The Web Ontology Language

OWL 2 Ontology

Syntax layer

Semantics layer

Direct Semantics

RDF-Based Semantics

correspondence theorem (for DL subset)
Part 1

DL Basics
Description Logics by Example

SubClassOf(ObjectIntersectionOf(Person, ObjectSomeValuesFrom(takesCourse, Course)), Student)
  \[\text{Person} \cap \exists \text{takesCourse}.\text{Course} \sqsubseteq \text{Student}\]

SubObjectPropertyOf(mastersDegreeFrom, degreeFrom)
  \[\text{mastersDegreeFrom} \sqsubseteq \text{degreeFrom}\]

SubClassOf(ObjectSomeValuesFrom(ObjectInverseOf(takesCourse), owl:Thing), Course)
  \[\text{takesCourse}^\sim.\top \sqsubseteq \text{Course}\]

ClassAssertion(Student, john)
  \[\text{Student}(\text{john})\]

ObjectPropertyAssertion(takesCourse, john, sw)
  \[\text{takesCourse}(\text{john}, \text{sw})\]
**DL Syntax**

**concepts** (classes)

\[
C ::= A_i | \top | \bot | \neg C | C_1 \sqcap C_2 | C_1 \sqcup C_2 | \exists R.C | \forall R.C
\]

- \(A_i\): concept name
- \(\top\): owl:Thing
- \(\bot\): owl:Nothing
- \(\neg C\): ObjectComplementOf(\(C\))
- \(C_1 \sqcap C_2\): ObjectIntersectionOf(\(C_1, C_2\))
- \(C_1 \sqcup C_2\): ObjectUnionOf(\(C_1, C_2\))
- \(\exists R.C\): ObjectSomeValuesFrom(\(R, C\))
- \(\forall R.C\): ObjectAllValuesFrom(\(R, C\))

**roles** (object properties)

\[
R ::= P_i | P_i^-
\]

- \(P_i\): role name

**TBox** \(\mathcal{T}\)

\[
C_1 \sqsubseteq C_2 \quad \text{and} \quad R_1 \sqsubseteq R_2
\]

- \(\sqsubseteq\): SubClassOf(\(C_1, C_2\))
- \(\sqsubseteq\): SubObjectPropertyOf(\(R_1, R_2\))

**ABox** \(\mathcal{A}\)

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

**knowledge base** \(\mathcal{K} = (\mathcal{T}, \mathcal{A})\) (ontology)
**DL Semantics**

**interpretation** $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$

$\Delta^\mathcal{I}$

- individuals $a_i \rightarrow$ elements $a_i^\mathcal{I} \in \Delta^\mathcal{I}$
- concept names $A_i \rightarrow$ sets $A_i^\mathcal{I} \subseteq \Delta^\mathcal{I}$
- role names $P_i \rightarrow$ binary relations $P_i^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
DL Semantics (2)

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

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

\[(-C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\]

\[(C_1 \cap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \quad \text{and} \quad (C_1 \cup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}\]
DL Semantics (3)

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

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

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

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

$sp1 \in (\forall \text{takesCourse}. \bot)^I$
DL Semantics (4)

\[ I \models C_1 \subseteq C_2 \iff C^I_1 \subseteq C^I_2 \]

\[ I \models R_1 \subseteq R_2 \iff R^I_1 \subseteq R^I_2 \]

\[ I \models C(a) \iff a^I \in C^I \]

\[ I \models R(a, b) \iff (a^I, b^I) \in R^I \]

\[ I \text{ is a model of } (\mathcal{T}, \mathcal{A}) \text{ if } I \models \alpha, \text{ for all inclusions } \alpha \text{ in } \mathcal{T} \]

and assertions \( \alpha \) in \( \mathcal{A} \)
Open World Assumption

\( \mathcal{T} = \{ \) GraduateStudent \( \sqsubseteq \) Student \( \) GraduateStudent \( \sqsubseteq \) \( \exists \) takesCourse.GraduateCourse \( \} \)

\( \mathcal{A} = \{ \) GraduateStudent(john) \( \} \)

\( \Delta \mathcal{I}_1 \)

\begin{align*}
\text{GraduateStudent} & \quad \text{Student} \\
\text{GraduateCourse} & \quad \text{\( \ni \)} \\
\text{\( j \)} & \quad \text{\( \text{\( \ni \) takesCourse} \text{\( s \)} \)} \\
\text{john} & \quad \text{\( \ni \)} \\
\end{align*}

\( \text{john}^{\mathcal{I}_1} = j \)

\( \text{GraduateStudent}^{\mathcal{I}_1} = \{ j \} \)

\( \text{Student}^{\mathcal{I}_1} = \{ j \} \)

\( \text{GraduateCourse}^{\mathcal{I}_1} = \{ s \} \)

\( \text{takesCourse}^{\mathcal{I}_1} = \{(j, s)\} \)

is a model

\( \Delta \mathcal{I}_2 \)

\begin{align*}
\text{GraduateStudent} & \quad \text{Student} \\
\text{GraduateCourse} & \quad \text{\( \ni \)} \\
\text{\( a \)} & \quad \text{\( \text{\( \ni \) takesCourse} \text{\( a \)} \)} \\
\text{john} & \quad \text{\( \ni \)} \\
\end{align*}

\( \text{john}^{\mathcal{I}_2} = a \)

\( \text{GraduateStudent}^{\mathcal{I}_2} = \{ a \} \)

\( \text{Student}^{\mathcal{I}_2} = \{ a \} \)

\( \text{GraduateCourse}^{\mathcal{I}_2} = \{ a \} \)

\( \text{takesCourse}^{\mathcal{I}_2} = \{(a, a)\} \)

is a model

\( \Delta \mathcal{I}_3 \)

\begin{align*}
\text{GraduateStudent} & \quad \text{Student} \\
\text{GraduateCourse} & \quad \text{\( \ni \)} \\
\text{\( j \)} & \quad \text{\( \ni \)} \\
\text{john} & \quad \text{\( \ni \)} \\
\end{align*}

\( \text{john}^{\mathcal{I}_3} = j \)

\( \text{GraduateStudent}^{\mathcal{I}_3} = \{ j \} \)

\( \text{Student}^{\mathcal{I}_3} = \{ j \} \)

\( \text{GraduateCourse}^{\mathcal{I}_3} = \emptyset \)

\( \text{takesCourse}^{\mathcal{I}_3} = \emptyset \)

is not a model
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**

$\mathcal{T}$:

- UndergraduateStudent $\sqsubseteq \forall$ takesCourse. UndergraduateCourse
- UndergraduateCourse $\sqcap$ GraduateCourse $\sqsubseteq \bot$

$\mathcal{A}$:

- UndergraduateStudent(john)
- takesCourse(john, sw)
- GraduateCourse(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** \( \mathcal{K} \)  

\[ \mathcal{K} \models C_1 \sqsubseteq C_2 \]

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)

\( \mathcal{T} \): \

\[ \forall \text{takesCourse. UndergraduateCourse} \sqsubseteq \text{UndergraduateStudent} \]

\[ \text{FirstYearStudent} \sqsubseteq \exists \text{takesCourse. UndergraduateCourse}. \]

\( \mathcal{I}_1 \)

\[
\begin{array}{c}
\text{FirstYearStudent} \\
\text{UndergraduateStudent}
\end{array}
\]

\( \mathcal{I}_2 \)

\[
\begin{array}{c}
\text{FirstYearStudent} \\
\text{UndergraduateCourse}
\end{array}
\]

\( \mathcal{I}_1 \models \mathcal{T} \)

\( \mathcal{I}_1 \models \text{FirstYearStudent} \sqsubseteq \text{UndergraduateStudent} \)

\( \mathcal{I}_2 \models \mathcal{T} \)

\( \mathcal{I}_2 \models \text{FirstYearStudent} \not\sqsubseteq \text{UndergraduateStudent} \)
Reasoning: Entailment (2)

**Proposition** $(\mathcal{T}, \mathcal{A}) \models C_1 \sqsubseteq C_2$ iff $(\mathcal{T}, \mathcal{A} \cup \{C_1(a), \neg C_2(a)\})$ is not satisfiable for a fresh $a$

**Proof**

$(\Rightarrow, \text{only if})$ Let $(\mathcal{T}, \mathcal{A}) \models C_1 \sqsubseteq C_2$.

Assume, for the sake of contradiction, that $(\mathcal{T}, \mathcal{A} \cup \{C_1(a), \neg C_2(a)\})$ is satisfiable.

Then there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ (a model of $(\mathcal{T}, \mathcal{A})$), $a^\mathcal{I} \in C_1^\mathcal{I}$ and $a^\mathcal{I} \notin C_2^\mathcal{I}$, contrary to $(\mathcal{T}, \mathcal{A}) \models C_1 \sqsubseteq C_2$.

$(\Leftarrow, \text{if})$ Let $(\mathcal{T}, \mathcal{A} \cup \{C_1(a), \neg C_2(a)\})$ be not satisfiable.

Assume, for the sake of contradiction, that $(\mathcal{T}, \mathcal{A}) \not\models C_1 \sqsubseteq C_2$.

Then there is a model of $(\mathcal{T}, \mathcal{A})$ and some $u \in C_1^\mathcal{I}$ such that $u \notin C_2^\mathcal{I}$.

Since $a$ is fresh, we can redefine $a^\mathcal{I}$ by taking $a^\mathcal{I} = u$, which will contradict inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{C_1(a), \neg C_2(a)\})$.

**Proposition** $(\mathcal{T}, \mathcal{A}) \models C(a)$ iff $(\mathcal{T}, \mathcal{A} \cup \{\neg C(a)\})$ is not satisfiable.
Conjunctive Queries (CQs)

A conjunctive query is given by

\[
q(\vec{x}) = \exists \vec{y} \, \varphi(\vec{x}, \vec{y})
\]

where \(\varphi(\vec{x}, \vec{y})\) is a conjunction of atoms such as \(A(z)\) and \(P(z_1, z_2)\).

\(z, z_1\) and \(z_2\) are terms = an individual name or a variable from \(\vec{x}\) or \(\vec{y}\)

Example

\[q(x) = \exists y, z \ (\text{friend}(x, y) \land \text{Female}(y) \land \text{loves}(y, z) \land \text{Male}(z))\]

```
SELECT ?x
WHERE {
  ?y a :Female.
  ?z a :Male.
}
```
Certain Answers to CQs

\[ q(\vec{x}) = \exists \vec{y} \varphi(\vec{x}, \vec{y}) \] is a CQ with \( \vec{x} = (x_1, \ldots, x_n) \)

\( \vec{a} = (a_1, \ldots, a_n) \) 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 \( (\mathcal{T}, \mathcal{A}) \)

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

\( q \) without answer variables is Boolean

a certain answer to \( q \) over \( (\mathcal{T}, \mathcal{A}) \) is ‘yes’ if \( (\mathcal{T}, \mathcal{A}) \models q \) and ‘no’ otherwise
Andrea’s Example

\[ \mathcal{T}: \quad T \subseteq \text{Male} \sqcup \text{Female}, \quad \text{Male} \cap \text{Female} \subseteq \bot \]

\[ \mathcal{A}: \quad \text{friend}(\text{john, susan}), \quad \text{friend}(\text{john, andrea}), \quad \text{Female}(\text{susan}) \]
\[ \text{loves}(\text{susan, andrea}), \quad \text{loves}(\text{andrea, bill}), \quad \text{Male}(\text{bill}) \]

\[ q(x) = \exists y, z \left( \text{friend}(x, y) \land \text{Female}(y) \land \text{loves}(y, z) \land \text{Male}(z) \right) \]

\[ \text{NB: the same as finding instances of } \exists \text{friend.} (\text{Female} \cap \exists \text{loves. Male}) \]
From OWL to DL

**ObjectPropertyDomain** *(takesCourse, Student)*

\[ \exists \text{takesCourse}. \top \sqsubseteq \text{Student} \]

**ObjectPropertyRange** *(takesCourse, Course)*

\[ \exists \text{takesCourse}^-. \top \sqsubseteq \text{Course} \]

or

\[ \top \sqsubseteq \forall \text{takesCourse}. \text{Course} \]

**Proposition** The following are equivalent (have the same models)

\[ C_1 \sqsubseteq \forall P.C_2 \quad \text{and} \quad \exists P-.C_1 \sqsubseteq C_2 \]

\[ C_1 \sqcup C_2 \sqsubseteq C \quad \text{and} \quad \{ C_1 \sqsubseteq C, \ C_2 \sqsubseteq C \} \]

\[ C \sqsubseteq C_1 \sqcap C_2 \quad \text{and} \quad \{ C \sqsubseteq C_1, \ C \sqsubseteq C_2 \} \]

\[ C_1 \sqsubseteq \neg C_2 \quad \text{and} \quad C_1 \sqcap C_2 \sqsubseteq \bot \]
From OWL to DL (2)

**EquivalentClasses**\( (C_1, C_2, \ldots, C_n) \)

\[
C_1 \sqsubseteq C_2, \quad C_2 \sqsubseteq C_3, \quad \ldots, \quad C_{n-1} \sqsubseteq C_n, \quad C_n \sqsubseteq C_1
\]

**NB:** \( A \equiv B \) stands for ‘\( A \sqsubseteq B \) and \( B \sqsubseteq A \)’

**DisjointClasses**\( (C_1, C_2, \ldots, C_n) \)

\[
C_i \cap C_j \sqsubseteq \bot, \quad \text{for all } i, j \text{ with } 1 \leq i < j \leq n
\]

**DisjointUnion**\( (C, C_1, C_2, \ldots, C_n) \),

\[
C \equiv C_1 \sqcup \cdots \sqcup C_n
\]

**SymmetricObjectProperty**\( (P) \)

\[
P^- \sqsubseteq P
\]
**DL Zoo**

**ALCHI**

**AL** – attributive language
- **C** – complement \( \neg C \)
- **I** – role inverses \( P^- \)
- **H** – role inclusions \( R_1 \subseteq R_2 \)

**S** – **ALC** + transitive roles
- **N** – unqualified number restrictions \( \geq q \, R \cdot T \)
- **O** – nominals \{a\}
- **Q** – qualified number restrictions \( \geq q \, R \cdot C \)
- **F** – functionality constraints \( \geq 2 \, R \cdot T \subseteq \bot \)
- **R** – role chains and \( \exists R \cdot \text{Self} \)

**SHOIN** \( \approx \) OWL 1.0

**SHIF** \( \approx \) OWL Lite

**SROIQ** \( \approx \) OWL 2
Complexity of Reasoning

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

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

CQ entailment over $\text{ALCHI}$ KBs is $\text{2ExpTime}$-complete.

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

DL Complexity Navigator: [www.cs.man.ac.uk/~ezolin/dl](http://www.cs.man.ac.uk/~ezolin/dl)

Practical reasoners for OWL 2 DL: FaCT++, HermiT, Pellet
Part 2

OWL 2 Profiles
(In)tractable Reasoning

“'I can't find an efficient algorithm, I guess I'm just too dumb.'

Garey & Johnson, 1979
(In)tractable Reasoning

“I can’t find an efficient algorithm, because no such algorithm is possible!”

Garey & Johnson, 1979
(In)tractable Reasoning

“I can’t find an efficient algorithm, but neither can all these famous people.”

Garey & Johnson, 1979

*Computers and Intractability: A Guide to the Theory of NP-Completeness*
**OWL 2 Profiles: No Disjunction**

**graph 3-colourability** problem is **NP-complete**

given an (undirected) graph $G = (V, E)$,

decide whether each of its vertices can be painted in one of the three colours

in such a way that no pair of adjacent vertices has the same colour

$A_G$: $\text{edge}(v_1, v_2)$, for each $\{v_1, v_2\} \in E$

$T$: $\top \sqsubseteq C_1 \sqcup C_2 \sqcup C_3$

$\forall i \neq j \leq 3, \quad C_i \sqcap C_j \sqsubseteq \bot$

$\forall i \leq 3, \quad C_i \sqcap \exists \text{edge}.C_i \sqsubseteq \bot$

models of $(T, A_G) \sim 3$-colouring of $G$

$(T, A_G)$ is satisfiable $\iff G$ is 3-colourable

the satisfiability problem for KBs in any DL able to express this TBox

is **NP-hard** (intractable)
OWL 2 RL

Description Logic Programs (Grosof, Horrocks, Volz, Decker ’03) and pD* (ter Horst ’05) supported by Oracle Database 11g, OWLIM, BaseVISor, ELLY, Jena and RDFox

**RL** TBox:

\[ B \sqsubseteq A, \quad R_1 \sqsubseteq R_2 \quad \text{and} \quad B \sqsubseteq \bot \quad \text{only concept names on the right} \]

\[ B \ ::= \ A \mid \exists R. T \mid \exists R. B \mid B_1 \cap B_2 \]

**Simple** ABox:

\[ A(a) \text{ and } P(a, b) \]

**NB:** this is a simplified version — see the equivalences
UndergraduateStudent ⊑ Student
\[ \forall x \ (\text{UndergraduateStudent}(x) \rightarrow \text{Student}(x)) \]

\exists \text{takesCourse}. \text{UndergraduateCourse} \sqsubseteq \text{UndergraduateStudent}
\[ \forall x \forall y \ (\text{takesCourse}(x, y) \land \text{UndergraduateCourse}(y) \rightarrow \text{UndergraduateStudent}(x)) \]

**standard translation**
\[ ST_x(A) = A(x) \text{ and } ST_{x,y}(P) = P(x, y) \]

\[ ST_{x,y}(P^-) = P(y, x) \quad ST_x(B_1 \sqcap B_2) = ST_x(B_1) \land ST_x(B_2) \]
\[ ST_x(\exists R. \top) = \exists y \ ST_{x,y}(R) \quad ST_x(\exists R. B) = \exists y \ (ST_{x,y}(R) \land ST_y(B)) \]

\[ (B \sqsubseteq A)^* = \forall x \ (ST_x(B) \rightarrow ST_x(A)) \]
\[ (R_1 \sqsubseteq R_2)^* = \forall x \forall y \ (ST_{x,y}(R_1) \rightarrow ST_{x,y}(R_2)) \]
\[ (B \sqsubseteq \bot)^* = \forall x \ (ST_x(B) \rightarrow \bot) \]

\[ \forall \vec{y} \ (\gamma_1(\vec{y}) \land \cdots \land \gamma_k(\vec{y}) \rightarrow \gamma_0(\vec{y})) \]
\[ \forall \vec{y} \ (\gamma_1(\vec{y}) \land \cdots \land \gamma_k(\vec{y}) \rightarrow \bot) \]

Horn clauses (datalog programs)
Chase: Example

\[ \mathcal{T}: \exists \text{takesCourse. UndergraduateCourse } \sqsubseteq \text{UndergraduateStudent} \]
\[ \text{UndergraduateStudent } \sqsubseteq \text{Student} \]

\[ \mathcal{A}: \text{takesCourse}(\text{john, sp1}), \text{ UndergraduateCourse}(\text{sp1}) \]

step 0: \[ \mathcal{I}_0 \approx \mathcal{A} \]
\[ \Delta^{\mathcal{I}_0} = \{\text{john, sp1}\} \]
\[ \text{takesCourse}^{\mathcal{I}_0} = \{(\text{john, sp1})\} \]
\[ \text{UndergraduateCourse}^{\mathcal{I}_0} = \{\text{sp1}\} \]
\[ \text{UndergraduateStudent}^{\mathcal{I}_0} = \emptyset \]
\[ \text{Student}^{\mathcal{I}_0} = \emptyset \]
\[ \mathcal{I}_0 \not\models \mathcal{T} \]

step 1: \[ \Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_0} \] and \[ \mathcal{I}_1 \] expands \[ \mathcal{I}_0 \] with
\[ \text{UndergraduateStudent}^{\mathcal{I}_1} = \{\text{john}\} \]
\[ \mathcal{I}_1 \not\models \mathcal{T} \]

step 2: \[ \Delta^{\mathcal{I}_2} = \Delta^{\mathcal{I}_1} \] and \[ \mathcal{I}_1 \] expands \[ \mathcal{I}_1 \] with
\[ \text{Student}^{\mathcal{I}_2} = \{\text{john}\} \]
\[ \mathcal{I}_2 \models \mathcal{T} \]

stop: \[ \mathcal{I}_2 \] is a model of \( (\mathcal{T}, \mathcal{A}) \)

RW 2014, Athens, 10.09.14
Chase for RL

the standard model $\mathcal{I}_A$ of a simple ABox $A$ is

- $\Delta^{\mathcal{I}_A} = \text{ind}(A)$
- $a^{\mathcal{I}_A} = a$, for $a \in \text{ind}(A)$
- $A^{\mathcal{I}_A} = \{a \mid A(a) \in \mathcal{A}\}$, for concept name $A$
- $P^{\mathcal{I}_A} = \{(a, b) \mid P(a, b) \in \mathcal{A}\}$, for role name $P$

chase rules:

(c) if $a \in B^{\mathcal{I}_k}$, $B \sqsubseteq A \in \mathcal{T}$ but $a \notin A^{\mathcal{I}_k}$, then add $a$ to $A^{\mathcal{I}_{k+1}}$

(r) if $(a, b) \in R_1^{\mathcal{I}_k}$, $R_1 \sqsubseteq R_2 \in \mathcal{T}$ but $(a, b) \notin R_2^{\mathcal{I}_k}$, then add $(a, b)$ to $R_2^{\mathcal{I}_{k+1}}$

(b) if $a \in B^{\mathcal{I}_k}$ and $B \sqsubseteq \bot \in \mathcal{T}$, then the process terminates

the domains of the $\mathcal{I}_k$ are finite and all coincide with $\text{ind}(\mathcal{A})$,

$\Rightarrow$ the process terminates after a finite number of steps

- if (b) is not applicable then the result is a model of $(\mathcal{T}, A)$, the canonical model $C_{\mathcal{T}, A}$ of $(\mathcal{T}, A)$

- otherwise, $(\mathcal{T}, A)$ is inconsistent
Canonical Model is Universal

A homomorphism $h$ from an interpretation $\mathcal{I}_1$ to an interpretation $\mathcal{I}_2$ is a map from $\Delta^{\mathcal{I}_1}$ to $\Delta^{\mathcal{I}_2}$ such that

- $h(a^{\mathcal{I}_1}) = a^{\mathcal{I}_2}$, for each individual name $a$
- $h(u) \in A^{\mathcal{I}_2}$, for any $u \in A^{\mathcal{I}_1}$ and any concept name $A$
- $(h(u), h(v)) \in P^{\mathcal{I}_2}$, for any $(u, v) \in P^{\mathcal{I}_1}$ and any role name $P$

Lemma $\mathcal{C}_{\mathcal{T}, \mathcal{A}}$ is universal

$\mathcal{C}_{\mathcal{T}, \mathcal{A}}$ can be homomorphically mapped into any other model of $(\mathcal{T}, \mathcal{A})$
Complexity of Reasoning in RL

Theorem Let \((\mathcal{T}, \mathcal{A})\) be a consistent RL KB. Then \(\mathcal{C}_{\mathcal{T}, \mathcal{A}} \models (\mathcal{T}, \mathcal{A})\). In addition,

\[(\mathcal{T}, \mathcal{A}) \vdash B \subseteq A \iff \mathcal{C}_{\mathcal{T}, \mathcal{A}} \models B \subseteq A\]

\[(\mathcal{T}, \mathcal{A}) \vdash R_1 \subseteq R_2 \iff \mathcal{C}_{\mathcal{T}, \mathcal{A}} \models R_1 \subseteq R_2\]

\[(\mathcal{T}, \mathcal{A}) \vdash q(\vec{a}) \iff \mathcal{C}_{\mathcal{T}, \mathcal{A}} \models q(\vec{a})\]

Theorem The problems of KB consistency, concept and role subsumption and instance checking are \(\text{P}\)-complete in RL.

The problem of CQ entailment in RL is \(\text{NP}\)-complete

Proof Construct the canonical model (in a polynomial number of steps).
Check the condition in the canonical model.
SNOMED CT

Pericardium ⊑ Tissue ∩ containedIn.Heart
Pericarditis ⊑ Inflammation ∩ hasLocation.Pericardium
Inflammation ⊑ Disease ∩ actsOn.Tissue
Disease ∩ hasLocation.containedIn.Heart ⊑ HeartDisease ∩ NeedsTreatment

EL TBox:

\[
C_1 \sqsubseteq C_2 \quad \text{and} \quad P_1 \sqsubseteq P_2
\]

\[
C ::= A \mid \exists P. \top \mid \exists P.C \mid C_1 \sqcap C_2
\]

no inverses

simple ABox:

\[
A(a) \quad \text{and} \quad P(a, b)
\]
EL and Existential Rules

\[ \text{Person} \sqsubseteq \exists \text{parent}.\text{Person} \]

\[ \forall x (\text{Person}(x) \rightarrow \exists y (\text{parent}(x, y) \land \text{Person}(y))) \]

the standard translation can be extended to EL

\[ \forall \bar{y} (\gamma_1(\bar{y}) \land \cdots \land \gamma_k(\bar{y}) \rightarrow \exists \bar{x} \gamma_0(\bar{x}, \bar{y})) \]

existential rules

chase rules: (Johnson & Klug ‘84)

\textbf{(c')} if \( d \in C^k \) and \( C \sqsubseteq A \in \mathcal{T} \), then add \( d \) to \( A^{k+1} \)

\textbf{(r')} if \( (d, d') \in P_1^k \) and \( P_1 \sqsubseteq P_2 \in \mathcal{T} \), then we add \( (d, d') \) to \( P_2^{k+1} \)

\textbf{(e)} if \( d \in C^k \) and \( C \sqsubseteq \exists P.D \in \mathcal{T} \), where \( D \) is a concept name or \( \top \),

then take a fresh labelled null, \( d' \), and add \( d' \) to \( D^{k+1} \) and \( (d, d') \) to \( P^{k+1} \)

\textbf{NB:} oblivious chase — no check whether there is such an element
**EL Chase: Example**

\( \mathcal{T} \): Person ⊑ ∃parent. Person

\( \mathcal{A} \): Person(john)

**step 0:** \( \Delta^{I_0} = \{ \text{john} \} \), \( \text{Person}^{I_0} = \{ \text{john} \} \), \( \text{parent}^{I_0} = \emptyset \)

**step 1:** \( \Delta^{I_1} = \{ \text{john}, d_1 \} \), \( \text{Person}^{I_1} = \{ \text{john}, d_1 \} \), \( \text{parent}^{I_1} = \{ (\text{john}, d_1) \} \)

**step 2:** \( \Delta^{I_2} = \{ \text{john}, d_1, d_2 \} \), \( \text{Person}^{I_2} = \{ \text{john}, d_1, d_2 \} \), \( \text{parent}^{I_2} = \{ (\text{john}, d_1), (d_1, d_2) \} \)

1. this canonical model is infinite
2. there is no finite universal model
3. there are many universal models (each is infinite)
Normal Form in EL

normal form of concept inclusions

\[ A_1 \cap A_2 \sqsubseteq A, \quad \exists P.D \sqsubseteq A \quad \text{or} \quad A \sqsubseteq \exists P.D \]

\( P \) is a role name, \( A, A_1 \) and \( A_2 \) are concept names and \( D \) is either a concept name or \( T \)

**Example:** \( \exists P.A \cap B \sqsubseteq \exists R.\exists P.A \)

introduce abbreviations: \( C \) is \( \exists P.A \) and \( D \) is \( \exists R.C \)

result:

\( C \cap B \sqsubseteq D, \quad \exists P.A \sqsubseteq C, \quad C \sqsubseteq \exists P.A, \quad D \sqsubseteq \exists R.C, \quad \exists R.C \sqsubseteq D \)
Reasoning in EL

Do not construct the infinite chase, re-use the labelled nulls

for each positive $\exists S.D$, take a fixed witness $w_{\exists S.D}$

(e-c) if $d \in C^{\mathcal{I}_k}$ and $C \sqsubseteq \exists P.D \in \mathcal{T}$, where $D$ is a concept name or $\top$,

then add $w_{\exists S.D}$ to $D^{\mathcal{I}_{k+1}}$ and $(d, d')$ to $P^{\mathcal{I}_{k+1}}$

the generating model $\mathcal{G}_{\mathcal{T}, \mathcal{A}}$ of $(\mathcal{T}, \mathcal{A})$

NB: alternatively, take the canonical model and identify all labelled nulls introduced for the same $\exists S.D$
**Reasoning in EL (2)**

**Theorem** Let \((\mathcal{T}, \mathcal{A})\) be an EL KB. Then \(\mathcal{G}_{\mathcal{T}, \mathcal{A}} \models (\mathcal{T}, \mathcal{A})\). In addition,

\[
\begin{align*}
(\mathcal{T}, \mathcal{A}) \models C_1 \sqsubseteq C_2 & \iff \mathcal{G}_{\mathcal{T}, \mathcal{A}} \models C_1 \sqsubseteq C_2 \\
(\mathcal{T}, \mathcal{A}) \models P_1 \sqsubseteq P_2 & \iff \mathcal{G}_{\mathcal{T}, \mathcal{A}} \models P_1 \sqsubseteq P_2 \\
(\mathcal{T}, \mathcal{A}) \models C(a) & \iff \mathcal{G}_{\mathcal{T}, \mathcal{A}} \models C(a) \\
(\mathcal{T}, \mathcal{A}) \models P(a, b) & \iff \mathcal{G}_{\mathcal{T}, \mathcal{A}} \models P(a, b)
\end{align*}
\]

**Theorem** The problems of concept and role subsumption and instance checking are \(\text{P}-\text{complete}\) in EL.

**NB:** \(\mathcal{G}_{\mathcal{T}, \mathcal{A}}\) does not give correct answers to queries

\[
q = \exists x \text{ parent}(x, x)
\]
OWL 2 QL

(Calvanese et al. ‘05)

\[
\exists \text{worksFor}. \top \sqsubseteq \text{Employee} \\
\exists \text{worksFor}^- \cdot \top \sqsubseteq \text{Project} \\
\text{Manager} \sqsubseteq \text{Employee} \\
\text{Project} \sqsubseteq \exists \text{worksFor}^- \cdot \top
\]

QL TBox:

\[
B \sqsubseteq C \quad \text{and} \quad R_1 \sqsubseteq R_2
\]

\[
B ::= A \mid \exists R. \top \\
C ::= A \mid \exists R. \top \mid \exists R.C
\]

no \exists R.C on the left

simple ABox:

\[A(a) \text{ and } P(a, b)\]
Chase for QL

normal form of concept inclusions

\[
A' \sqsubseteq A, \quad \exists R \sqsubseteq A \quad \text{or} \quad A \sqsubseteq \exists R.D
\]

\(R\) is a role, \(A\) and \(A'\) are concept names and \(D\) is either a concept name or \(\top\)

chase rules:

(c') if \(d \in B^T_k\) and \(B \sqsubseteq A \in T\), then add \(d\) to \(A^{T_{k+1}}\)

(r') if \((d, d') \in R^T_k\) and \(R_1 \sqsubseteq R_2 \in T\), then add \((d, d')\) to \(R_2^{T_{k+1}}\)

(e) if \(d \in B^T_k\) and \(B \sqsubseteq \exists R.D \in T\), where \(D\) is a concept name or \(\top\),

then take a fresh labelled null, \(d'\), and add \(d'\) to \(D^{T_{k+1}}\) and \((d, d')\) to \(R^{T_{k+1}}\)
Unravelling of the Generating Structure

for each $\exists R.D$ that occurs positively in $\mathcal{T}$, take a witness $w_{\exists R.D}$

generating relation $\leadsto_{\mathcal{T},A}$:

\[
\begin{align*}
    a \leadsto_{\mathcal{T},A} w_{\exists R.D} & \quad \text{if} \quad a \in \text{ind}(A), \mathcal{I}_A \models B(a) \text{ and } \mathcal{T} \models B \subseteq \exists R.D \\
w_{\exists S.B} \leadsto_{\mathcal{T},A} w_{\exists R.D} & \quad \text{if} \quad \mathcal{T} \models \exists S^- \subseteq \exists R.D \text{ or } \mathcal{T} \models B \subseteq \exists R.D
\end{align*}
\]

$\leadsto_{\mathcal{T},A}$-path $\sigma$ is any $aw_{\exists R_1.D_1} \cdots w_{\exists R_n.D_n}$

- $a \in \text{ind}(A)$ and,
- if $n > 0$ then $a \leadsto_{\mathcal{T},A} w_{\exists R_1.D_1}$ and $w_{\exists R_i.D_i} \leadsto_{\mathcal{T},A} w_{\exists R_{i+1}.D_{i+1}}$, for $i < n$
Canonical Model: Example

\[ \mathcal{T}: \quad \text{RA} \sqsubseteq \exists \text{worksOn.} \text{Project} \quad \text{worksOn}^{-1} \sqsubseteq \exists \text{involves} \\
\text{Project} \sqsubseteq \exists \text{isManagedBy.} \text{Prof} \quad \text{isManagedBy} \sqsubseteq \exists \text{involves} \]

\[ \mathcal{A}: \quad \text{RA(chris), worksOn(chris, dyn), Project(dyn), Lecturer(dave), worksOn(dave, dyn)} \]

\[ w_1 = w_{\exists \text{worksOn.} \text{Project}} \quad \text{and} \quad w_2 = w_{\exists \text{isManagedBy.} \text{Prof}} \]

**generating relation**

\[ \text{chris} \sim_{\mathcal{T}, \mathcal{A}} w_1, \quad \text{dyn} \sim_{\mathcal{T}, \mathcal{A}} w_2, \quad w_1 \sim_{\mathcal{T}, \mathcal{A}} w_2 \]

**\( \sim_{\mathcal{T}, \mathcal{A}} \)-paths**

\[ \text{chris, chris} w_1, \quad \text{chris} w_1 w_2, \quad \text{dyn, dyn} w_2 \quad \text{and} \quad \text{dave} \]
Canonical Model: Example (2)

$\mathcal{T}$: 
\[
\begin{align*}
\text{RA} & \subseteq \exists \text{worksOn}. \text{Project} & \text{worksOn}^{-} & \subseteq \text{involves} \\
\text{Project} & \subseteq \exists \text{isManagedBy}. \text{Prof} & \text{isManagedBy} & \subseteq \text{involves}
\end{align*}
\]

$\mathcal{A}$: 
\[
\begin{align*}
\text{RA}(\text{chris}), & \ \text{worksOn}(\text{chris}, \text{dyn}), \ \text{Project}(\text{dyn}), \ \text{Lecturer}(\text{dave}), \\
& \ \text{worksOn}(\text{dave}, \text{dyn})
\end{align*}
\]

\[w_1 = w_{\exists \text{worksOn}. \text{Project}}\]
\[w_2 = w_{\exists \text{isManagedBy}. \text{Prof}}\]
## Comparing Profiles

<table>
<thead>
<tr>
<th></th>
<th>RL</th>
<th>EL</th>
<th>QL</th>
</tr>
</thead>
<tbody>
<tr>
<td>inverse roles $P^-$</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>$\exists R.C$ on the right</td>
<td>−</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>$\exists R.C$ on the left</td>
<td>+</td>
<td>+</td>
<td>−</td>
</tr>
</tbody>
</table>
Part 3

OBDA and Query Rewriting
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
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:
- Employee = \{john, mary, nick\}
- Manager = \{john, nick\}
- Project = \{A, B\}
- worksFor = \{(john, A), (mary, B)\}

query: \ \ \ q(x) \leftarrow \text{Employee}(x)

answer: \{john, mary, nick\}

\textbf{NB}: not having nick in Employee would violate the integrity constraint
\[ \forall x \ (\text{Manager}(x) \rightarrow \text{Employee}(x)) \]
Why do Databases Work?

**query answering problem** (as a recognition problem):

\[
\text{given a finite data } \mathcal{D}, \text{ a query } q(\vec{x}) \text{ and a tuple } \vec{a}, \\
\text{decide whether } \mathcal{I}_\mathcal{D} \models q(\vec{a}) \\
\text{ } \mathcal{I}_\mathcal{D} \text{ makes the facts in } \mathcal{D} \text{ true (and only them)}
\]

what is the complexity of CQ answering?

naive algorithm:

- guess values for all existential variables and then
- evaluate the query in polynomial time \(\text{ in NP}\)

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?’

\[
\begin{align*}
\mathcal{D} &= \{A(r, g), A(g, b), A(b, r), A(g, r), A(r, b), A(b, g)\} \\
q_G &= \exists v_1, \ldots, v_n \bigwedge_{(v_i, v_j) \in E} A(v_i, v_j)
\end{align*}
\]

in fact, the query answering algorithm runs in \( O(|\mathcal{D}|^{\|q\|}) \)

\text{data is large, query is short}

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

(Vardi, 1982): query answering is in \( \text{AC}^0 \) for data complexity
Circuits and $\text{AC}^0$

A circuit is an acyclic graph of AND-, OR- and NOT-gates (with $n$ inputs and a single output, sink).

A database instance $\mathcal{D}$ can be encoded on inputs (one input for each possible ground atom).

An $\text{FO}$-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.

$\text{AC}^0 = \text{circuits of constant depth with AND- and OR-nodes of unbounded fan-in}$

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: $\text{AC}^0$ is a proper subclass of $\text{LOGSPACE} \subseteq \text{P}$ (Parity does not belong to $\text{AC}^0$)

given a word $w$, decide whether its length is even
Query Rewriting Approach

use off-the-shelf RDBMS

(Calvanese et al. ’05)

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

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

**FO-rewritability**: only possible in DL with query answering in \( \text{FO} (=\text{AC}^0) \)

for data complexity:

**OWL 2 QL**
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 $G$’

\[
ABox: \quad A_{G,t} = \{ \text{edge}(v_1, v_2) \mid (v_1, v_2) \in E \} \cup \{ \text{ReachableFromTarget}(t) \}
\]

\[
TBox: \quad T = \{ \exists \text{edge}. \text{ReachableFromTarget} \sqsubseteq \text{ReachableFromTarget} \}
\]

\[
CQ: \quad q \leftarrow \text{ReachableFromTarget}(s)
\]

$T$ and $q$ do not depend on $G$, $s$, $t$

‘$(T, A_{G,t}) \models q$’ is NLogSpace-hard for data complexity

$q$ and $T$ are not FO-rewritable
Practical Query Answering in OWL 2 QL

systems
- QuOnto (Rome, 2005)
- REQUIEM (Oxford, 2009)
- Presto (Rome, 2010)
- IQAROS (Athens, 2011)
- Nyaya (Milan-Oxford, 2010) for TGDs
- Clipper (Vienna, 2012) for Horn-SHIQ
- (Montpellier, 2013) for TGDs
- Quest/ontop (Bolzano, 2011)

not so smoothly: the size of implemented rewritings $q'$ is $O((|q| \cdot |T|)^{|q|})$
(can’t say ‘query is small or fixed’ any longer)
Does a Rewriting Have to be Exponential?

TBox: mother ⊑ parent and father ⊑ parent

Query: \( \text{grandparent}(x, z) \leftarrow \text{parent}(x, y) \land \text{parent}(y, z) \)

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

\[ \text{grandparent}(x, z) \leftarrow \text{parent}(x, y) \land \text{parent}(y, z) \]
\[ \text{grandparent}(x, z) \leftarrow \text{father}(x, y) \land \text{father}(y, z) \]
\[ \text{grandparent}(x, z) \leftarrow \text{mother}(x, y) \land \text{father}(y, z) \]
\[ \ldots \]

PE-rewritings (positive existential queries \( \approx \) select-project-join-union):

\[ \text{grandparent}(x, z) \leftarrow (\text{parent}(x, y) \lor \text{father}(x, y) \lor \text{mother}(x, y)) \land \]
\[ (\text{parent}(y, z) \lor \text{father}(y, z) \lor \text{mother}(y, z)) \]

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

\[ \text{grandparent}(x, z) \leftarrow \text{ext-parent}(x, y) \land \text{ext-parent}(y, z) \]
\[ \text{ext-parent}(x, y) \leftarrow \text{parent}(x, y) \]
\[ \text{ext-parent}(x, y) \leftarrow \text{father}(x, y) \]
\[ \text{ext-parent}(x, y) \leftarrow \text{mother}(x, y) \]

FO-rewriting (first-order queries \( \approx \) SQL):

\( \exists \forall \land \neg \)
Case 1: Flat QL TBoxes

A TBox $\mathcal{T}$ is **flat** if it does not contain generating axioms $B' \sqsubseteq \exists R.B \approx$ RDF Schema ≈ RDF Schema

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

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

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

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 $O(|q| \cdot |\mathcal{T}|)$

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:

\[ q(x) \leftarrow \text{worksOn}(x, y) \land \text{involves}(y, z) \land \text{Professor}(z) \]

\[ \text{worksOn}(z, y) \lor \text{isManagedBy}(y, z) \lor \text{involves}(y, z) \]
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 $\mathcal{T} \models R \sqsubseteq P$

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

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

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

(thus we ignore the axioms considered in the flat rewriting)
Case 2: Who Works with Professors (2)?

**T**: RA $\sqsubseteq \exists$ worksOn. Project

Project $\sqsubseteq \exists$ isManagedBy. Prof

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

isManagedBy $\sqsubseteq$ involves

**A**: RA(chris), worksOn(chris, dyn), Project(dyn), Lecturer(dave), worksOn(dave, dyn)

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

$w_2 = w_{\exists \text{isManagedBy}.\text{Prof}}$
Case 2: Rewriting the Labelled Nulls

\[ q(x) \]

\[ \text{RA}(x) \land \text{Professor}(z) \land (x = z) \]

**PE-rewriting (over H-complete ABoxes):**

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

(x and z are the roots of the tree witness)
Tree-Witness Rewriting

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

$$q_{tw}(\vec{x}) = \bigvee_{\Theta} \exists \vec{y} \left( \bigwedge_{S(\vec{z}) \in q \setminus q_{\Theta}} S(\vec{z}) \land \bigwedge_{t \in \Theta} tw_t \right)$$

$\Theta$ independent set of tree witnesses
Rewritings as Boolean functions

hypergraph $H$:
vertices = atoms of the query
hyperedges = tree witnesses

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

$$f_H = \bigvee_{X \subseteq E} \left( \bigwedge_{v \in V \setminus V_X} p_v \wedge \bigwedge_{e \in X} p_e \right)$$

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 \( q_{t_1} \cap q_{t_2} = \emptyset \) or \( q_{t_1} \subseteq q_{t_2} \) or \( q_{t_2} \subseteq q_{t_1} \), for each pair \( t_1 \) and \( t_2 \), then compatible

\[
q'_{tw}(\vec{x}) = \bigwedge_{S(\vec{z}) \in q} \left( S(\vec{z}) \lor \bigvee_{t : S(\vec{z}) \in q_t} tw_t \right)
\]

is a rewriting (over H-complete ABoxes)

\textbf{EL:} add (recursive) datalog rules that ‘complete’ ABox

\[ \rightarrow \text{polynomial datalog rewriting} \] (tree witnesses in \( \mathcal{EL} \) are compatible)

\textbf{QL:} replace \( S(\vec{z}) \) with \[ \bigvee_{\tau \models S' \subseteq S} S'(\vec{z}) \]

\[ \rightarrow \text{polynomial positive existential rewriting} \]

provided that the number of tree witnesses is \text{polynomial} and they are \text{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, SPARQL end-point,
  OWLAPI and Sesame libraries
Why SQL rewritings are large:

(1) a large number of tree witnesses

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

(3) many inclusions in $\mathcal{T}$ follow from $\Sigma$ and $\mathcal{M}$
**Ontop Example**

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

- **database**

<table>
<thead>
<tr>
<th>movie ID</th>
<th>title</th>
<th>production year</th>
</tr>
</thead>
<tbody>
<tr>
<td>728</td>
<td>‘Django Unchained’</td>
<td>2012</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **dependencies**

\[
\forall m \left( \exists p, r \text{ castinfo}(p, m, r) \rightarrow \exists t, y \text{ title}(m, t, y) \right) \] (FK)
\[
\forall m \forall t_1 \forall t_2 \left( \exists y \text{ title}(m, t_1, y) \land \exists y \text{ title}(m, t_2, y) \rightarrow (t_1 = t_2) \right) \] (PK₁)
\[
\forall m \forall y_1 \forall y_2 \left( \exists t \text{ title}(m, t, y_1) \land \exists t \text{ title}(m, t, y_2) \rightarrow (y_1 = y_2) \right) \] (PK₂)

**Movie Ontology MO** [http://www.movieontology.org](http://www.movieontology.org)

\[
\text{mo:Movie} \equiv \exists \text{mo:title}, \quad \text{mo:Movie} \sqsubseteq \exists \text{mo:year},
\]
\[
\text{mo:Movie} \equiv \exists \text{mo:cast}, \quad \exists \text{mo:cast}^{-} \sqsubseteq \text{mo:Person}, \ldots
\]

**Mapping** (created by the Ontop development team)

\[
\text{mo:Movie}(m), \text{mo:title}(m, t), \text{mo:year}(m, y) \leftarrow \text{title}(m, t, y) \] (M₁)
\[
\text{mo:cast}(m, p), \text{mo:Person}(p) \leftarrow \text{castinfo}(p, m, r) \] (M₂)
Ontop: T-mappings

\[ \text{flat TBox } \mathcal{T} \]

forward chaining

\[ \text{H-complete ABox } \mathcal{A}' \]

virtualisation

composition

virtualisation

\[ \text{data } D \]

mapping \( \mathcal{M} \)

\[ \mathcal{T} \]

\[ \text{mo:Movie } \equiv \exists \text{mo:title}, \]
\[ \text{mo:Movie } \sqsubseteq \exists \text{mo:year}, \]
\[ \text{mo:Movie } \equiv \exists \text{mo:cast}, \]
\[ \exists \text{mo:cast} \sqsubseteq \text{mo:Person} \]

\[ \mathcal{M} \]

\[ \text{mo:Movie}(m), \text{mo:title}(m, t), \text{mo:year}(m, y) \leftarrow \text{title}(m, t, y) \quad (M_1) \]
\[ \text{mo:cast}(m, p), \text{mo:Person}(p) \leftarrow \text{castinfo}(p, m, r) \quad (M_2) \]

\[ \mathcal{M}^\mathcal{T} \]

\[ \text{mo:Movie}(m) \leftarrow \text{title}(m, t, y) \quad \text{by } (M_1) \]
\[ \text{mo:Movie}(m) \leftarrow \text{castinfo}(p, m, r) \quad \text{by } (M_2) + \exists \text{mo:cast} \sqsubseteq \text{mo:Movie} \]

redundant by (FK)

\[ \forall m \left( \exists p, r \text{ castinfo}(p, m, r) \rightarrow \exists t, y \text{ title}(m, t, y) \right) \]

RW 2014, Athens, 10.09.14
Optimising T-mappings

- using foreign keys (inclusion dependencies)

- using disjunction

\[ T \]

\[
\begin{align*}
    \text{mo:Actor} & \sqsubseteq \text{mo:Artist}, & \text{mo:Artist} & \sqsubseteq \text{mo:Person}, \\
    \text{mo:Director} & \sqsubseteq \text{mo:Person}, & \text{mo:Editor} & \sqsubseteq \text{mo:Person}, \ldots
\end{align*}
\]

\[ M \]

\[
\begin{align*}
    \text{mo:Actor}(p) & \leftarrow \text{castinfo}(p, m, r), (r = 1) \quad (M_1) \\
    \ldots
\end{align*}
\]

\[
\begin{align*}
    \text{mo:Editor}(p) & \leftarrow \text{castinfo}(p, m, r), (r = 6) \quad (M_6)
\end{align*}
\]

\[ M^T \]

\[
\begin{align*}
    \text{mo:Person}(p) & \leftarrow \text{castinfo}(p, m, r), ((r = 1) \lor \cdots \lor (r = 6))
\end{align*}
\]
Unfolding with Semantic Query Optimisation

Query

\[ q(t, y) \leftarrow \text{mo:Movie}(m), \text{mo:title}(m, t), \text{mo:year}(m, y), (y > 2010) \]

Rewriting

\[ q'(t, y) \leftarrow \text{mo:Movie}(m), \text{mo:title}(m, t), \text{mo:year}(m, y), (y > 2010) \]

\[ M \]

\[ \text{mo:Movie}(m) \leftarrow \text{title}(m, t, y) \] (M₁)
\[ \text{mo:title}(m, t) \leftarrow \text{title}(m, t, y) \] (M₂)
\[ \text{mo:year}(m, y) \leftarrow \text{title}(m, t, y) \] (M₃)

Unfolding

\[ q^*(t, y) \leftarrow \text{title}(m, t₀, y₀), \text{title}(m, t, y₁), \text{title}(m, t₂, y), (y > 2010) \]

primary keys

\[ \forall m \forall t₁ \forall t₂ (\exists y \text{title}(m, t₁, y) \land \exists y \text{title}(m, t₂, y) \rightarrow (t₁ = t₂)) \] (PK₁)
\[ \forall m \forall y₁ \forall y₂ (\exists t \text{title}(m, t, y₁) \land \exists t \text{title}(m, t, y₂) \rightarrow (y₁ = y₂)) \] (PK₂)

Semantic Query Optimisation

\[ q^+(t, y) \leftarrow \text{title}(m, t, y), (y > 2010) \]
Practical OBDA with Ontop

1. **tree-witness rewriting** $q_{tw}$ over **H-complete ABoxes** (no concept/role hierarchies)
2. $\mathcal{T}$-mapping $= \text{system mapping } \mathcal{M} + \mathcal{T}$ makes virtual ABoxes H-complete
3. $\mathcal{T}$-mapping is simplified using **SQO** and **SQL** features constructed and optimised for $\mathcal{T}$ and $\Sigma$ only once
4. unfolding uses **SQO** to produce small and efficient SQL queries
Recommended Reading (1)


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


Recommended Reading (2)


