrace{\neg C}_{\text{ObjectComplementOf}(C)} \mid \underbrace{C_1 \sqcap C_2}_{\text{ObjectIntersectionOf}(C_1,C_2)} \mid \underbrace{C_1 \sqcup C_2}_{\text{ObjectUnionOf}(C_1,C_2)} \mid$$

$$\underbrace{\exists R.C}_{\text{ObjectSomeValuesFrom}(R,C)} \mid \underbrace{\forall R.C}_{\text{ObjectAllValuesFrom}(R,C)}$$

roles (object properties, binary relations)

$$R \quad ::= \quad \underbrace{P_i}_{\text{role name}} \mid P_i^-$$

TBox $\mathcal{T}$

$$\underbrace{C_1 \sqsubseteq C_2}_{\text{SubClassOf}(C_1,C_2)} \quad \text{and} \quad \underbrace{R_1 \sqsubseteq R_2}_{\text{SubObjectPropertyOf}(R_1,R_2)}$$

ABox $\mathcal{A}$

$$C(a) \quad \text{and} \quad R(a, b)$$

DLs and OWL: Syntax

concepts (classes, sets of elements)

$$C \;::=\; \underbrace{A_i}_{\text{concept name}} \mid \underbrace{\top}_{\text{owl:Thing}} \mid \underbrace{\bot}_{\text{owl:Nothing}} \mid$$

$$\underbrace{\neg C}_{\text{ObjectComplementOf}(C)} \mid \underbrace{C_1 \sqcap C_2}_{\text{ObjectIntersectionOf}(C_1,C_2)} \mid \underbrace{C_1 \sqcup C_2}_{\text{ObjectUnionOf}(C_1,C_2)} \mid$$

$$\underbrace{\exists R.C}_{\text{ObjectSomeValuesFrom}(R,C)} \mid \underbrace{\forall R.C}_{\text{ObjectAllValuesFrom}(R,C)}$$

roles (object properties, binary relations)

$$R \;::=\; \underbrace{P_i}_{\text{role name}} \mid P_i^-$$

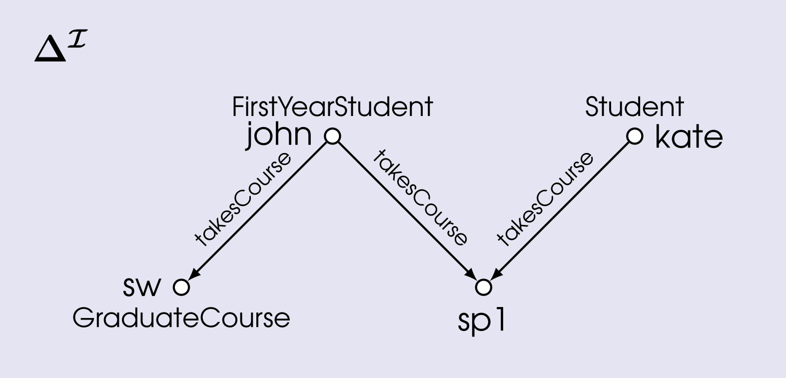
TBox $\mathcal{T}$ $\quad \underbrace{C_1 \sqsubseteq C_2}_{\text{SubClassOf}(C_1,C_2)}$ and $\underbrace{R_1 \sqsubseteq R_2}_{\text{SubObjectPropertyOf}(R_1,R_2)}$

ABox $\mathcal{A}$ $\quad C(a)$ and $R(a, b)$

knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (ontology)

DL Semantics

interpretation $\mathcal{I} = (\underbrace{\Delta^{\mathcal{I}}}_{\text{domain}}, \cdot^{\mathcal{I}})$

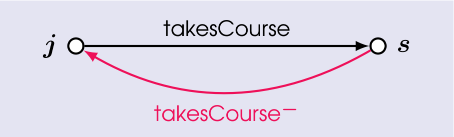


$\cdot^{\mathcal{I}}$ (interpretation function)

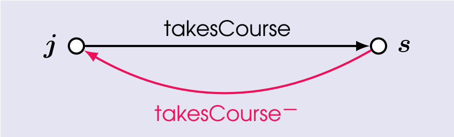
individuals a_i	$\rightarrow$	elements $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
concept names A_i	$\rightarrow$	subsets $A_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role names P_i	$\rightarrow$	binary relations $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

DL Semantics (2)

$$(P^{-})^{\mathcal{I}} \ = \ \{(v, u) \mid (u, v) \in P^{\mathcal{I}}\}$$



DL Semantics (2)

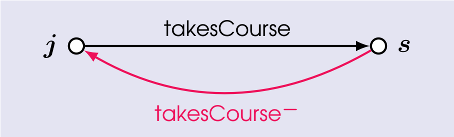


$$(P^-)^{\mathcal{I}} = \{(v,u) \mid (u,v) \in P^{\mathcal{I}}\}$$

$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$

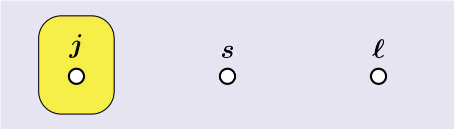
DL Semantics (2)

$$(P^-)^\mathcal{I} \;=\; \{(v,u) \mid (u,v) \in P^\mathcal{I}\}$$



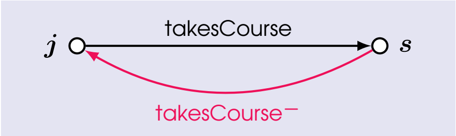
$\top^\mathcal{I} \;=\; \Delta^\mathcal{I}$ and $\bot^\mathcal{I} \;=\; \emptyset$

$$(\neg C)^\mathcal{I} \;=\; \Delta^\mathcal{I} \setminus C^\mathcal{I}$$



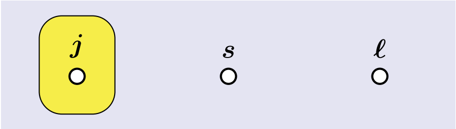
DL Semantics (2)

$$(P^-)^\mathcal{I} = \{(v,u) \mid (u,v) \in P^\mathcal{I}\}$$



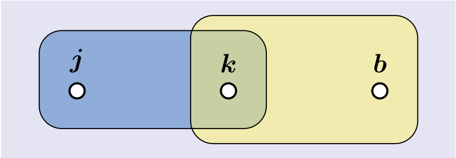
$$\top^\mathcal{I} = \Delta^\mathcal{I} \quad \text{and} \quad \bot^\mathcal{I} = \emptyset$$

$$(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$$



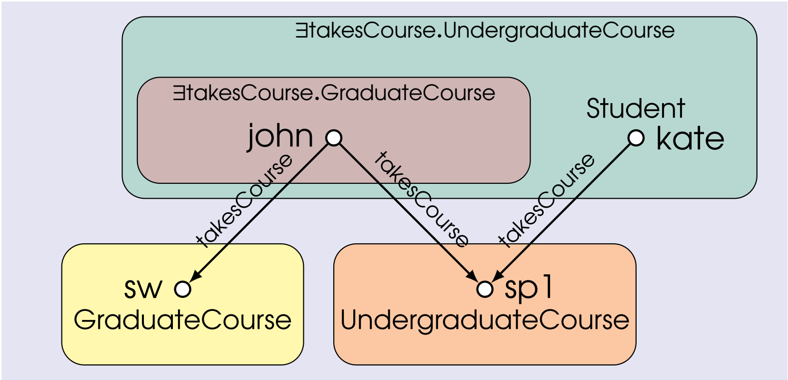
$$(C_1 \sqcap C_2)^\mathcal{I} = C_1^\mathcal{I} \cap C_2^\mathcal{I}$$

$$(C_1 \sqcup C_2)^\mathcal{I} = C_1^\mathcal{I} \cup C_2^\mathcal{I}$$



DL Semantics (3)

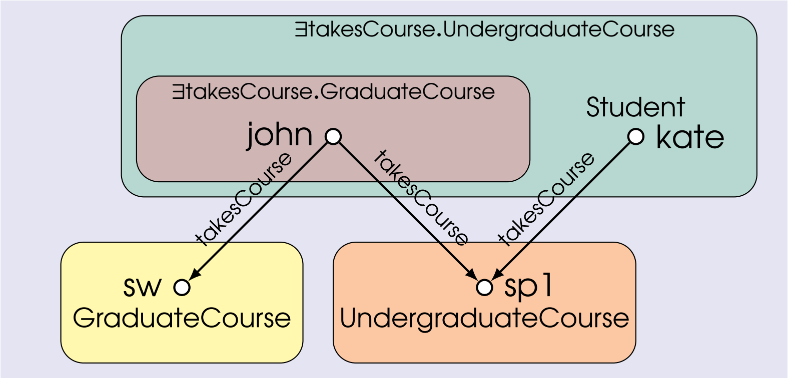
$$(\exists R.C)^{\mathcal{I}} = \{ u \mid \text{ there is } v \in C^{\mathcal{I}} \text{ such that } (u, v) \in R^{\mathcal{I}} \}$$



$\Diamond_R C$ or $\exists y\,(R(x,y) \wedge C(y))$

DL Semantics (3)

$$(\exists R.C)^{\mathcal{I}} = \{\, u \mid \text{ there is } v \in C^{\mathcal{I}} \text{ such that } (u, v) \in R^{\mathcal{I}} \,\}$$



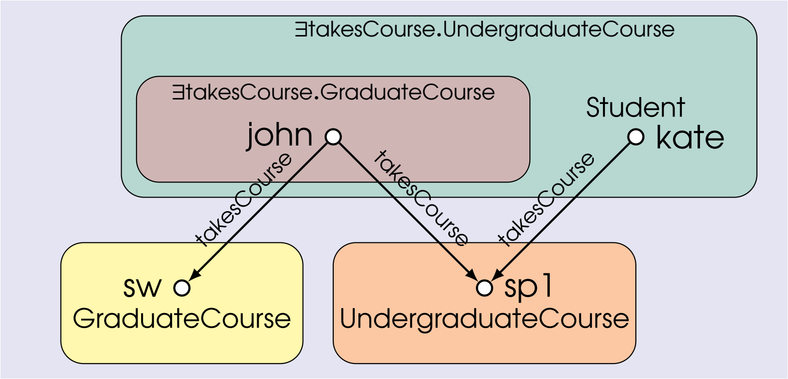
$\Diamond_R C$ or $\exists y\,(R(x, y) \land C(y))$

$$(\forall R.C)^{\mathcal{I}} = \{\, u \mid v \in C^{\mathcal{I}}, \text{ for all } v \text{ with } (u, v) \in R^{\mathcal{I}} \,\}$$

$\forall R.C = \neg\exists R.\neg C$

DL Semantics (3)

$$(\exists R.C)^{\mathcal{I}} = \{\, u \mid \text{ there is } v \in C^{\mathcal{I}} \text{ such that } (u,v) \in R^{\mathcal{I}} \,\}$$



$\Diamond_R C$ or $\exists y\,(R(x,y) \land C(y))$

$$(\forall R.C)^{\mathcal{I}} = \{\, u \mid v \in C^{\mathcal{I}}, \text{ for all } v \text{ with } (u,v) \in R^{\mathcal{I}} \,\}$$

$\forall R.C = \neg\exists R.\neg C$

NB. "for all" is true when there are no v with $(u,v) \in R^{\mathcal{I}}$

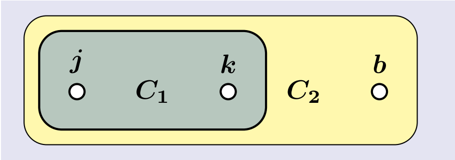
e.g., sp1 $\in$ ($\forall$takesCourse.UndergraduateCourse)$^{\mathcal{I}}$

sp1 $\in$ ($\forall$takesCourse.$\bot$)$^{\mathcal{I}}$

$\Box_R C$ or $\forall y\,(R(x,y) \to C(y))$

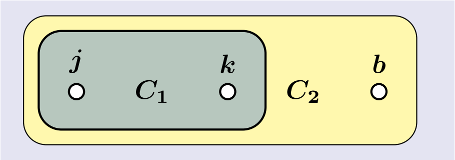
DL Semantics (4)

$$\mathcal{I} \models C_1 \sqsubseteq C_2 \iff C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$$



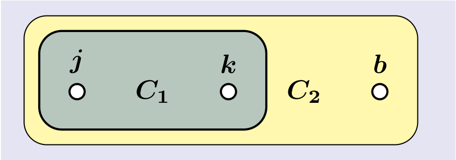
DL Semantics (4)

$$\mathcal{I} \models C_1 \sqsubseteq C_2 \iff C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$$



$$\mathcal{I} \models R_1 \sqsubseteq R_2 \iff R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$$

DL Semantics (4)



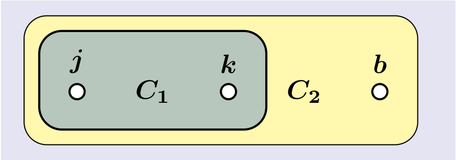
$$\mathcal{I} \models C_1 \sqsubseteq C_2 \iff C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$$

$$\mathcal{I} \models R_1 \sqsubseteq R_2 \iff R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$$

$$\mathcal{I} \models C(a) \iff a^{\mathcal{I}} \in C^{\mathcal{I}}$$

$$\mathcal{I} \models R(a, b) \iff (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$$

DL Semantics (4)



$$\mathcal{I} \models C_1 \sqsubseteq C_2 \iff C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$$

$$\mathcal{I} \models R_1 \sqsubseteq R_2 \iff R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$$

$$\mathcal{I} \models C(a) \iff a^{\mathcal{I}} \in C^{\mathcal{I}}$$

$$\mathcal{I} \models R(a, b) \iff (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$$

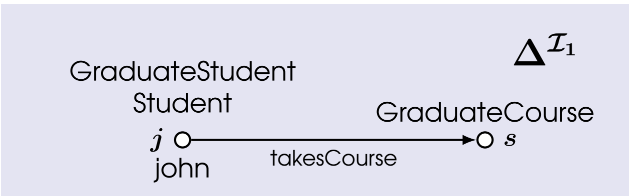
$\mathcal{I}$ is a **model** of $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$, for all inclusions α in $\mathcal{T}$ and assertions α in $\mathcal{A}$

Open World Assumption

$$\mathcal{T} = \{ \text{GraduateStudent} \sqsubseteq \text{Student}$$
$$\text{GraduateStudent} \sqsubseteq \exists \text{takesCourse}.\text{GraduateCourse} \}$$
$$\mathcal{A} = \{ \text{GraduateStudent(john)} \}$$

Open World Assumption

$$\mathcal{T} = \{ \text{GraduateStudent} \sqsubseteq \text{Student}$$
$$\text{GraduateStudent} \sqsubseteq \exists \text{takesCourse}.\text{GraduateCourse} \}$$
$$\mathcal{A} = \{ \text{GraduateStudent(john)} \}$$

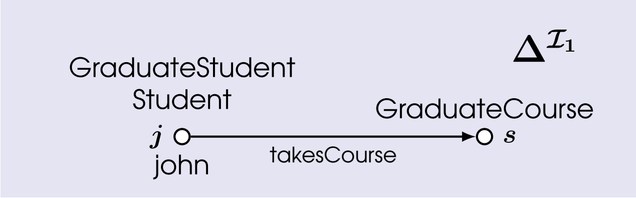


$\text{john}^{\mathcal{I}_1} = j$
$\text{GraduateStudent}^{\mathcal{I}_1} = \{j\}$
$\text{Student}^{\mathcal{I}_1} = \{j\}$
$\text{GraduateCourse}^{\mathcal{I}_1} = \{s\}$
$\text{takesCourse}^{\mathcal{I}_1} = \{(j, s)\}$

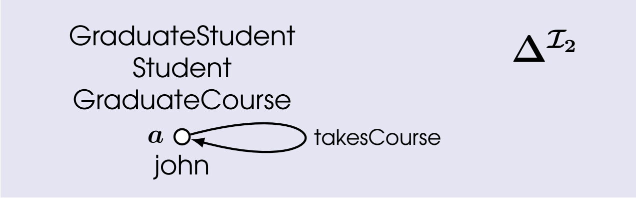
is a **model** of $(\mathcal{T}, \mathcal{A})$

Open World Assumption

$$\mathcal{T} = \{ \text{GraduateStudent} \sqsubseteq \text{Student}$$
$$\text{GraduateStudent} \sqsubseteq \exists \text{takesCourse}.\text{GraduateCourse} \}$$
$$\mathcal{A} = \{ \text{GraduateStudent(john)} \}$$



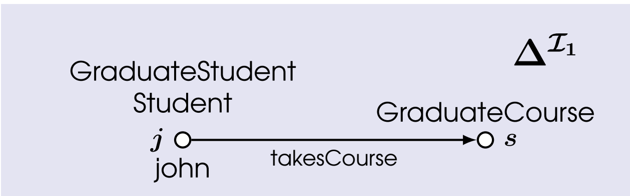
$\text{john}^{\mathcal{I}_1} = j$
$\text{GraduateStudent}^{\mathcal{I}_1} = \{j\}$
$\text{Student}^{\mathcal{I}_1} = \{j\}$
$\text{GraduateCourse}^{\mathcal{I}_1} = \{s\}$
$\text{takesCourse}^{\mathcal{I}_1} = \{(j, s)\}$
is a **model** of $(\mathcal{T}, A)$



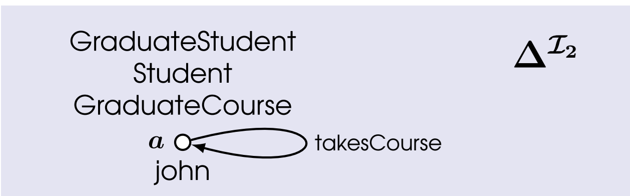
$\text{john}^{\mathcal{I}_2} = a$
$\text{GraduateStudent}^{\mathcal{I}_2} = \{a\}$
$\text{Student}^{\mathcal{I}_2} = \{a\}$
$\text{GraduateCourse}^{\mathcal{I}_2} = \{a\}$
$\text{takesCourse}^{\mathcal{I}_2} = \{(a, a)\}$
is a **model** of $(\mathcal{T}, A)$

Open World Assumption

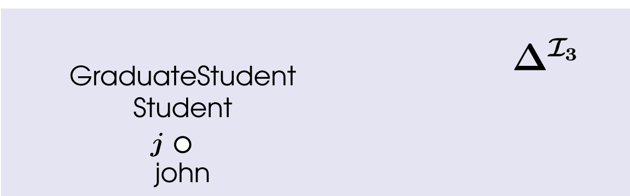
$$\mathcal{T} = \{ \text{GraduateStudent} \sqsubseteq \text{Student}$$
$$\text{GraduateStudent} \sqsubseteq \exists \text{takesCourse.GraduateCourse} \}$$
$$\mathcal{A} = \{ \text{GraduateStudent(john)} \}$$



$\text{john}^{\mathcal{I}_1} = j$
$\text{GraduateStudent}^{\mathcal{I}_1} = \{j\}$
$\text{Student}^{\mathcal{I}_1} = \{j\}$
$\text{GraduateCourse}^{\mathcal{I}_1} = \{s\}$
$\text{takesCourse}^{\mathcal{I}_1} = \{(j, s)\}$
is a **model** of $(\mathcal{T}, \mathcal{A})$



$\text{john}^{\mathcal{I}_2} = a$
$\text{GraduateStudent}^{\mathcal{I}_2} = \{a\}$
$\text{Student}^{\mathcal{I}_2} = \{a\}$
$\text{GraduateCourse}^{\mathcal{I}_2} = \{a\}$
$\text{takesCourse}^{\mathcal{I}_2} = \{(a, a)\}$
is a **model** of $(\mathcal{T}, \mathcal{A})$



$\text{john}^{\mathcal{I}_3} = j$
$\text{GraduateStudent}^{\mathcal{I}_3} = \{j\}$
$\text{Student}^{\mathcal{I}_3} = \{j\}$
$\text{GraduateCourse}^{\mathcal{I}_3} = \emptyset$
$\text{takesCourse}^{\mathcal{I}_3} = \emptyset$
is **not a model** of $(\mathcal{T}, \mathcal{A})$

Reasoning: Consistency

a knowledge base $\mathcal{K}$ is **satisfiable** (or **consistent**)

if there exists at least one model of $\mathcal{K}$

(in other words, $\mathcal{K}$ implies **no contradictions**)

Reasoning: Consistency

a knowledge base $\mathcal{K}$ is **satisfiable** (or **consistent**)

if there exists at least one model of $\mathcal{K}$

(in other words, $\mathcal{K}$ implies **no contradictions**)

Example

$\mathcal{T}$:

$\text{UndergraduateStudent} \sqsubseteq \forall \text{takesCourse.UndergraduateCourse}$

$\text{UndergraduateCourse} \sqcap \text{GraduateCourse} \sqsubseteq \bot$

$\mathcal{A}$:

UndergraduateStudent(john)

takesCourse(john, sw)

GraduateCourse(sw)

Reasoning: Consistency

a knowledge base $\mathcal{K}$ is **satisfiable** (or **consistent**)

if there exists at least one model of $\mathcal{K}$

(in other words, $\mathcal{K}$ implies **no contradictions**)

Example

$\mathcal{T}$:

$$\mathsf{UndergraduateStudent} \sqsubseteq \forall \mathsf{takesCourse}.\mathsf{UndergraduateCourse}$$

$$\mathsf{UndergraduateCourse} \sqcap \mathsf{GraduateCourse} \sqsubseteq \bot$$

$\mathcal{A}$:

$$\mathsf{UndergraduateStudent}(\mathsf{john})$$

$$\mathsf{takesCourse}(\mathsf{john}, \mathsf{sw})$$

$$\mathsf{GraduateCourse}(\mathsf{sw})$$

$(\mathcal{T}, \mathcal{A})$ is **inconsistent**:

John (as an undergraduate student) can take only undergraduate courses.
We know, however, that he takes a graduate course,
which cannot be an undergraduate one.

Reasoning: Entailment

$C_1 \sqsubseteq C_2$ is **entailed by** $\mathcal{K}$ $\qquad$ $\mathcal{K} \models C_1 \sqsubseteq C_2$

if $\mathcal{I} \models C_1 \sqsubseteq C_2$ for all models $\mathcal{I}$ of $\mathcal{K}$

(entailment for role inclusions and concept and role assertions is defined similarly)

Reasoning: Entailment

$C_1 \sqsubseteq C_2$ is **entailed by** $\mathcal{K}$ $\quad\quad$ $\mathcal{K} \models C_1 \sqsubseteq C_2$

if $\mathcal{I} \models C_1 \sqsubseteq C_2$ for all models $\mathcal{I}$ of $\mathcal{K}$

(entailment for role inclusions and concept and role assertions is defined similarly)

$\mathcal{T}$: $\forall$takesCourse.UndergraduateCourse $\sqsubseteq$ UndergraduateStudent

FirstYearStudent $\sqsubseteq$ $\exists$takesCourse.UndergraduateCourse.

Reasoning: Entailment

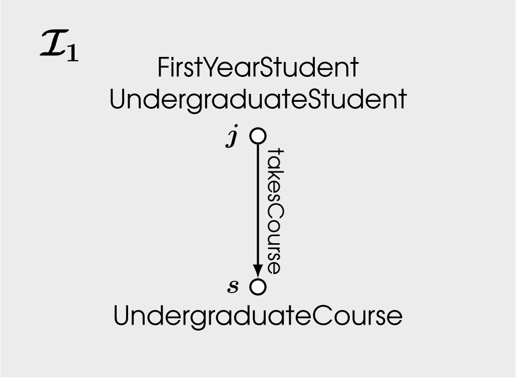
$C_1 \sqsubseteq C_2$ is **entailed by** $\mathcal{K}$ $\qquad$ $\mathcal{K} \models C_1 \sqsubseteq C_2$

if $\mathcal{I} \models C_1 \sqsubseteq C_2$ for all models $\mathcal{I}$ of $\mathcal{K}$

(entailment for role inclusions and concept and role assertions is defined similarly)

$\mathcal{T}$: $\forall$takesCourse.UndergraduateCourse $\sqsubseteq$ UndergraduateStudent

FirstYearStudent $\sqsubseteq$ $\exists$takesCourse.UndergraduateCourse.



$\mathcal{I}_1 \models \mathcal{T}$

$\mathcal{I}_1 \models$ FirstYearStudent $\sqsubseteq$ UndergraduateStudent

Reasoning: Entailment

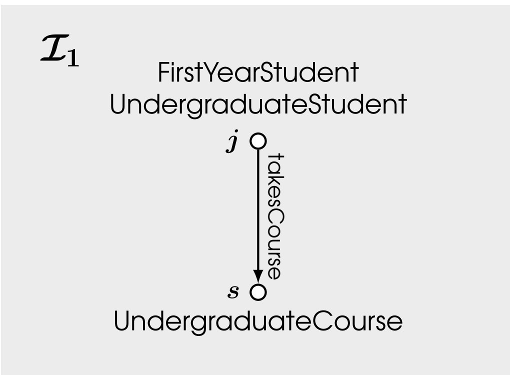
$C_1 \sqsubseteq C_2$ is **entailed by** $\mathcal{K}$ $\qquad \mathcal{K} \models C_1 \sqsubseteq C_2$

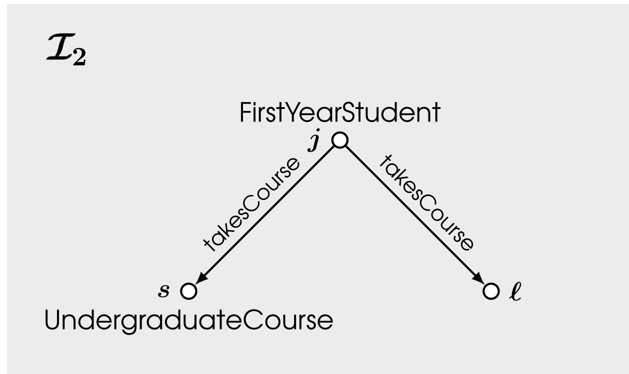
if $\mathcal{I} \models C_1 \sqsubseteq C_2$ for all models $\mathcal{I}$ of $\mathcal{K}$

(entailment for role inclusions and concept and role assertions is defined similarly)

$\mathcal{T}$: $\quad \forall$takesCourse.UndergraduateCourse $\sqsubseteq$ UndergraduateStudent

FirstYearStudent $\sqsubseteq$ $\exists$takesCourse.UndergraduateCourse.





$\mathcal{I}_1 \models \mathcal{T}$

$\mathcal{I}_1 \models$ FirstYearStudent $\sqsubseteq$ UndergraduateStudent

$\mathcal{I}_2 \models \mathcal{T}$

$\mathcal{I}_2 \models$ FirstYearStudent $\not\sqsubseteq$ UndergraduateStudent

Certain Answers to CQs

$q(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ is a CQ with $\vec{x} = (x_1, \dots, x_n)$

$\vec{a} = (a_1, \dots, a_n)$ is a tuple of individual names from $\mathcal{A}$

$q(\vec{a})$ is the result of replacing each x_i in $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ with a_i

Certain Answers to CQs

$q(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ is a CQ with $\vec{x} = (x_1, \dots, x_n)$

$\vec{a} = (a_1, \dots, a_n)$ is a tuple of individual names from $\mathcal{A}$

$q(\vec{a})$ is the result of replacing each x_i in $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ with a_i

$\vec{a}$ is a **certain answer** to $q(\vec{x})$ over $\mathcal{T}, \mathcal{A}$ $\quad$ $(\mathcal{T}, \mathcal{A}) \models q(\vec{a})$

if, <u>for any model</u> $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, the sentence $q(\vec{a})$ is true in $\mathcal{I}$

$\mathcal{I} \models q(\vec{a})$

Andrea's Example (Schaerf, 1993)

$\mathcal{T}$: $\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}$, $\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot$

$\mathcal{A}$: friend(john, susan), friend(john, andrea), Female(susan)
loves(susan, andrea), loves(andrea, bill), Male(bill)

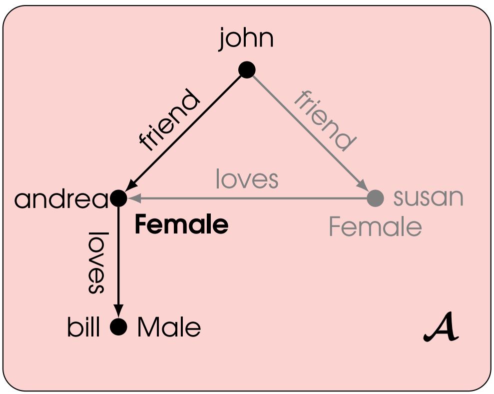
$$q = \exists y, z \left(\mathsf{friend}(\mathsf{john}, y) \wedge \mathsf{Female}(y) \wedge \mathsf{loves}(y, z) \wedge \mathsf{Male}(z)\right)$$

Andrea's Example (Schaerf, 1993)

$\mathcal{T}$: $\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}$, $\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot$

$\mathcal{A}$: friend(john, susan), friend(john, andrea), Female(susan)
loves(susan, andrea), loves(andrea, bill), Male(bill)

$$q = \exists y, z\ (\mathsf{friend}(\mathsf{john}, y) \land \mathsf{Female}(y) \land \mathsf{loves}(y, z) \land \mathsf{Male}(z))$$

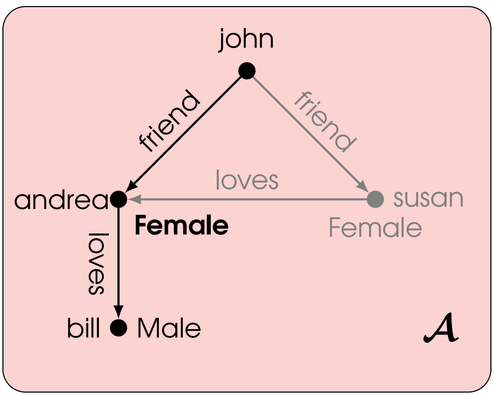


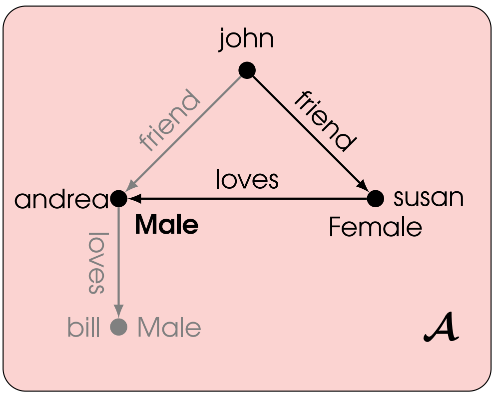
Andrea's Example (Schaerf, 1993)

$\mathcal{T}$: $\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}$, $\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot$

$\mathcal{A}$: friend(john, susan), friend(john, andrea), Female(susan)
loves(susan, andrea), loves(andrea, bill), Male(bill)

$q = \exists y, z\ (\mathsf{friend}(\mathsf{john}, y) \wedge \mathsf{Female}(y) \wedge \mathsf{loves}(y, z) \wedge \mathsf{Male}(z))$



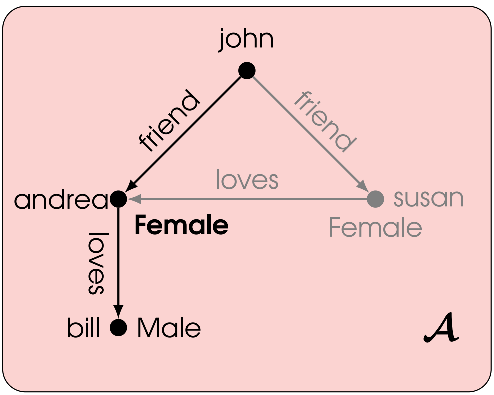


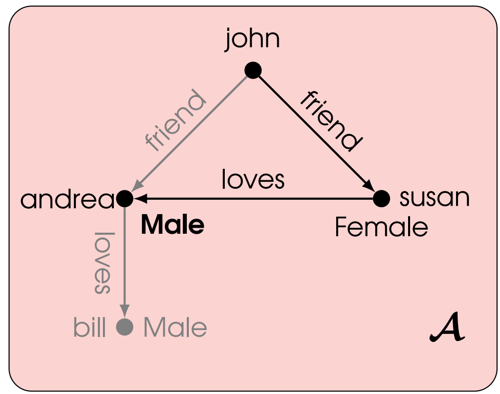
Andrea's Example (Schaerf, 1993)

$\mathcal{T}$: $\top \sqsubseteq \text{Male} \sqcup \text{Female}, \quad \text{Male} \sqcap \text{Female} \sqsubseteq \bot$

$\mathcal{A}$: friend(john, susan), friend(john, andrea), Female(susan)
loves(susan, andrea), loves(andrea, bill), Male(bill)

$q = \exists y, z \; (\text{friend}(\text{john}, y) \wedge \text{Female}(y) \wedge \text{loves}(y, z) \wedge \text{Male}(z))$





NB: the same as checking whether john is an instance of $\exists$friend.(Female $\sqcap$ $\exists$loves.Male)

DL Zoo

$\mathcal{ALCHI}$

$\mathcal{AL}$ – attributive language

$\mathcal{C}$ – complement $\neg C$ ($\mathcal{ALC}$ is multi-modal $\mathbf{K}_m$)

$\mathcal{I}$ – role inverses P^-

$\mathcal{H}$ – role inclusions $R_1 \sqsubseteq R_2$

DL Zoo

$\mathcal{ALCHI}$

$\mathcal{AL}$ – attributive language

- $\mathcal{C}$ – complement $\neg C$ ($\mathcal{ALC}$ is multi-modal $\mathbf{K}_m$)
- $\mathcal{I}$ – role inverses P^-
- $\mathcal{H}$ – role inclusions $R_1 \sqsubseteq R_2$

$\mathcal{S}$ – $\mathcal{ALC}$ + transitive roles

- $\mathcal{N}$ – unqualified number restrictions $\geq q\, R.\top$
- $\mathcal{O}$ – nominals $\{a\}$

$\mathcal{SHOIN} \approx$ OWL 1.0

- $\mathcal{Q}$ – qualified number restrictions $\geq q\, R.C$

DL Zoo

$\mathcal{ALCHI}$

$\mathcal{AL}$ – attributive language

$\mathcal{C}$ – complement $\neg C$ ($\mathcal{ALC}$ is multi-modal $\mathbf{K}_m$)

$\mathcal{I}$ – role inverses P^-

$\mathcal{H}$ – role inclusions $R_1 \sqsubseteq R_2$

$\mathcal{S}$ – $\mathcal{ALC}$ + transitive roles

$\mathcal{N}$ – unqualified number restrictions $\geq q\, R.\top$

$\mathcal{O}$ – nominals $\{a\}$

$\mathcal{SHOIN} \approx$ OWL 1.0

$\mathcal{Q}$ – qualified number restrictions $\geq q\, R.C$

$\mathcal{F}$ – functionality constraints $\geq 2\, R.\top \sqsubseteq \bot$

$\mathcal{SHIF} \approx$ OWL Lite

DL Zoo

$\mathcal{ALCHI}$

$\mathcal{AL}$ – attributive language

$\mathcal{C}$ – complement $\neg C$ ($\mathcal{ALC}$ is multi-modal $\mathbf{K}_m$)

$\mathcal{I}$ – role inverses P^-

$\mathcal{H}$ – role inclusions $R_1 \sqsubseteq R_2$

$\mathcal{S}$ – $\mathcal{ALC}$ + transitive roles

$\mathcal{N}$ – unqualified number restrictions $\geq q\, R.\top$

$\mathcal{O}$ – nominals $\{a\}$

$\mathcal{SHOIN} \approx$ OWL 1.0

$\mathcal{Q}$ – qualified number restrictions $\geq q\, R.C$

$\mathcal{F}$ – functionality constraints $\geq 2\, R.\top \sqsubseteq \bot$

$\mathcal{SHIF} \approx$ OWL Lite

$\mathcal{R}$ – role chains and $\exists R.$Self

$\mathcal{SROIQ} \approx$ OWL 2

Complexity of Reasoning

The satisfiability problem is **ExpTime**-complete for $\mathcal{ALCHI}$ KBs
and **N2ExpTime**-complete for $\mathcal{SROIQ}$ KBs

Complexity of Reasoning

The satisfiability problem is **ExpTime**-complete for $\mathcal{ALCHI}$ KBs
and **N2ExpTime**-complete for $\mathcal{SROIQ}$ KBs

Concept and role subsumption and instance checking are **ExpTime**- and **coN2ExpTime**-complete for, respectively, $\mathcal{ALCHI}$ and $\mathcal{SROIQ}$ KBs

Complexity of Reasoning

The satisfiability problem is **ExpTime**-complete for $\mathcal{ALCHI}$ KBs
and **N2ExpTime**-complete for $\mathcal{SROIQ}$ KBs

Concept and role subsumption and instance checking are **ExpTime**- and **coN2ExpTime**-complete for, respectively, $\mathcal{ALCHI}$ and $\mathcal{SROIQ}$ KBs

CQ entailment over $\mathcal{ALCHI}$ KBs is **2ExpTime**-complete

CQ entailment over $\mathcal{SROIQ}$ is not even known to be decidable

Complexity of Reasoning

The satisfiability problem is **ExpTime**-complete for $\mathcal{ALCHI}$ KBs
and **N2ExpTime**-complete for $\mathcal{SROIQ}$ KBs

Concept and role subsumption and instance checking are **ExpTime**- and
coN2ExpTime-complete for, respectively, $\mathcal{ALCHI}$ and $\mathcal{SROIQ}$ KBs

CQ entailment over $\mathcal{ALCHI}$ KBs is **2ExpTime**-complete

CQ entailment over $\mathcal{SROIQ}$ is not even known to be decidable

DL Complexity Navigator: `www.cs.man.ac.uk/~ezolin/dl`

Complexity of Reasoning

The satisfiability problem is **ExpTime**-complete for $\mathcal{ALCHI}$ KBs
and **N2ExpTime**-complete for $\mathcal{SROIQ}$ KBs

Concept and role subsumption and instance checking are **ExpTime**- and
coN2ExpTime-complete for, respectively, $\mathcal{ALCHI}$ and $\mathcal{SROIQ}$ KBs

CQ entailment over $\mathcal{ALCHI}$ KBs is **2ExpTime**-complete

CQ entailment over $\mathcal{SROIQ}$ is not even known to be decidable

DL Complexity Navigator: `www.cs.man.ac.uk/~ezolin/dl`

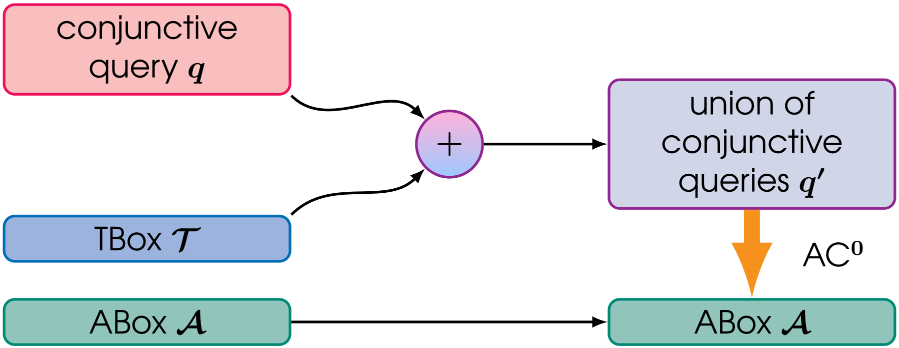
practical reasoners for OWL 2 DL: FaCT++, HermiT, Pellet

Part 3

Conjunctive Query Rewriting

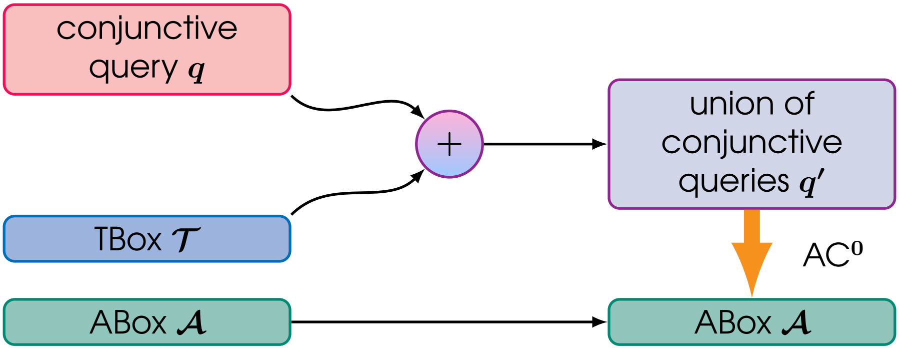
Query Rewriting Approach

(Calvanese et al. 2008): use off-the-shelf RDBMS



Query Rewriting Approach

(Calvanese et al. 2008): use off-the-shelf RDBMS

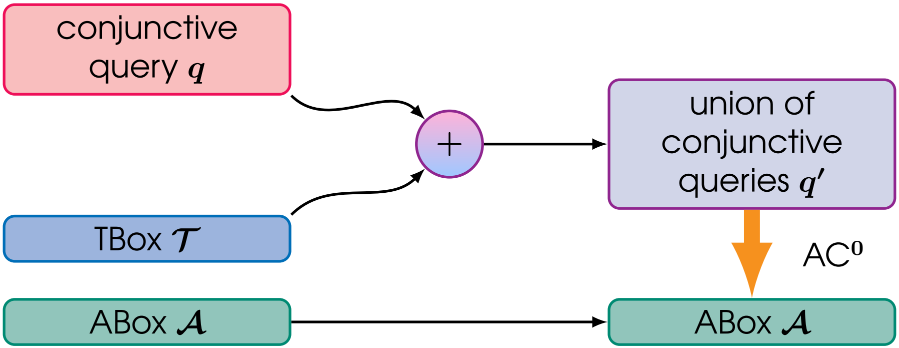


given a CQ $q(\vec{x})$ over $\mathcal{T}$, rewrite $q(\vec{x})$ into an FO query $q'(\vec{x})$ such that

for all $\mathcal{A}$ and $\vec{a}$, $\quad \mathcal{T}, \mathcal{A} \models q(\vec{a}) \quad$ iff $\quad \mathcal{A} \models q'(\vec{a})$

Query Rewriting Approach

(Calvanese et al. 2008): use off-the-shelf RDBMS



given a CQ $q(\vec{x})$ over $\mathcal{T}$, rewrite $q(\vec{x})$ into an FO query $q'(\vec{x})$ such that

for all $\mathcal{A}$ and $\vec{a}$, $\quad \mathcal{T}, \mathcal{A} \models q(\vec{a}) \quad$ iff $\quad \mathcal{A} \models q'(\vec{a})$

FO-rewritability: only possible in DL with query answering in **FO** (=AC^0) for data complexity:

OWL 2 QL

W3C Standard OWL 2 QL

OWL 2 QL is a profile of **OWL 2** designed with the aim of OBDA

(and based on the *DL-Lite* family of DLs)

roles	R	$::=$	P_i	$\mid$	P_i^-		
basic concepts	B	$::=$	$\bot$	$\mid$	A_i	$\mid$	$\exists R$
concepts	C	$::=$	B	$\mid$	$\exists R.B$		

($\exists R$ is an abbreviation for $\exists R.\top$)

a TBox $\mathcal{T}$ is a finite set of axioms of the form

$$B \sqsubseteq C, \qquad R_1 \sqsubseteq R_2, \qquad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqcap R_2 \sqsubseteq \bot$$

(plus reflexivity/irreflexivity assertions for roles)

an ABox $\mathcal{A}$ s a finite set of atoms the form $A_k(a_i)$ and $P_k(a_i, a_j)$

(plus *inequality constraints* $a_i \neq a_j$ for $i \neq j$)

W3C Standard OWL 2 QL

OWL 2 QL is a profile of **OWL 2** designed with the aim of OBDA

(and based on the *DL-Lite* family of DLs)

roles	R	$::=$	P_i	$\mid$	P_i^-		
basic concepts	B	$::=$	$\bot$	$\mid$	A_i	$\mid$	$\exists R$
concepts	C	$::=$	B	$\mid$	$\exists R.B$		

($\exists R$ is an abbreviation for $\exists R.\top$)

a TBox $\mathcal{T}$ is a finite set of axioms of the form

$$B \sqsubseteq C, \qquad R_1 \sqsubseteq R_2, \qquad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqcap R_2 \sqsubseteq \bot$$

(plus reflexivity/irreflexivity assertions for roles)

an ABox $\mathcal{A}$ s a finite set of atoms the form $A_k(a_i)$ and $P_k(a_i, a_j)$

(plus *inequality constraints* $a_i \neq a_j$ for $i \neq j$)

NB. axioms $B' \sqsubseteq \exists R.B$ are `syntactic sugar'

W3C Standard OWL 2 QL

OWL 2 QL is a profile of **OWL 2** designed with the aim of OBDA

(and based on the *DL-Lite* family of DLs)

roles	R	$::=$	P_i	$\mid$	P_i^-		
basic concepts	B	$::=$	$\bot$	$\mid$	A_i	$\mid$	$\exists R$
concepts	C	$::=$	B	$\mid$	~~$\exists R.B$~~		

($\exists R$ is an abbreviation for $\exists R.\top$)

a TBox $\mathcal{T}$ is a finite set of axioms of the form

$$B \sqsubseteq C, \qquad R_1 \sqsubseteq R_2, \qquad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqcap R_2 \sqsubseteq \bot$$

(plus reflexivity/irreflexivity assertions for roles)

an ABox $\mathcal{A}$ s a finite set of atoms the form $A_k(a_i)$ and $P_k(a_i, a_j)$

(plus *inequality constraints* $a_i \neq a_j$ for $i \neq j$)

NB. axioms $B' \sqsubseteq \exists R.B$ are 'syntactic sugar' $\quad B' \sqsubseteq \exists R_{R.B},\ \exists R_B^- \sqsubseteq B,\ R_B \sqsubseteq R$

Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

'given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G'

ABox: $\mathcal{A}_{G,t} = \{ \text{edge}(v_1, v_2) \mid (v_1, v_2) \in E \} \cup \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} = \{ \exists\text{edge.ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget} \}$

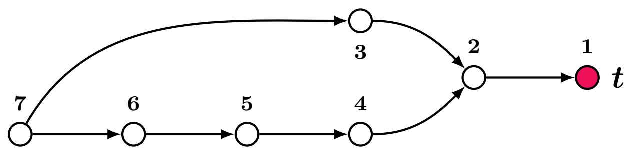
Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

'given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G'

ABox: $\mathcal{A}_{G,t} = \{ \text{edge}(v_1, v_2) \mid (v_1, v_2) \in E \} \cup \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} = \{ \exists \text{edge}.\text{ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget} \}$



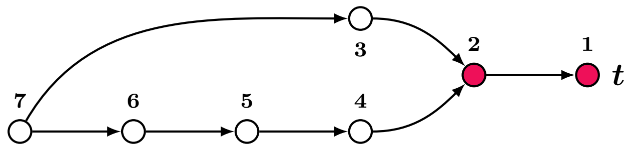
Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

‘given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G’

ABox: $\mathcal{A}_{G,t} = \{\,\text{edge}(v_1, v_2) \mid (v_1, v_2) \in E\,\} \cup \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} = \{\,\exists\text{edge.ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget}\,\}$



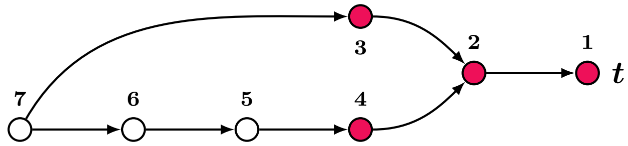
Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

'given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G'

ABox: $\mathcal{A}_{G,t} = \{\, \text{edge}(v_1, v_2) \mid (v_1, v_2) \in E \,\} \cup \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} = \{\, \exists \text{edge}.\text{ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget} \,\}$



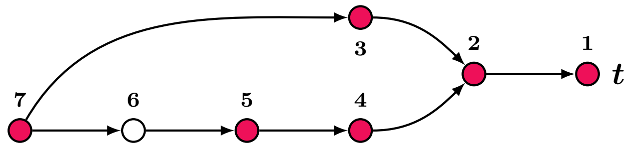
Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

'given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G'

ABox: $\mathcal{A}_{G,t} = \{\,\text{edge}(v_1, v_2) \mid (v_1, v_2) \in E\,\} \cup \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} = \{\,\exists\text{edge}.\text{ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget}\,\}$



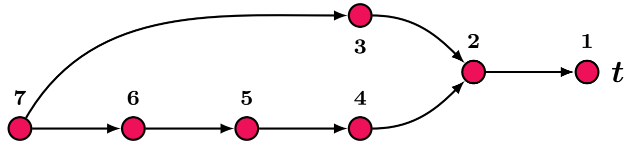
Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

'given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G'

ABox: $\mathcal{A}_{G,t} = \{\, \text{edge}(v_1, v_2) \mid (v_1, v_2) \in E \,\} \cup \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} = \{\, \exists \text{edge.ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget} \,\}$



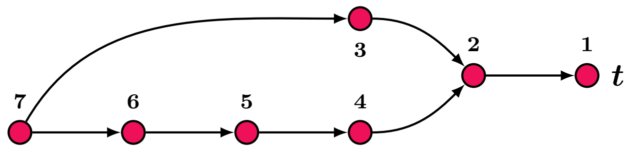
Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

'given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G'

ABox: $\mathcal{A}_{G,t} \;=\; \{\,\text{edge}(v_1, v_2) \mid (v_1, v_2) \in E\,\} \;\cup\; \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} \;=\; \{\,\exists\text{edge.ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget}\,\}$



CQ: $q \leftarrow \text{ReachableFromTarget}(s)$

$(\mathcal{T}, \mathcal{A}_{G,t}) \models q$ iff there is a path from s to t in G

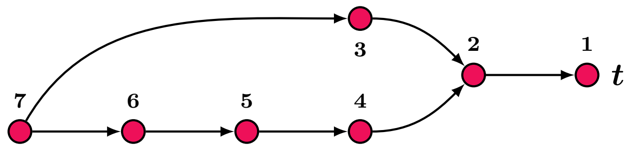
Can We Use $\exists R.A \sqsubseteq B$ in QL?

reachability problem for directed graphs is **NLogSpace-complete**:

'given a directed graph $G = (V, E)$ and $s, t \in V$, decide whether there is a directed path from s to t in G'

ABox: $\mathcal{A}_{G,t} = \{\, \text{edge}(v_1, v_2) \mid (v_1, v_2) \in E \,\} \cup \{\text{ReachableFromTarget}(t)\}$

TBox: $\mathcal{T} = \{\, \exists\text{edge}.\text{ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget} \,\}$



CQ: $q \leftarrow \text{ReachableFromTarget}(s)$

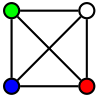
$(\mathcal{T}, \mathcal{A}_{G,t}) \models q$ iff there is a path from s to t in G

$\mathcal{T}$ and q do not depend on G, s, t

→ '$(\mathcal{T}, \mathcal{A}_{G,t}) \models q$?' is **NLogSpace**-hard for data complexity

→ q and $\mathcal{T}$ are not FO-rewritable

Can We Use $A \sqsubseteq B \sqcup C$ in QL?



graph 3-colouring problem is **NP-complete**:

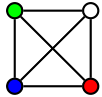
'given a graph $G = (V, E)$, decide whether its vertices can be painted in one of three colours so that no adjacent vertices have the same colour'

represent G as the ABox $\mathcal{A}_G = \{ R(v_1, v_2) \mid \{v_1, v_2\} \in E \}$

3-colouring is encoded by the TBox $\mathcal{T}$ with the axioms

$$\top \sqsubseteq C_1 \sqcup C_2 \sqcup C_3, \qquad C_i \sqcap C_j \sqsubseteq \bot, \qquad C_i \sqcap \exists R.C_i \sqsubseteq B, \quad 1 \leq i \leq 3$$

Can We Use $A \sqsubseteq B \sqcup C$ in QL?



graph 3-colouring problem is **NP-complete**:

'given a graph $G = (V, E)$, decide whether its vertices can be painted in one of three colours so that no adjacent vertices have the same colour'

represent G as the ABox $\mathcal{A}_G = \{ R(v_1, v_2) \mid \{v_1, v_2\} \in E \}$

3-colouring is encoded by the TBox $\mathcal{T}$ with the axioms

$$\top \sqsubseteq C_1 \sqcup C_2 \sqcup C_3, \qquad C_i \sqcap C_j \sqsubseteq \bot, \qquad C_i \sqcap \exists R.C_i \sqsubseteq B, \quad 1 \leq i \leq 3$$

consider CQ $q \leftarrow B(y)$

$(\mathcal{T}, \mathcal{A}_G) \not\models q$ iff G is 3-colourable

$\mathcal{T}$ and q do not depend on G

→ '$(\mathcal{T}, \mathcal{A}_G) \models q$?' is **coNP**-hard for data complexity

→ q and $\mathcal{T}$ are not FO-rewritable

OWL 2 QL as TGDs

(aka Datalog$^{\pm}$ aka existential rules)

concept inclusion		tuple-generating dependency
PhDStudent $\sqsubseteq$ Student	$\approx$	$\forall x\,(\text{PhDStudent}(x) \rightarrow \text{Student}(x))$
Student $\sqsubseteq$ $\exists$HasTutor	$\approx$	$\forall x\,(\text{Student}(x) \rightarrow \exists y\,\text{HasTutor}(x, y))$
...		...

OWL 2 QL as TGDs

(aka Datalog$^{\pm}$ aka existential rules)

concept inclusion		tuple-generating dependency
PhDStudent $\sqsubseteq$ Student	$\approx$	$\forall x\,(\text{PhDStudent}(x) \to \text{Student}(x))$
Student $\sqsubseteq$ $\exists$HasTutor	$\approx$	$\forall x\,(\text{Student}(x) \to \exists y\, \text{HasTutor}(x, y))$
...		...

TGDs: $$\forall \vec{x} \forall \vec{y}\, \big(\varphi(\vec{x}, \vec{y}) \to \exists \vec{z}\, \psi(\vec{x}, \vec{z})\big)$$

φ and ψ are conjunctions of predicate atoms

OWL 2 QL as TGDs

(aka Datalog$^{\pm}$ aka existential rules)

concept inclusion		tuple-generating dependency
PhDStudent $\sqsubseteq$ Student	$\approx$	$\forall x\,(\text{PhDStudent}(x) \to \text{Student}(x))$
Student $\sqsubseteq$ $\exists$HasTutor	$\approx$	$\forall x\,(\text{Student}(x) \to \exists y\,\text{HasTutor}(x, y))$
...		...

TGDs: $\forall \vec{x} \forall \vec{y}\,(\varphi(\vec{x}, \vec{y}) \to \exists \vec{z}\,\psi(\vec{x}, \vec{z}))$

φ and ψ are conjunctions of predicate atoms

linear TGDs: φ and ψ are atoms (all *OWL 2 QL* axioms are linear)

Calì, Gottlob & Pieris, 2010: sets of sticky TGDs are FO-rewritable

in particular, linear TGDs

OWL 2 QL as TGDs

(aka Datalog$^{\pm}$ aka existential rules)

concept inclusion		tuple-generating dependency
PhDStudent $\sqsubseteq$ Student	$\approx$	$\forall x\,(\text{PhDStudent}(x) \to \text{Student}(x))$
Student $\sqsubseteq$ $\exists$HasTutor	$\approx$	$\forall x\,(\text{Student}(x) \to \exists y\,\text{HasTutor}(x, y))$
...		...

TGDs: $\forall \vec{x} \forall \vec{y}\,(\varphi(\vec{x}, \vec{y}) \to \exists \vec{z}\,\psi(\vec{x}, \vec{z}))$

φ and ψ are conjunctions of predicate atoms

linear TGDs: φ and ψ are atoms (all *OWL 2 QL* axioms are linear)

Calì, Gottlob & Pieris, 2010: sets of sticky TGDs are FO-rewritable

in particular, linear TGDs

NB: these TGDs are used with the **open world assumption** (for enriching data)

TGDs in **DBs** are used with the **closed world assumption** (integrity constraints)

Practical Query Answering in OWL 2 QL

systems

- QuOnto (Rome, 2005)
- REQUIEM (Oxford, 2009) / Stardog (Washington, DC, 2011)
- Presto (Rome, 2010)
- IQAROS (Athens, 2011)
- Rapid (Athens-Oxford, 2011)
- Nyaya (Milan-Oxford, 2010) for TGDs
- Clipper (Vienna, 2012) for Horn-SHIQ
- kyrie (Madrid, 2013)
- Pure (Montpellier, 2013) for TGDs
- Quest/ontop (Bolzano, 2011)

Practical Query Answering in OWL 2 QL

systems

- QuOnto (Rome, 2005)
- REQUIEM (Oxford, 2009) / Stardog (Washington, DC, 2011)
- Presto (Rome, 2010)
- IQAROS (Athens, 2011)
- Rapid (Athens-Oxford, 2011)
- Nyaya (Milan-Oxford, 2010) for TGDs
- Clipper (Vienna, 2012) for Horn-SHIQ
- kyrie (Madrid, 2013)
- Pure (Montpellier, 2013) for TGDs
- Quest/ontop (Bolzano, 2011)

not so smoothly: the size of implemented rewritings q' is $O\big((|q| \cdot |\mathcal{T}|)^{|q|}\big)$

(can't say 'query is small or fixed' any longer)

Does a Rewriting Have to be Exponential?

TBox	mother $\sqsubseteq$ parent and father $\sqsubseteq$ parent
query	grandparent$(x, z) \leftarrow$ parent$(x, y) \wedge$ parent(y, z)

Does a Rewriting Have to be Exponential?

TBox: mother $\sqsubseteq$ parent and father $\sqsubseteq$ parent

query: grandparent(x, z) $\leftarrow$ parent(x, y) $\wedge$ parent(y, z)

UCQ-rewritings (unions of CQs) are **exponential** in the worst case

grandparent(x, z) $\leftarrow$ parent(x, y) $\wedge$ parent(y, z)
grandparent(x, z) $\leftarrow$ father(x, y) $\wedge$ father(y, z)
grandparent(x, z) $\leftarrow$ mother(x, y) $\wedge$ father(y, z)
...

Does a Rewriting Have to be Exponential?

TBox: mother $\sqsubseteq$ parent and father $\sqsubseteq$ parent

query: grandparent$(x, z) \leftarrow$ parent$(x, y) \land$ parent(y, z)

UCQ-rewritings (unions of CQs) are **exponential** in the worst case

grandparent$(x, z) \leftarrow$ parent$(x, y) \land$ parent(y, z)
grandparent$(x, z) \leftarrow$ father$(x, y) \land$ father(y, z)
grandparent$(x, z) \leftarrow$ mother$(x, y) \land$ father(y, z)
$\dots$

PE-rewritings (positive existential queries $\approx$ select-project-join-union) $\exists \; \lor \; \land$

grandparent$(x, z) \leftarrow$ (parent$(x, y) \lor$ father$(x, y) \lor$ mother$(x, y)) \land$
(parent$(y, z) \lor$ father$(y, z) \lor$ mother(y, z))

Does a Rewriting Have to be Exponential?

TBox: mother $\sqsubseteq$ parent and father $\sqsubseteq$ parent

query: grandparent$(x, z) \leftarrow$ parent$(x, y) \land$ parent(y, z)

UCQ-rewritings (unions of CQs) are **exponential** in the worst case

grandparent$(x, z) \leftarrow$ parent$(x, y) \land$ parent(y, z)
grandparent$(x, z) \leftarrow$ father$(x, y) \land$ father(y, z)
grandparent$(x, z) \leftarrow$ mother$(x, y) \land$ father(y, z)
...

PE-rewritings (positive existential queries ≈ select-project-join-union)

$\exists \ \lor \ \land$

grandparent$(x, z) \leftarrow$ (parent$(x, y) \lor$ father$(x, y) \lor$ mother$(x, y)) \land$
(parent$(y, z) \lor$ father$(y, z) \lor$ mother(y, z)))

NDL-rewriting (non-recursive Datalog ≈ SQL with views)

$\exists \ \lor \ \land$ + structure sharing

grandparent$(x, z) \leftarrow$ ext-parent$(x, y) \land$ ext-parent(y, z)
ext-parent$(x, y) \leftarrow$ parent(x, y)
ext-parent$(x, y) \leftarrow$ father(x, y)
ext-parent$(x, y) \leftarrow$ mother(x, y)

Does a Rewriting Have to be Exponential?

TBox	mother $\sqsubseteq$ parent and father $\sqsubseteq$ parent
query	grandparent$(x, z) \leftarrow$ parent$(x, y) \land$ parent(y, z)

UCQ-rewritings (unions of CQs) are **exponential** in the worst case

grandparent$(x, z) \leftarrow$ parent$(x, y) \land$ parent(y, z)
grandparent$(x, z) \leftarrow$ father$(x, y) \land$ father(y, z)
grandparent$(x, z) \leftarrow$ mother$(x, y) \land$ father(y, z)
...

PE-rewritings (positive existential queries ≈ select-project-join-union) $\exists \ \lor \ \land$

grandparent$(x, z) \leftarrow$ (parent$(x, y) \lor$ father$(x, y) \lor$ mother$(x, y)) \land$
(parent$(y, z) \lor$ father$(y, z) \lor$ mother(y, z))

NDL-rewriting (non-recursive Datalog ≈ SQL with views) $\exists \ \lor \ \land$ + structure sharing

grandparent$(x, z) \leftarrow$ ext-parent$(x, y) \land$ ext-parent(y, z)
ext-parent$(x, y) \leftarrow$ parent(x, y)
ext-parent$(x, y) \leftarrow$ father(x, y)
ext-parent$(x, y) \leftarrow$ mother(x, y)

FO-rewriting (first-order queries ≈ SQL) $\exists \forall \ \lor \ \land \neg$

Case 1: Flat QL TBoxes

a TBox $\mathcal{T}$ is **flat** if it does not contain **generating axioms** $B' \sqsubseteq \exists R.B$

≈ RDF Schema

Case 1: Flat QL TBoxes

a TBox $\mathcal{T}$ is **flat** if it does not contain **generating axioms** $B' \sqsubseteq \exists R.B$

≈ RDF Schema

$q(\vec{x})$ and a flat $\mathcal{T}$ → $q_{\text{ext}}(\vec{x})$ by replacing

$$A(u) \longrightarrow \bigvee_{\mathcal{T} \models A' \sqsubseteq A} A'(u) \lor \bigvee_{\mathcal{T} \models \exists R \sqsubseteq A} \exists v\, R(u,v)$$

$$P(u,v) \longrightarrow \bigvee_{\mathcal{T} \models R \sqsubseteq P} R(u,v)$$

for any CQ $q(\vec{x})$ and any flat OWL 2 QL TBox $\mathcal{T}$,
$q_{\text{ext}}(\vec{x})$ is a PE-rewriting of q and $\mathcal{T}$ of size $O(|q| \cdot |\mathcal{T}|)$

Case 1: Flat QL TBoxes

a TBox $\mathcal{T}$ is **flat** if it does not contain **generating axioms** $B' \sqsubseteq \exists R.B$

$\approx$ RDF Schema

$q(\vec{x})$ and a flat $\mathcal{T}$ $\longrightarrow$ $q_{\mathsf{ext}}(\vec{x})$ by replacing

$$A(u) \longrightarrow \bigvee_{\mathcal{T} \models A' \sqsubseteq A} A'(u) \lor \bigvee_{\mathcal{T} \models \exists R \sqsubseteq A} \exists v\, R(u, v)$$

$$P(u, v) \longrightarrow \bigvee_{\mathcal{T} \models R \sqsubseteq P} R(u, v)$$

for any CQ $q(\vec{x})$ and any flat OWL 2 QL TBox $\mathcal{T}$,
$q_{\mathsf{ext}}(\vec{x})$ is a PE-rewriting of q and $\mathcal{T}$ of size $O(|q| \cdot |\mathcal{T}|)$

easy in theory, not so in practice

Who Works with Professors?

TBox:

$$\text{worksOn}^{-} \sqsubseteq \text{involves}$$
$$\text{isManagedBy} \sqsubseteq \text{involves}$$

in English: find those who work with professors

query: $q(x) \leftarrow \text{worksOn}(x, y) \wedge \text{involves}(y, z) \wedge \text{Professor}(z)$

Who Works with Professors?

TBox:

$\text{worksOn}^- \sqsubseteq \text{involves}$

$\text{isManagedBy} \sqsubseteq \text{involves}$

in English: find those who work with professors

query: $q(x) \leftarrow \text{worksOn}(x, y) \wedge \text{involves}(y, z) \wedge \text{Professor}(z)$

$\text{worksOn}(z, y) \vee \text{isManagedBy}(y, z) \vee \text{involves}(y, z)$

Rewriting over H-complete ABoxes

an ABox $\mathcal{A}$ is **H-complete with respect to $\mathcal{T}$** if

- $A(a) \in \mathcal{A}$ whenever $A'(a) \in \mathcal{A}$ and $\mathcal{T} \models A' \sqsubseteq A$
- $A(a) \in \mathcal{A}$ whenever $R(a, b) \in \mathcal{A}$ and $\mathcal{T} \models \exists R \sqsubseteq A$
- $P(a, b) \in \mathcal{A}$ whenever $R(a, b) \in \mathcal{A}$ and $\mathcal{T} \models R \sqsubseteq P$

Rewriting over H-complete ABoxes

an ABox $\mathcal{A}$ is **H-complete with respect to $\mathcal{T}$** if

- $A(a) \in \mathcal{A}$ whenever $A'(a) \in \mathcal{A}$ and $\mathcal{T} \models A' \sqsubseteq A$
- $A(a) \in \mathcal{A}$ whenever $R(a, b) \in \mathcal{A}$ and $\mathcal{T} \models \exists R \sqsubseteq A$
- $P(a, b) \in \mathcal{A}$ whenever $R(a, b) \in \mathcal{A}$ and $\mathcal{T} \models R \sqsubseteq P$

an FO-query $q'(\vec{x})$ is an **FO-rewriting of $q(\vec{x})$ and $\mathcal{T}$ over H-complete ABoxes** if,

for any H-complete (w.r.t. $\mathcal{T}$) ABox $\mathcal{A}$ and any $\vec{a}$,

$(\mathcal{T}, \mathcal{A}) \models q(\vec{a})$ iff $\mathcal{A} \models q'(\vec{a})$

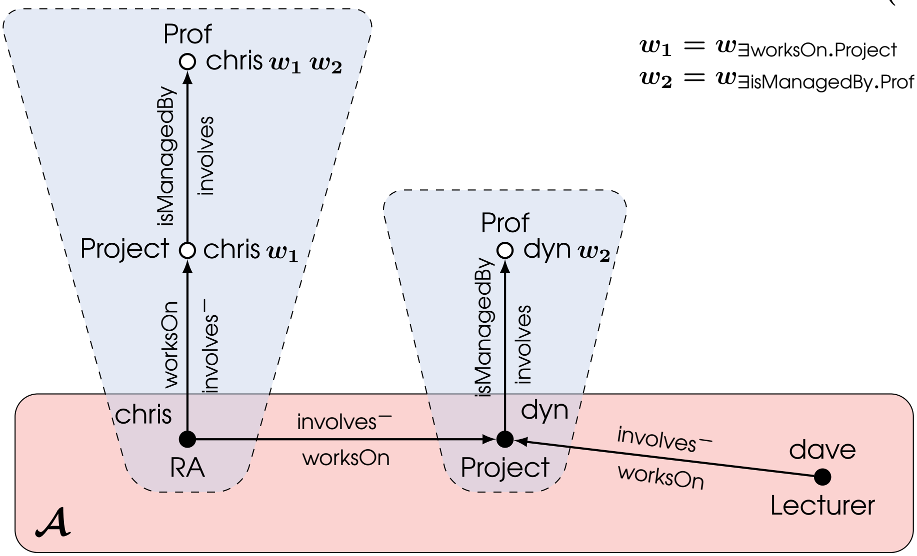
(thus we ignore the axioms considered in the flat rewriting)

Case 2: Who Works with Professors (2)?

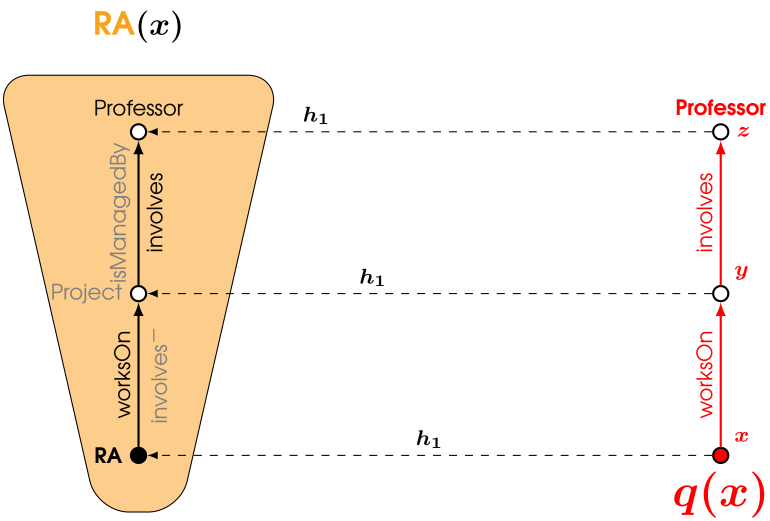
$\mathcal{T}$: RA $\sqsubseteq$ ∃worksOn.Project   worksOn$^-$ $\sqsubseteq$ involves
Project $\sqsubseteq$ ∃isManagedBy.Prof   isManagedBy $\sqsubseteq$ involves

$\mathcal{A}$: RA(chris), worksOn(chris, dyn), Project(dyn), Lecturer(dave),
worksOn(dave, dyn)

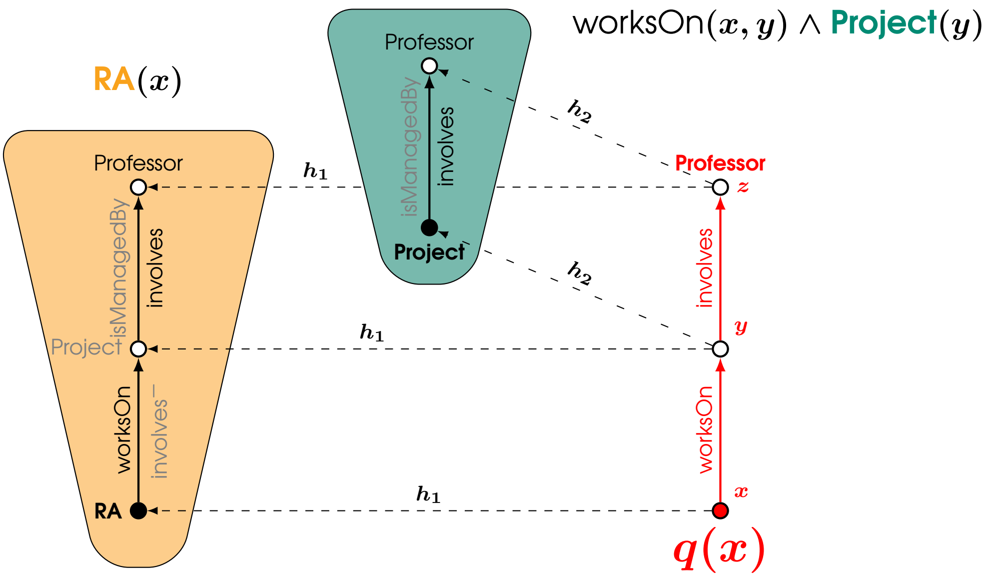
$w_1 = w_{\exists\text{worksOn.Project}}$
$w_2 = w_{\exists\text{isManagedBy.Prof}}$



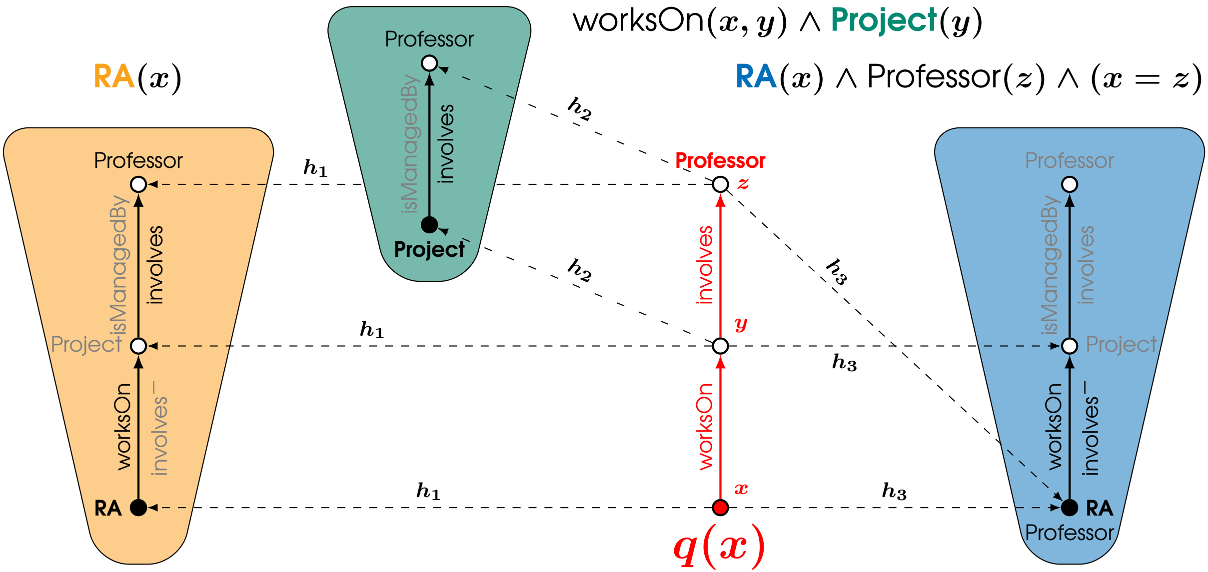
Case 2: Rewriting the Labelled Nulls



Case 2: Rewriting the Labelled Nulls

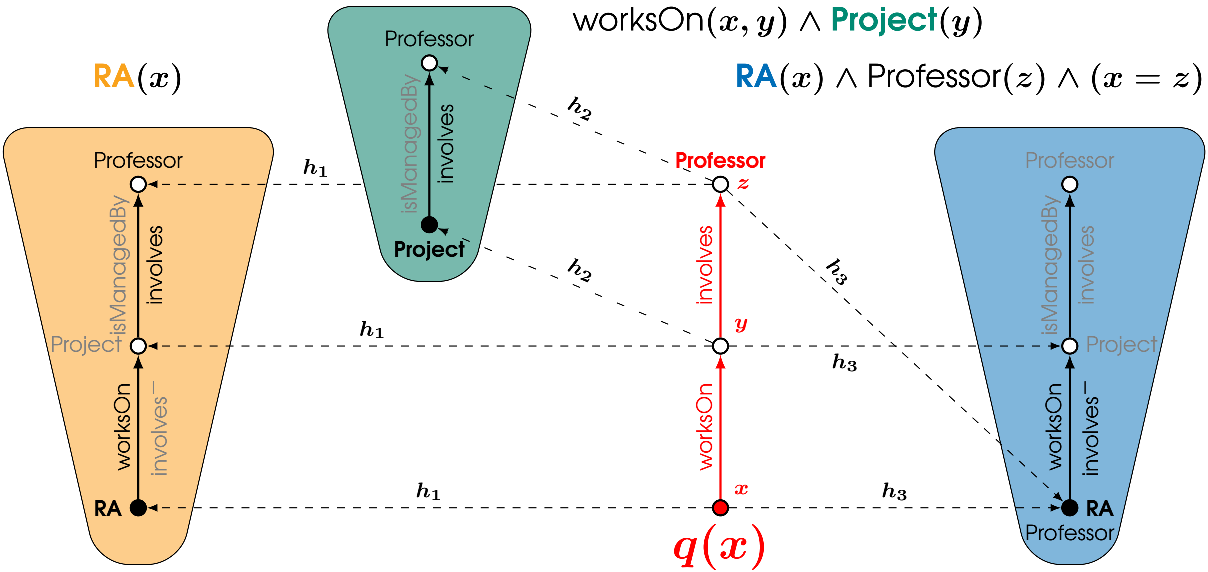


Case 2: Rewriting the Labelled Nulls



(x and z are the **roots** of the tree witness)

Case 2: Rewriting the Labelled Nulls



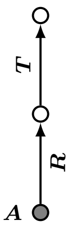
(x and z are the **roots** of the tree witness)

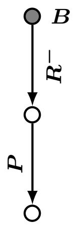
PE-rewriting (over H-complete ABoxes):

$$q'(x) \leftarrow \text{RA}(x) \lor (\text{worksOn}(x, y) \land \text{Project}(y)) \lor$$
$$(\text{RA}(x) \land \text{Professor}(z) \land (x = z)) \lor$$
$$(\text{worksOn}(x, y) \land \text{involves}(y, z) \land \text{Professor}(z))$$

Tree-Witness Rewriting

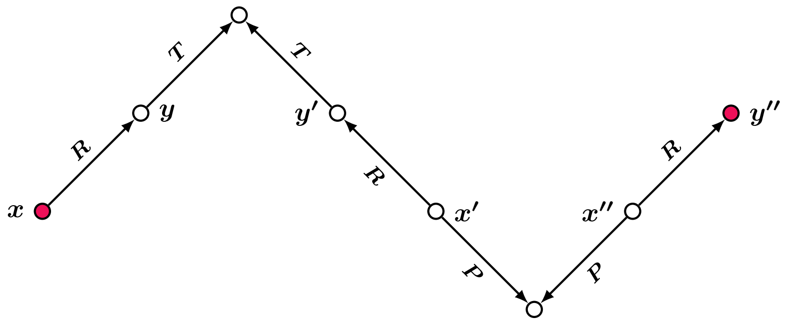
TBox $\mathcal{T}$: $\quad A \sqsubseteq \exists R, \quad \exists R^- \sqsubseteq \exists T, \quad B \sqsubseteq \exists R^-, \quad \exists R \sqsubseteq \exists S$





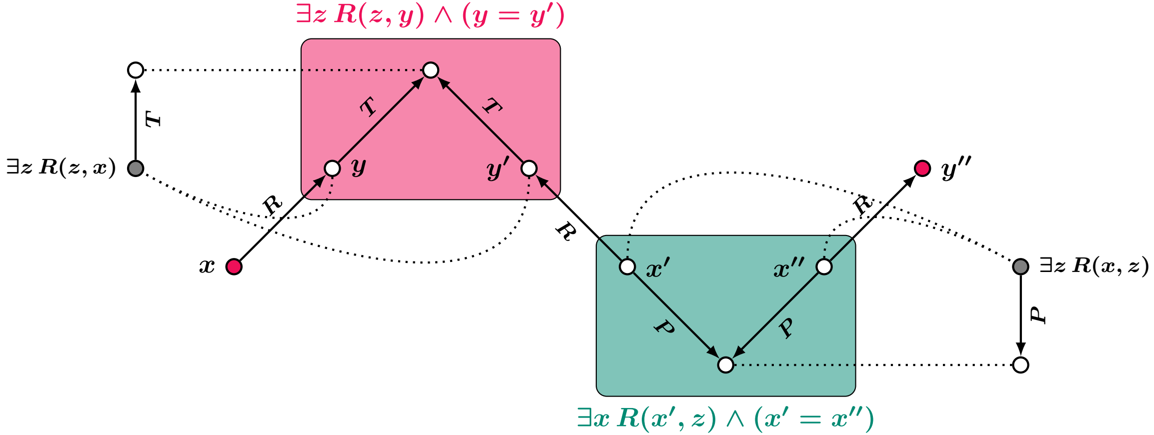
Tree-Witness Rewriting

TBox $\mathcal{T}$: $\quad A \sqsubseteq \exists R, \quad \exists R^- \sqsubseteq \exists T, \quad B \sqsubseteq \exists R^-, \quad \exists R \sqsubseteq \exists S$



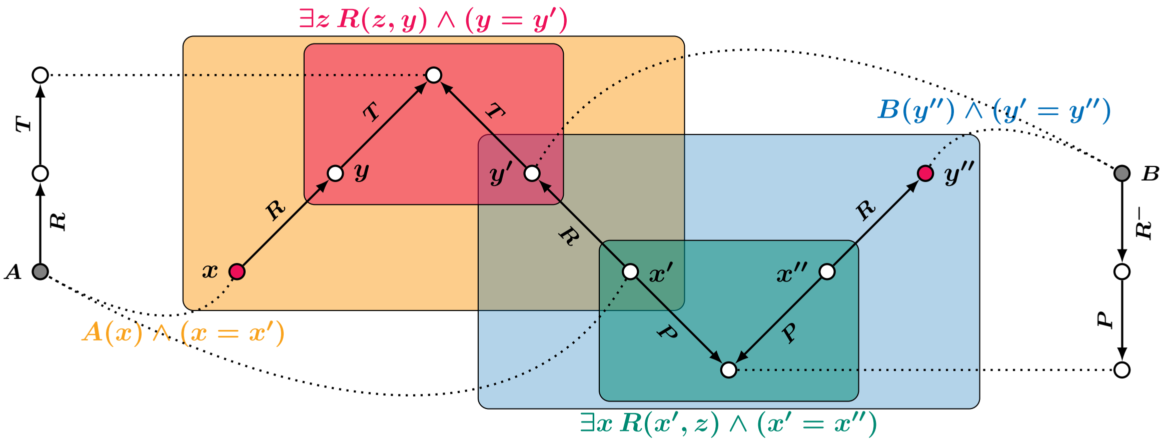
Tree-Witness Rewriting

TBox $\mathcal{T}$: $A \sqsubseteq \exists R, \quad \exists R^- \sqsubseteq \exists T, \quad B \sqsubseteq \exists R^-, \quad \exists R \sqsubseteq \exists S$



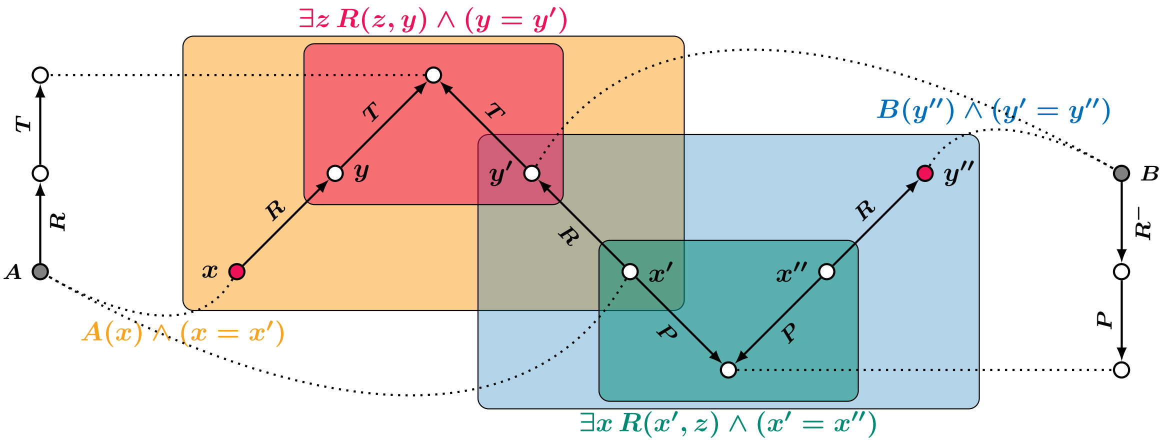
Tree-Witness Rewriting

TBox $\mathcal{T}$: $\quad A \sqsubseteq \exists R, \quad \exists R^- \sqsubseteq \exists T, \quad B \sqsubseteq \exists R^-, \quad \exists R \sqsubseteq \exists S$



Tree-Witness Rewriting

TBox $\mathcal{T}$: $A \sqsubseteq \exists R, \quad \exists R^- \sqsubseteq \exists T, \quad B \sqsubseteq \exists R^-, \quad \exists R \sqsubseteq \exists S$



$$q_{\mathsf{tw}}(\vec{x}) \;=\; \bigvee_{\Theta \text{ independent set of tree witnesses}} \exists \vec{y}\, \Big(\bigwedge_{S(\vec{z}) \in q \setminus q_\Theta} S(\vec{z}) \;\wedge\; \bigwedge_{\mathfrak{t} \in \Theta} \mathsf{tw}_{\mathfrak{t}} \Big)$$

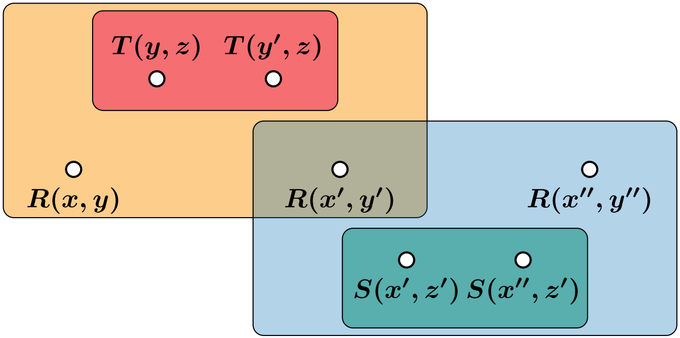
(Kikot, K & Zakharyaschev, 2012)

Rewritings as Boolean Functions

hypergraph H:

vertices = atoms of the query

hyperedges = tree witnesses

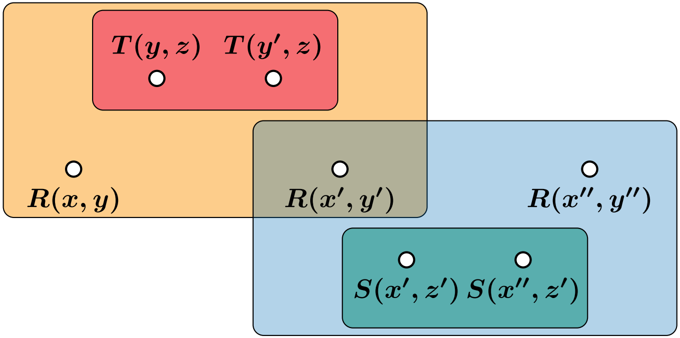


Rewritings as Boolean Functions

hypergraph H:

vertices = atoms of the query

hyperedges = tree witnesses



hypergraph function of $H = (V, E)$:

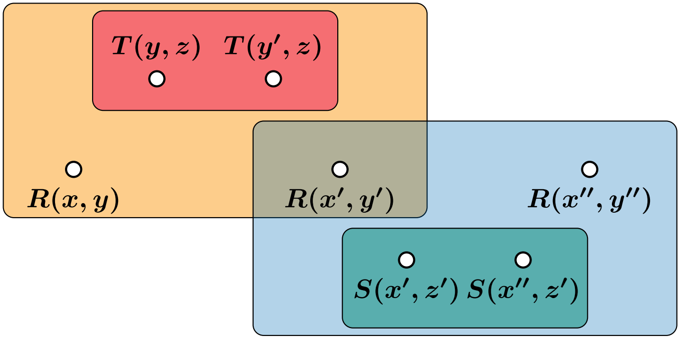
$$f_H = \bigvee_{\substack{X \subseteq E \\ X \text{ independent}}} \Big(\bigwedge_{v \in V \setminus V_X} p_v \wedge \bigwedge_{e \in X} p_e \Big)$$

Rewritings as Boolean Functions

hypergraph H:

vertices = atoms of the query

hyperedges = tree witnesses



hypergraph function of $H = (V, E)$:

$$f_H = \bigvee_{\substack{X \subseteq E \\ X \text{ independent}}} \Big(\bigwedge_{v \in V \setminus V_X} p_v \wedge \bigwedge_{e \in X} p_e \Big)$$

(Kikot, K, Podolskii & Zakharyaschev, 2012) lower bounds from

circuit complexity

- **exponential** non-recursive datalog (and positive existential) rewritings
- **superpolynomial** first-order rewritings (unless NP $\subseteq$ P/poly)

Short Rewritings in Theory

if $q_{\mathfrak{t}_1} \cap q_{\mathfrak{t}_2} = \emptyset$ or $q_{\mathfrak{t}_1} \subseteq q_{\mathfrak{t}_2}$ or $q_{\mathfrak{t}_2} \subseteq q_{\mathfrak{t}_1}$, for each pair $\underbrace{\mathfrak{t}_1 \text{ and } \mathfrak{t}_2}_{\text{compatible}}$, then

$$q'_{\mathsf{tw}}(\vec{x}) = \bigwedge_{S(\vec{z}) \in q} \Big(S(\vec{z}) \lor \bigvee_{\mathfrak{t}\colon S(\vec{z}) \in q_{\mathfrak{t}}} \mathsf{tw}_{\mathfrak{t}} \Big)$$

is a **rewriting** (over H-complete ABoxes)

Short Rewritings in Theory

if $q_{t_1} \cap q_{t_2} = \emptyset$ or $q_{t_1} \subseteq q_{t_2}$ or $q_{t_2} \subseteq q_{t_1}$, for each pair $\underbrace{\mathfrak{t}_1 \text{ and } \mathfrak{t}_2}_{\text{compatible}}$, then

$$q'_{\mathsf{tw}}(\vec{x}) = \bigwedge_{S(\vec{z}) \in q} \Big(S(\vec{z}) \lor \bigvee_{\mathfrak{t}:\ S(\vec{z}) \in q_{\mathfrak{t}}} \mathsf{tw}_{\mathfrak{t}} \Big)$$

is a **rewriting** (over H-complete ABoxes)

QL: replace $S(\vec{z})$ with $\bigvee_{\mathcal{T} \models S' \sqsubseteq S} S'(\vec{z})$

$\Longrightarrow$ **polynomial positive existential rewriting**

provided that the number of tree witnesses is **polynomial** and they are **compatible**

not the case in general!

Part 4

Practical OBDA with Ontop

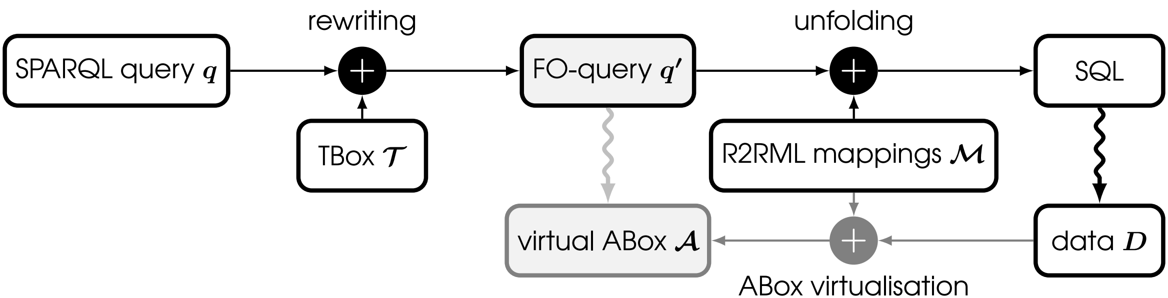
OBDA system Ontop

`http://ontop.inf.unibz.it`

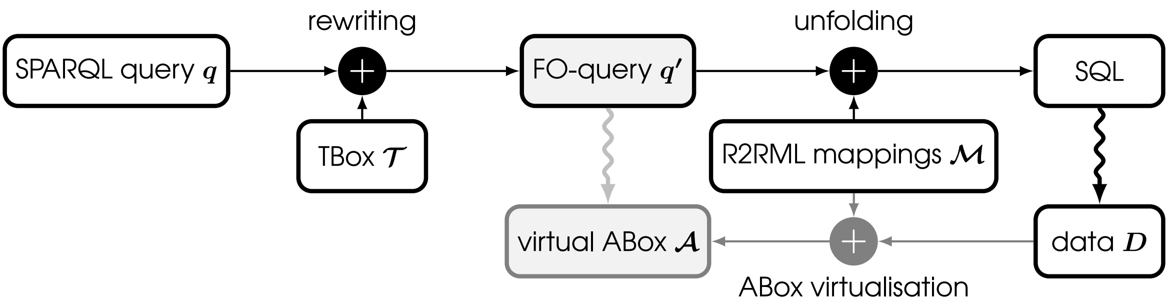
- implemented at the Free University of Bozen-Bolzano
  (Mariano Rodríguez-Muro, Martin Rezk, Guohui Xiao)
- open-source
- available as a plugin for Protégé 4 & 5, SPARQL end-point,
  OWL API and Sesame libraries

ontop framework

OBDA with Databases



OBDA with Databases



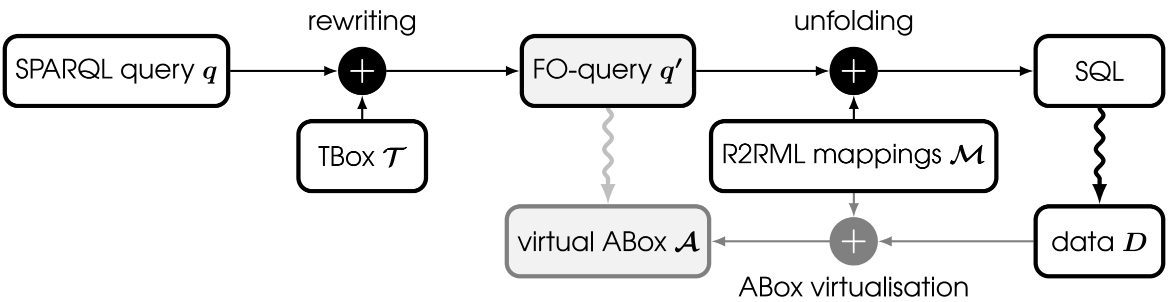
Why SQL rewritings are large:

(1) a large number of tree witnesses

(2) large concept/role hierarchies in OWL 2 QL ontology $\mathcal{T}$

(3) multiple definitions of the ontology terms in R2RML mappings $\mathcal{M}$

OBDA with Databases



Why SQL rewritings are large:

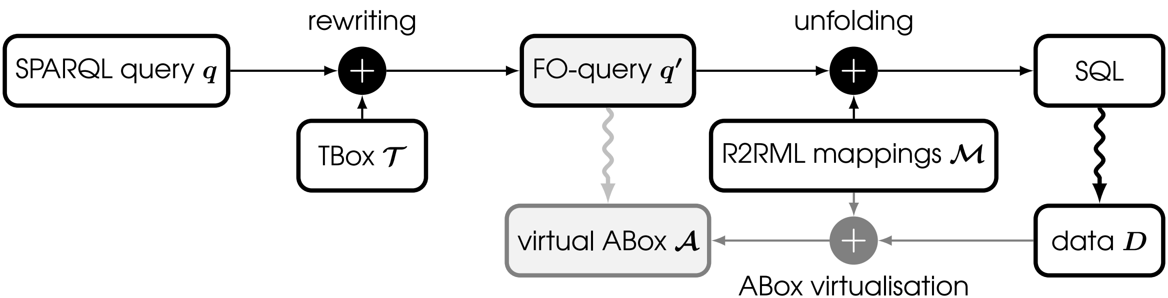
(1) a large number of free witnesses

very few for real-world CQs/ontologies

(2) large concept/role hierarchies in OWL 2 QL ontology $\mathcal{T}$

(3) multiple definitions of the ontology terms in R2RML mappings $\mathcal{M}$

OBDA with Databases



Why SQL rewritings are large:

(1) a large number of tree witnesses

very few for real-world CQs/ontologies

(2) large concept/role hierarchies in OWL 2 QL ontology $\mathcal{T}$

many inclusions in $\mathcal{T}$ follow from Σ and $\mathcal{M}$

(3) multiple definitions of the ontology terms in R2RML mappings $\mathcal{M}$

Ontop Example

IMDb (simplified): `http://www.imdb.com/interfaces`

– database

title

movie ID	title	production year
$\mathbf{728}$	'Django Unchained'	$\mathbf{2012}$
...	...	...

castinfo

person ID	movie ID	person role
n$\mathbf{37}$	$\mathbf{728}$	$\mathbf{1}$
n$\mathbf{38}$	$\mathbf{728}$	$\mathbf{1}$
...	...	...

– dependencies

$$\forall m \, \big(\exists p, r \; \mathit{castinfo}(p, m, r) \to \exists t, y \; \mathit{title}(m, t, y)\big) \quad \text{(FK)}$$

$$\forall m \forall t_1 \forall t_2 \, \big(\exists y \; \mathit{title}(m, t_1, y) \land \exists y \; \mathit{title}(m, t_2, y) \to (t_1 = t_2)\big) \quad (\text{PK}_1)$$

$$\forall m \forall y_1 \forall y_2 \, \big(\exists t \; \mathit{title}(m, t, y_1) \land \exists t \; \mathit{title}(m, t, y_2) \to (y_1 = y_2)\big) \quad (\text{PK}_2)$$

Ontop Example

IMDb (simplified): http://www.imdb.com/interfaces

- database

title

movie ID	title	production year
728	'Django Unchained'	**2012**
...	...	...

castinfo

person ID	movie ID	person role
n**37**	**728**	**1**
n**38**	**728**	**1**
...	...	...

- dependencies

$$\forall m\,(\exists p, r\ \mathit{castinfo}(p, m, r) \rightarrow \exists t, y\ \mathit{title}(m, t, y)) \qquad \text{(FK)}$$

$$\forall m \forall t_1 \forall t_2\,(\exists y\ \mathit{title}(m, t_1, y) \wedge \exists y\ \mathit{title}(m, t_2, y) \rightarrow (t_1 = t_2)) \qquad (\text{PK}_1)$$

$$\forall m \forall y_1 \forall y_2\,(\exists t\ \mathit{title}(m, t, y_1) \wedge \exists t\ \mathit{title}(m, t, y_2) \rightarrow (y_1 = y_2)) \qquad (\text{PK}_2)$$

Movie Ontology MO http://www.movieontology.org

$\mathit{mo{:}Movie} \equiv \exists \mathit{mo{:}title}$, $\quad \mathit{mo{:}Movie} \sqsubseteq \exists \mathit{mo{:}year}$,

$\mathit{mo{:}Movie} \equiv \exists \mathit{mo{:}cast}$, $\quad \exists \mathit{mo{:}cast}^{-} \sqsubseteq \mathit{mo{:}Person}, \ldots$

Ontop Example

IMDb (simplified): http://www.imdb.com/interfaces

– database

title

movie ID	title	production year
728	'Django Unchained'	2012
...	...	...

castinfo

person ID	movie ID	person role
n37	728	1
n38	728	1
...	...	...

– dependencies

$$\forall m\,(\exists p, r\ \textit{castinfo}(p, m, r) \to \exists t, y\ \textit{title}(m, t, y)) \qquad \text{(FK)}$$

$$\forall m \forall t_1 \forall t_2\,(\exists y\ \textit{title}(m, t_1, y) \wedge \exists y\ \textit{title}(m, t_2, y) \to (t_1 = t_2)) \qquad (\text{PK}_1)$$

$$\forall m \forall y_1 \forall y_2\,(\exists t\ \textit{title}(m, t, y_1) \wedge \exists t\ \textit{title}(m, t, y_2) \to (y_1 = y_2)) \qquad (\text{PK}_2)$$

Movie Ontology MO http://www.movieontology.org

$$\textit{mo:Movie} \equiv \exists \textit{mo:title}, \qquad \textit{mo:Movie} \sqsubseteq \exists \textit{mo:year},$$

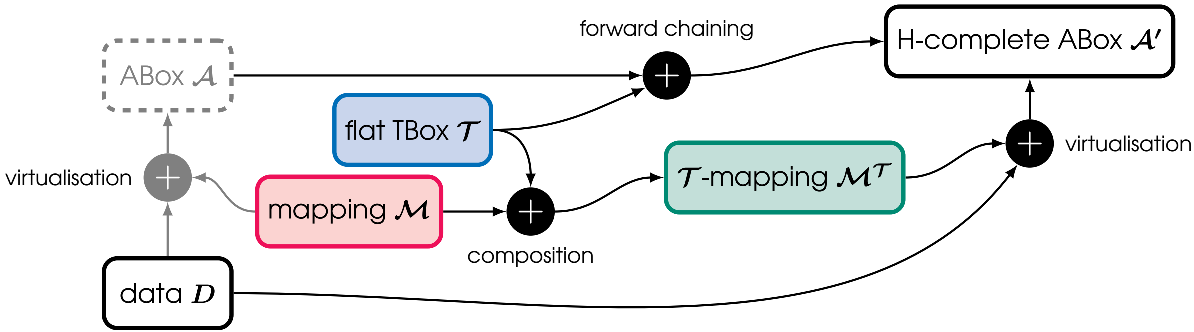
$$\textit{mo:Movie} \equiv \exists \textit{mo:cast}, \qquad \exists \textit{mo:cast}^- \sqsubseteq \textit{mo:Person}, \ldots$$

Mappings (created by the Ontop development team)

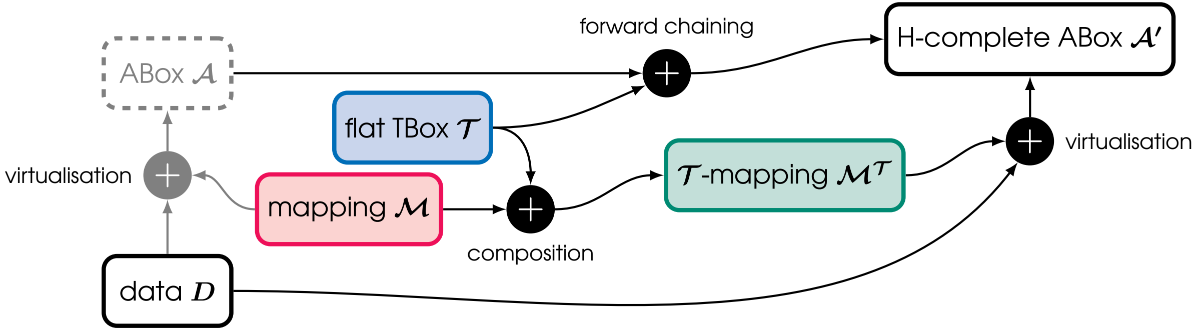
$$\textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y) \leftarrow \textit{title}(m, t, y) \qquad (\text{M}_1)$$

$$\textit{mo:cast}(m, p), \textit{mo:Person}(p) \leftarrow \textit{castinfo}(p, m, r) \qquad (\text{M}_2)$$

Ontop: T-mappings



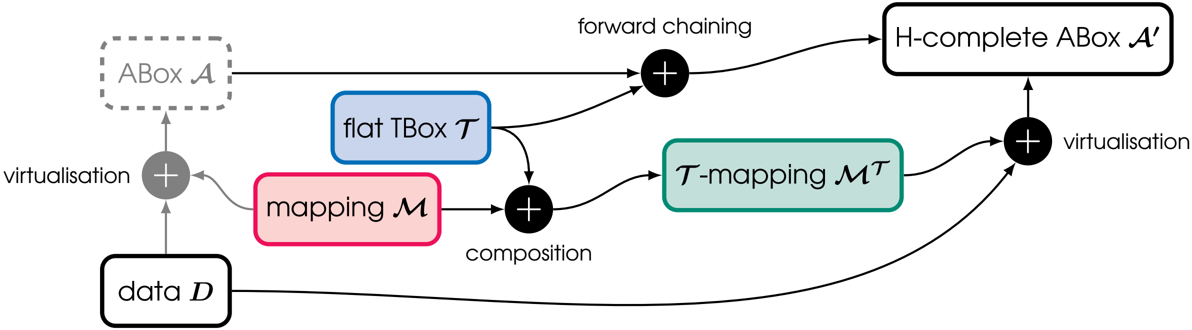
Ontop: T-mappings



$\mathcal{T}$

$mo{:}Movie \equiv \exists mo{:}title$, $\quad mo{:}Movie \sqsubseteq \exists mo{:}year$,
$mo{:}Movie \equiv \exists mo{:}cast$, $\quad \exists mo{:}cast^{-} \sqsubseteq mo{:}Person$

Ontop: T-mappings



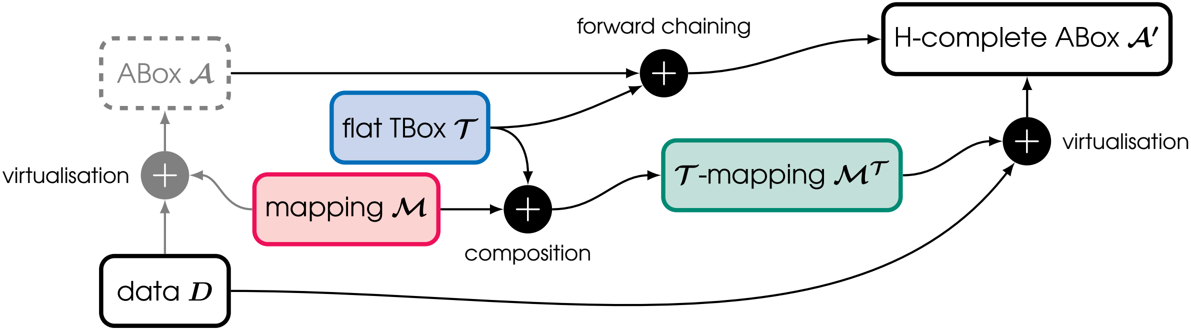
$\mathcal{T}$

$$\textit{mo:Movie} \equiv \exists \textit{mo:title}, \qquad \textit{mo:Movie} \sqsubseteq \exists \textit{mo:year},$$
$$\textit{mo:Movie} \equiv \exists \textit{mo:cast}, \qquad \exists \textit{mo:cast}^{-} \sqsubseteq \textit{mo:Person}$$

$\mathcal{M}$

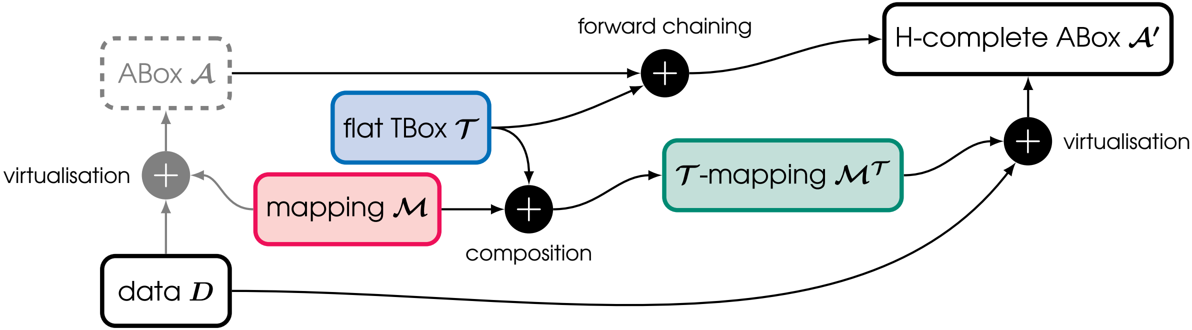
$$\textit{mo:Movie}(\boldsymbol{m}), \textit{mo:title}(\boldsymbol{m}, \boldsymbol{t}), \textit{mo:year}(\boldsymbol{m}, \boldsymbol{y}) \leftarrow \textit{title}(\boldsymbol{m}, \boldsymbol{t}, \boldsymbol{y}) \qquad (\mathrm{M}_1)$$
$$\textit{mo:cast}(\boldsymbol{m}, \boldsymbol{p}), \textit{mo:Person}(\boldsymbol{p}) \leftarrow \textit{castinfo}(\boldsymbol{p}, \boldsymbol{m}, \boldsymbol{r}) \qquad (\mathrm{M}_2)$$

Ontop: T-mappings



$\mathcal{T}$	*mo:Movie* $\equiv \exists$*mo:title*, *mo:Movie* $\sqsubseteq \exists$*mo:year*, *mo:Movie* $\equiv \exists$*mo:cast*, $\exists$*mo:cast*$^-$ $\sqsubseteq$ *mo:Person*
$\mathcal{M}$	*mo:Movie*$(\boldsymbol{m})$, *mo:title*$(\boldsymbol{m}, \boldsymbol{t})$, *mo:year*$(\boldsymbol{m}, \boldsymbol{y}) \leftarrow$ *title*$(\boldsymbol{m}, \boldsymbol{t}, \boldsymbol{y})$ (M_1) *mo:cast*$(\boldsymbol{m}, \boldsymbol{p})$, *mo:Person*$(\boldsymbol{p}) \leftarrow$ *castinfo*$(\boldsymbol{p}, \boldsymbol{m}, \boldsymbol{r})$ (M_2)
$\mathcal{M}^{\mathcal{T}}$	*mo:Movie*$(\boldsymbol{m}) \leftarrow$ *title*$(\boldsymbol{m}, \boldsymbol{t}, \boldsymbol{y})$ by (M_1) *mo:Movie*$(\boldsymbol{m}) \leftarrow$ *castinfo*$(\boldsymbol{p}, \boldsymbol{m}, \boldsymbol{r})$ by (M_2) + $\exists$*mo:cast* $\sqsubseteq$ *mo:Movie*

Ontop: T-mappings



$\mathcal{T}$

mo:Movie $\equiv \exists$*mo:title*, *mo:Movie* $\sqsubseteq \exists$*mo:year*,
mo:Movie $\equiv \exists$*mo:cast*, $\exists$*mo:cast*$^-$ $\sqsubseteq$ *mo:Person*

$\mathcal{M}$

mo:Movie(m), *mo:title*(m, t), *mo:year*$(m, y) \leftarrow$ *title*(m, t, y) (M_1)
mo:cast(m, p), *mo:Person*$(p) \leftarrow$ *castinfo*(p, m, r) (M_2)

$\mathcal{M}^{\mathcal{T}}$

mo:Movie$(m) \leftarrow$ *title*(m, t, y) by (M_1)
~~*mo:Movie*$(m) \leftarrow$ *castinfo*(p, m, r)~~ by (M_2) + $\exists$*mo:cast* $\sqsubseteq$ *mo:Movie*

redundant by (FK)

$\forall m\ (\exists p, r$ ***castinfo***$(p, m, r) \rightarrow \exists t, y$ ***title***$(m, t, y))$

Optimising T-mappings

- using foreign keys (inclusion dependencies)

Optimising T-mappings

- using foreign keys (inclusion dependencies)

- using disjunction

$\mathcal{T}$

mo:Actor $\sqsubseteq$ *mo:Artist*, *mo:Artist* $\sqsubseteq$ *mo:Person*,
mo:Director $\sqsubseteq$ *mo:Person*, *mo:Editor* $\sqsubseteq$ *mo:Person*, ...

$\mathcal{M}$

mo:Actor$(\boldsymbol{p}) \leftarrow$ *castinfo*$(\boldsymbol{p}, \boldsymbol{m}, \boldsymbol{r}), (\boldsymbol{r} = \mathbf{1})$ (M_1)

...

mo:Editor$(\boldsymbol{p}) \leftarrow$ *castinfo*$(\boldsymbol{p}, \boldsymbol{m}, \boldsymbol{r}), (\boldsymbol{r} = \mathbf{6})$ (M_6)

Optimising T-mappings

- using foreign keys (inclusion dependencies)

- using disjunction

$\mathcal{T}$

mo:Actor $\sqsubseteq$ *mo:Artist*, *mo:Artist* $\sqsubseteq$ *mo:Person*,
mo:Director $\sqsubseteq$ *mo:Person*, *mo:Editor* $\sqsubseteq$ *mo:Person*, ...

$\mathcal{M}$

mo:Actor$(p) \leftarrow$ *castinfo*$(p, m, r), (r = 1)$ (M$_1$)
...
mo:Editor$(p) \leftarrow$ *castinfo*$(p, m, r), (r = 6)$ (M$_6$)

$\mathcal{M}^{\mathcal{T}}$

mo:Person$(p) \leftarrow$ *castinfo*$(p, m, r), ((r = 1) \lor \cdots \lor (r = 6))$

Unfolding with Semantic Query Optimisation

Query

$$\boldsymbol{q}(\boldsymbol{t}, \boldsymbol{y}) \leftarrow \textit{mo:Movie}(\boldsymbol{m}), \textit{mo:title}(\boldsymbol{m}, \boldsymbol{t}), \textit{mo:year}(\boldsymbol{m}, \boldsymbol{y}), (\boldsymbol{y} > \mathbf{2010})$$

Unfolding with Semantic Query Optimisation

Query

$$q(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

Rewriting

$$q'(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

Unfolding with Semantic Query Optimisation

Query

$$q(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

Rewriting

$$q'(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

$\mathcal{M}$

$\textit{mo:Movie}(m) \leftarrow \textit{title}(m, t, y)$	(M$_1$)
$\textit{mo:title}(m, t) \leftarrow \textit{title}(m, t, y)$	(M$_2$)
$\textit{mo:year}(m, y) \leftarrow \textit{title}(m, t, y)$	(M$_3$)

Unfolding with Semantic Query Optimisation

Query

$$q(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

Rewriting

$$q'(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

$\mathcal{M}$

$\textit{mo:Movie}(m) \leftarrow \textit{title}(m, t, y)$	(M_1)
$\textit{mo:title}(m, t) \leftarrow \textit{title}(m, t, y)$	(M_2)
$\textit{mo:year}(m, y) \leftarrow \textit{title}(m, t, y)$	(M_3)

Unfolding

$$q^*(t, y) \leftarrow \textit{title}(m, t_0, y_0), \textit{title}(m, t, y_1), \textit{title}(m, t_2, y), (y > 2010)$$

Unfolding with Semantic Query Optimisation

Query

$$q(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

Rewriting

$$q'(t, y) \leftarrow \textit{mo:Movie}(m), \textit{mo:title}(m, t), \textit{mo:year}(m, y), (y > 2010)$$

$\mathcal{M}$

$$\textit{mo:Movie}(m) \leftarrow \textit{title}(m, t, y) \quad (\text{M}_1)$$
$$\textit{mo:title}(m, t) \leftarrow \textit{title}(m, t, y) \quad (\text{M}_2)$$
$$\textit{mo:year}(m, y) \leftarrow \textit{title}(m, t, y) \quad (\text{M}_3)$$

Unfolding

$$q^*(t, y) \leftarrow \textit{title}(m, t_0, y_0), \textit{title}(m, t, y_1), \textit{title}(m, t_2, y), (y > 2010)$$

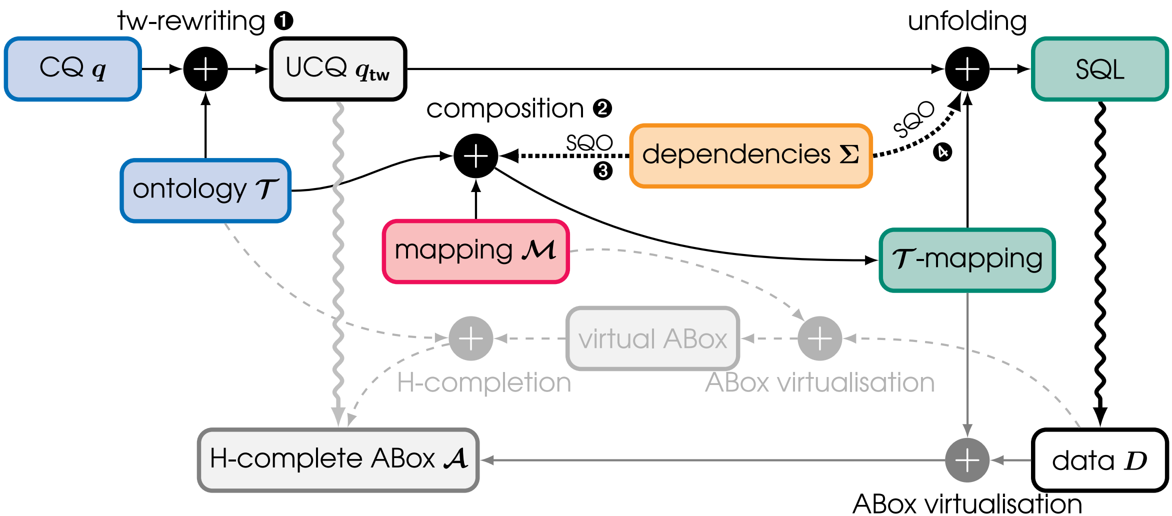
primary keys

$$\forall m \forall t_1 \forall t_2 \left(\exists y\, \textit{title}(m, t_1, y) \land \exists y\, \textit{title}(m, t_2, y) \to (t_1 = t_2)\right) \quad (\text{PK}_1)$$
$$\forall m \forall y_1 \forall y_2 \left(\exists t\, \textit{title}(m, t, y_1) \land \exists t\, \textit{title}(m, t, y_2) \to (y_1 = y_2)\right) \quad (\text{PK}_2)$$

Semantic Query Optimisation

$$q^\dagger(t, y) \leftarrow \textit{title}(m, t, y), (y > 2010)$$

Practical OBDA with Ontop



❶ tree-witness rewriting q_{tw} over H-complete ABoxes (no concept/role hierarchies)

❷ $\mathcal{T}$-mapping = system mapping $\mathcal{M} + \mathcal{T}$ makes virtual ABoxes H-complete

❸ $\mathcal{T}$-mapping is simplified using SQO and SQL features

constructed and optimised for $\mathcal{T}$ and Σ only once

❹ unfolding uses SQO to produce small and efficient SQL queries

References (1)

A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini and R. Rosati. Linking Data to Ontologies. J. Data Semantics 10: 133–173 (2008)

A. Artale, D. Calvanese, R. Kontchakov and M. Zakharyaschev: The DL-Lite Family and Relations. J. Artif. Intell. Res. (JAIR) 36:1–69 (2009)

A. Calì, G. Gottlob and A. Pieris. Query Rewriting under Non-Guarded Rules. In Proc. AMW 2010

M. Vardi. The Complexity of Relational Query Languages (Extended Abstract). In Proc. STOC 1982: 137–146

D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. ACM Trans. Database Syst., 4(4):455–469, 1979

D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. J. Comput. Syst. Sci., 28(1):167–189, 1984

A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. Proc. PODS 2008: 149–158

M. Rodriguez-Muro and D. Calvanese. Semantic Index: Scalable Query Answering without Forward Chaining or Exponential Rewritings Posters of ISWC 2011

M. Rodriguez-Muro and D. Calvanese. Dependencies: Making Ontology Based Data Access Work. Proc. AMW 2011

G. Gottlob and T. Schwentick. Rewriting Ontological Queries into Small Nonrecursive Datalog Programs. Proc. KR 2012

References (2)

S. Kikot, R. Kontchakov, V. Podolskii and M. Zakharyaschev: Exponential Lower Bounds and Separation for Query Rewriting. Proc. ICALP (2) 2012: 263–274

F. Baader, S. Brandt and C. Lutz. Pushing the EL envelope. Proc. IJCAI 2005

B. N. Grosof, I. Horrocks, R. Volz and S. Decker. Description logic programs: Combining logic programs with description logic. Proc. WWW 2003

A. Calì, G. Gottlob and A. Pieris. Advanced Processing for Ontological Queries. PVLDB 3(1): 554–565 (2010)

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. Proc. AAAI 2005: 602–607

A. Chortaras, D. Trivela and G. Stamou. Optimized query rewriting for OWL 2 QL. Proc. CADE 2011

T. Eiter, Ortiz, M. Šimkus, T.-K. Tran and G. Xiao. Query rewriting for Horn-SHIQ plus rules. Proc. AAAI 2012

M. König, M. Leclère, M.-L. Mugnier and M. Thomazo: A sound and complete backward chaining algorithm for existential rules. Proc. RR 2012

M. Rodríguez-Muro, R. Kontchakov and M. Zakharyaschev. Ontology-Based Data Access: Ontop of Databases. Proc. ISWC 2013

R. Kontchakov, M. Rezk, M. Rodríguez-Muro, G. Xiao and M. Zakharyaschev. Answering SPARQL Queries under the OWL 2 QL Entailment Regime with Databases. Proc. ISWC 2014