Symbolic Artificial Intelligence

(SD213)

Query answering with description logic ontologies

Camille Bourgaux

23-05-2018

Which European citizens have been married to Zsa Zsa Gabor?

`https://tinyurl.com/ybp9ljyd`
`https://www.wikidata.org/wiki/Q207405`
`https://tinyurl.com/y9c9f8mo`

Which family members of the president of America were born outside of America?

In which Asian restaurants can you eat vegetarian food in Paris?

What are the inhibitors of enzymes produced by genes on the Y chromosome?

In this course

- Knowledge representation and reasoning with description logics
- Focus on query answering
- A few words on inconsistency handling

1. Reminders on ontologies and Description Logics

Ontologies

Ontologies are logical theories that formalize domain-specific knowledge, thereby making it available for machine processing.

Ontologies

Ontologies are logical theories that formalize domain-specific knowledge, thereby making it available for machine processing.

Ontologies define the terminology (vocabulary) of the domain and the semantics relationships between terms.

Example (family domain)

- Terms: parent, mother, sister, sibling, ...
- Relationships between terms: "mother" is a subclass of "parent", "sister" is both in the domain and in the range of "has sibling", "parent" is the disjoint union of "father" and "mother"...

Ontologies

Reasons for using ontologies

- Standardize the terminology of an application domain (easy to share information)
  - complex industrial systems description, scientific knowledge (medicine, life science...)
- Support automated reasoning (logical inferences)
  - expert systems, semantic web, ontology-based data access
- Present an intuitive and unified view of data sources
  - ontology-based data access, semantic web

Description logics: a popular way of specifying ontologies

Description logics are a family of fragments of first-order logic

- Wide variety of languages
- Trade-off between expressivity and complexity of reasoning

Syntax

Basic buillding blocks

- atomic concepts (unary predicates)
  - Mother, Sister ...
- atomic roles (binary predicates)
  - hasChild, isMarriedTo ...
- individuals (constants)
  - *alice*, *bob* ...

Complex concepts

- concept constructors: $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$ ...
  - Mother $\sqcup$ Father : "mothers or fathers"
  - Mother $\sqcap \neg\exists$hasChild.Male : "mothers who don't have any male child"

Complex roles

- role constructors: $^{-}$ (inverse), o (composition) ...

Syntax

DL knowledge base $=$
TBox (terminology, ontology) $+$ ABox (assertions, data)

Syntax

DL knowledge base =
TBox (terminology, ontology) + ABox (assertions, data)

A TBox describes general knowledge about the domain. It contains concept inclusions, role inclusions and possibly properties about roles (transitivity, functionality...).

- Mother $\sqsubseteq$ Parent : "all mothers are parents"
- Spouse $\sqsubseteq$ $\exists$isMarriedTo : "all siblings are married"
- hasParent $\sqsubseteq$ hasChild^{-}: "if x has parent y, then y has child x"

Syntax

DL knowledge base $=$
TBox (terminology, ontology) $+$ ABox (assertions, data)

A TBox describes general knowledge about the domain. It contains concept inclusions, role inclusions and possibly properties about roles (transitivity, functionality...).

- $\text{Mother} \sqsubseteq \text{Parent}$: "all mothers are parents"
- $\text{Spouse} \sqsubseteq \exists\text{isMarriedTo}$: "all siblings are married"
- $\text{hasParent} \sqsubseteq \text{hasChild}^-$: "if x has parent y, then y has child x"

An ABox contains facts about specific individuals. It contains concept assertions and role assertions.

- $\text{Mother}(\mathit{alice})$: "alice is a mother"

Syntax

DL knowledge base =
TBox (terminology, ontology) + ABox (assertions, data)

A TBox describes general knowledge about the domain. It contains concept inclusions, role inclusions and possibly properties about roles (transitivity, functionality...).

- $\mathsf{Mother} \sqsubseteq \mathsf{Parent}$: "all mothers are parents"
- $\mathsf{Spouse} \sqsubseteq \exists \mathsf{isMarriedTo}$: "all siblings are married"
- $\mathsf{hasParent} \sqsubseteq \mathsf{hasChild}^{-}$: "if x has parent y, then y has child x"

An ABox contains facts about specific individuals. It contains concept assertions and role assertions.

- $\mathsf{Mother}(alice)$: "alice is a mother"

To define a particular DL, we need to specify

- which concept and role constructors can be used
- what types of statements can appear in the TBox

Semantics

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

- $\Delta^{\mathcal{I}}$ is a non-empty set called domain
- $\cdot^{\mathcal{I}}$ is a function which associates
  - each constant a with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - each atomic concept A with a unary relation $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - each atomic role R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Semantics

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

- $\Delta^{\mathcal{I}}$ is a non-empty set called domain
- $\cdot^{\mathcal{I}}$ is a function which associates
  - each constant a with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - each atomic concept A with a unary relation $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - each atomic role R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

The function $\cdot^{\mathcal{I}}$ is extended to complex concepts and roles to formalize the meaning of the constructors:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(R^{-})^{\mathcal{I}} = \{(u, v) \mid (v, u) \in R^{\mathcal{I}}\}$
- $(\exists R.C)^{\mathcal{I}} = \{u \mid \text{ there exists } v \text{ such that } (u, v) \in R^{\mathcal{I}} \text{ and } v \in C^{\mathcal{I}}\}$
- $(\forall R.C)^{\mathcal{I}} = \{u \mid \text{ for every } v, \text{ if } (u, v) \in R^{\mathcal{I}} \text{ then } v \in C^{\mathcal{I}}\}$
- ...

Semantics

Satisfaction of TBox axioms

- $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subset D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role inclusion $R \sqsubseteq S$ if $R^{\mathcal{I}} \subset S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $(\mathit{func}\ R)$ if $R^{\mathcal{I}}$ is a functional relation
- ...

Satisfaction of ABox assertions

- $\mathcal{I}$ satisfies a concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role assertion $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Semantics

Satisfaction of TBox axioms

- $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subset D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role inclusion $R \sqsubseteq S$ if $R^{\mathcal{I}} \subset S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $(\mathit{func}\ R)$ if $R^{\mathcal{I}}$ is a functional relation
- ...

Satisfaction of ABox assertions

- $\mathcal{I}$ satisfies a concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role assertion $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Models

- $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$
- $\mathcal{I}$ is a model of a TBox $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$
- $\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$

Entailment

- A TBox $\mathcal{T}$ entails an axiom α, written $\mathcal{T} \models \alpha$, if every model of $\mathcal{T}$ satisfies α
- A KB $\langle \mathcal{T}, \mathcal{A} \rangle$ entails an assertion α, written $\langle \mathcal{T}, \mathcal{A} \rangle \models \alpha$, if every model of $\langle \mathcal{T}, \mathcal{A} \rangle$ satisfies α

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Female} \sqcap \exists \text{hasChild}.\top, \quad \exists \text{hasChild}.\top \sqsubseteq \text{Parent} \quad \}$$

$$\mathcal{A} = \{ \quad \text{hasChild}(\mathit{alice}, \mathit{john}), \quad \text{Female}(\mathit{alice}), \quad \text{Mother}(\mathit{mary}) \quad \}$$

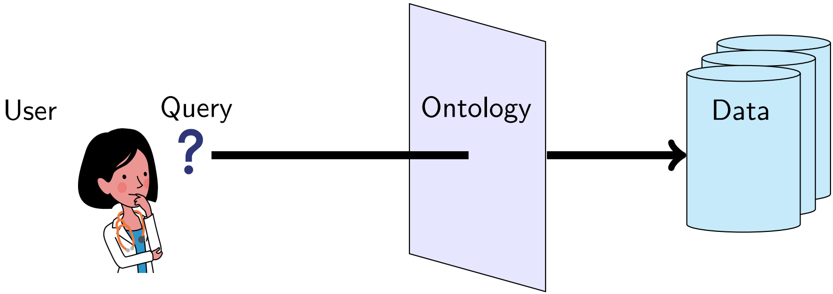
- $\mathcal{T} \models \text{Mother} \sqsubseteq \text{Parent}$
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \text{Parent}(\mathit{alice})$
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \text{Female}(\mathit{mary})$
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \text{Parent}(\mathit{mary})$

Classical reasoning tasks

- Satisfiability: does $\langle \mathcal{T}, \mathcal{A} \rangle$ have a model ?
- Subsumption: does $\mathcal{T} \models C \sqsubseteq D$?
- Classification: find all atomic A, B such that $\mathcal{T} \models A \sqsubseteq B$
- Instance checking: does $\langle \mathcal{T}, \mathcal{A} \rangle \models C(a)$?

2. Query answering

Ontology-mediated query answering



Queries

Beyond instance checking

Different kinds of queries: first order queries, path queries...

Focus on conjunctive queries
$\sim$ select-project-join queries in relational databases
$\sim$ Datalog rules (subset of Prolog)

Conjunctive queries

A conjunctive query (CQ) has the form

$$q(x_1, \ldots, x_k) = \exists x_{k+1}, \ldots, x_m \; \alpha_1 \wedge \cdots \wedge \alpha_n$$

where $\alpha_1, \ldots, \alpha_n$ are atoms of the form $A(v)$ or $R(v_1, v_2)$ with A atomic concept, R atomic role, and v, v_1, v_2 constants or variables from $x_1, \ldots, x_m$.

$x_1, \ldots, x_k$ are the answer variables and $x_{k+1}, \ldots, x_m$ are the (existentially) quantified variables.

Boolean CQs are CQs without answer variables.

In general, not expressible as instance queries!

Semantics

An interpretation $\mathcal{I}$ satisfies a Boolean CQ $q = \exists x_1, \ldots, x_m\ \alpha_1 \wedge \cdots \wedge \alpha_n$ if there exists a function $\pi : \{x_1, \ldots, x_m\} \rightarrow \Delta^{\mathcal{I}}$ such that $\mathcal{I}$ satisfies each assertion α_i^{π} obtained by replacing any occurrence of x_j by $\pi(x_j)$. We say that π is a match for q in $\mathcal{I}$.

A Boolean CQ q is entailed from $\langle \mathcal{T}, \mathcal{A} \rangle$ ($\langle \mathcal{T}, \mathcal{A} \rangle \models q$) iff every model of $\langle \mathcal{T}, \mathcal{A} \rangle$ satisfies q.

A tuple $(a_1, \ldots, a_k)$ of individuals is a certain answer to $q(x_1, \ldots, x_k)$ w.r.t. $\langle \mathcal{T}, \mathcal{A} \rangle$ iff $\langle \mathcal{T}, \mathcal{A} \rangle \models q(a_1, \ldots, a_k)$ where $q(a_1, \ldots, a_k)$ is the Boolean CQ obtained from $q(x_1, \ldots, x_k)$ by replacing each x_i by a_i.

Andrea's example

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

$$\begin{aligned} \mathcal{A} = \{ \quad & \mathsf{friend}(john, susan), && \mathsf{friend}(john, andrea), \\ & \mathsf{loves}(susan, andrea), && \mathsf{loves}(andrea, bill), \\ & \mathsf{Female}(susan), && \mathsf{Male}(bill) \quad \} \end{aligned}$$

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

Challenges

Answering CQ $=$ deciding if there is a match in every model

- infinitely many models
- models can be infinite

May lead to hight computational complexity

Challenges

Answering CQ = deciding if there is a match in every model

- infinitely many models
- models can be infinite

May lead to hight computational complexity

Some DLs are such that every satisfiable KB has a universal model $\mathcal{U}$ such that any CQ q is entailed by the KB iff $\mathcal{U}$ satisfies q

Such a model can be obtained by saturating the ABox using the TBox

Still challenging since $\mathcal{U}$ may be infinite

3. Query answering in DL-Lite

DL-Lite

Lightweight DLs

- good computational properties (classical reasoning tasks in PTIME)
- limited (but useful) expressivity

DL-Lite family

- designed to handle large ABoxes
- designed for efficient conjunctive query answering
- basis of the W3C standard OWL 2 QL for the Semantic Web

DL-Lite

DL-Lite$_{\mathcal{R}}$ TBox inclusions of the form $B \sqsubseteq C$ or $S \sqsubseteq Q$ where

$$B := A \mid \exists S, \quad C := B \mid \neg B, \quad S := R \mid R^{-}, \quad Q := S \mid \neg S$$

with A an atomic concept and R an atomic role

Note that the universal model may be infinite

DL-Lite for ontology-mediated query answering

Idea: exploit the efficiency of relational database systems

Approach: query rewriting

- ABox is stored as a traditional database
- the input query is rewritten to integrate the relevant information from the TBox
- the new query is evaluated over the database

Rewriting

Define the database-like interpretation $\mathcal{I}_\mathcal{A}$, of the ABox $\mathcal{A}$ by

- $\Delta^{\mathcal{I}_\mathcal{A}} = \mathit{Inds}(\mathcal{A})$
- $a^{\mathcal{I}_\mathcal{A}} = a$ for every individual a
- $A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\}$ for every atomic concept A
- $R^{\mathcal{I}_\mathcal{A}} = \{(a, b) \mid R(a, b) \in \mathcal{A}\}$ for every atomic concept R

A first-order query q' is called a perfect rewriting of a CQ q w.r.t. a TBox $\mathcal{T}$ if and only if for every ABox $\mathcal{A}$, the certain answers of q over $\langle \mathcal{T}, \mathcal{A} \rangle$ are the same as the answers of q' in $\mathcal{I}_\mathcal{A}$.

In DL-Lite, a perfect rewriting exists for every CQ and TBox.

$$\mathcal{T} = \{ \quad \mathsf{Mother} \sqsubseteq \mathsf{Parent}, \qquad \mathsf{Parent} \sqsubseteq \exists \mathsf{hasChild},$$
$$\mathsf{Parent} \sqsubseteq \mathsf{Person}, \qquad \exists \mathsf{isMarriedTo} \sqsubseteq \mathsf{Person} \quad \}$$

$$\mathcal{A} = \{ \ \mathsf{Mother}(\mathit{mary}), \quad \mathsf{hasChild}(\mathit{alice}, \mathit{john}), \quad \mathsf{isMarriedTo}(\mathit{alice}, \mathit{bob}) \ \}$$

$$q(x) = \exists y \mathsf{Person}(x) \wedge \mathsf{hasChild}(x, y)$$

$$\mathcal{T} = \{ \quad \mathsf{Mother} \sqsubseteq \mathsf{Parent}, \qquad \mathsf{Parent} \sqsubseteq \exists \mathsf{hasChild},$$
$$\mathsf{Parent} \sqsubseteq \mathsf{Person}, \qquad \exists \mathsf{isMarriedTo} \sqsubseteq \mathsf{Person} \quad \}$$

$$\mathcal{A} = \{ \mathsf{Mother}(\mathit{mary}), \quad \mathsf{hasChild}(\mathit{alice}, \mathit{john}), \quad \mathsf{isMarriedTo}(\mathit{alice}, \mathit{bob}) \}$$

$$q(x) = \exists y \mathsf{Person}(x) \wedge \mathsf{hasChild}(x, y)$$

$$q'(x) = (\exists y \ \mathsf{Person}(x) \wedge \mathsf{hasChild}(x, y))$$

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Parent}, \qquad \text{Parent} \sqsubseteq \exists \text{hasChild}, \\ \text{Parent} \sqsubseteq \text{Person}, \qquad \exists \text{isMarriedTo} \sqsubseteq \text{Person} \quad \}$$

$$\mathcal{A} = \{ \text{Mother}(\mathit{mary}), \quad \text{hasChild}(\mathit{alice}, \mathit{john}), \quad \text{isMarriedTo}(\mathit{alice}, \mathit{bob}) \}$$

$$q(x) = \exists y \text{Person}(x) \land \text{hasChild}(x, y)$$

$$q'(x) = (\exists y \ \text{Person}(x) \land \text{hasChild}(x, y)) \lor (\exists y \ \text{Parent}(x) \land \text{hasChild}(x, y))$$

$$\mathcal{T} = \{ \quad \mathsf{Mother} \sqsubseteq \mathsf{Parent}, \qquad \mathsf{Parent} \sqsubseteq \exists \mathsf{hasChild},$$
$$\mathsf{Parent} \sqsubseteq \mathsf{Person}, \qquad \exists \mathsf{isMarriedTo} \sqsubseteq \mathsf{Person} \quad \}$$

$$\mathcal{A} = \{ \ \mathsf{Mother}(mary), \quad \mathsf{hasChild}(alice, john), \quad \mathsf{isMarriedTo}(alice, bob) \ \}$$

$$q(x) = \exists y \mathsf{Person}(x) \wedge \mathsf{hasChild}(x, y)$$

$$q'(x) = (\exists y \ \mathsf{Person}(x) \wedge \mathsf{hasChild}(x, y)) \vee (\exists y \ \mathsf{Parent}(x) \wedge \mathsf{hasChild}(x, y))$$
$$\vee (\exists y \ \mathsf{Mother}(x) \wedge \mathsf{hasChild}(x, y))$$

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Parent}, \qquad \text{Parent} \sqsubseteq \exists\text{hasChild}, \\ \text{Parent} \sqsubseteq \text{Person}, \qquad \exists\text{isMarriedTo} \sqsubseteq \text{Person} \quad \}$$

$$\mathcal{A} = \{ \text{ Mother}(\textit{mary}), \quad \text{hasChild}(\textit{alice}, \textit{john}), \quad \text{isMarriedTo}(\textit{alice}, \textit{bob}) \}$$

$$q(x) = \exists y \text{Person}(x) \wedge \text{hasChild}(x, y)$$

$$q'(x) = (\exists y \ \text{Person}(x) \wedge \text{hasChild}(x, y)) \vee (\exists y \ \text{Parent}(x) \wedge \text{hasChild}(x, y)) \\ \vee (\exists y \ \text{Mother}(x) \wedge \text{hasChild}(x, y)) \vee (\text{Person}(x) \wedge \text{Parent}(x))$$

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Parent}, \qquad \text{Parent} \sqsubseteq \exists \text{hasChild}, \\ \text{Parent} \sqsubseteq \text{Person}, \qquad \exists \text{isMarriedTo} \sqsubseteq \text{Person} \quad \}$$

$$\mathcal{A} = \{ \text{Mother}(\mathit{mary}), \quad \text{hasChild}(\mathit{alice}, \mathit{john}), \quad \text{isMarriedTo}(\mathit{alice}, \mathit{bob}) \}$$

$$q(x) = \exists y \text{Person}(x) \wedge \text{hasChild}(x, y)$$

$$\begin{aligned} q'(x) = \ & (\exists y \ \text{Person}(x) \wedge \text{hasChild}(x, y)) \vee (\exists y \ \text{Parent}(x) \wedge \text{hasChild}(x, y)) \\ & \vee (\exists y \ \text{Mother}(x) \wedge \text{hasChild}(x, y)) \vee (\text{Person}(x) \wedge \text{Parent}(x)) \\ & \vee \text{Parent}(x) \end{aligned}$$

$$\mathcal{T} = \{ \quad \mathsf{Mother} \sqsubseteq \mathsf{Parent}, \qquad \mathsf{Parent} \sqsubseteq \exists \mathsf{hasChild},$$
$$\mathsf{Parent} \sqsubseteq \mathsf{Person}, \qquad \exists \mathsf{isMarriedTo} \sqsubseteq \mathsf{Person} \quad \}$$

$$\mathcal{A} = \{ \mathsf{Mother}(mary), \quad \mathsf{hasChild}(alice, john), \quad \mathsf{isMarriedTo}(alice, bob) \}$$

$$q(x) = \exists y \mathsf{Person}(x) \wedge \mathsf{hasChild}(x, y)$$

$$\begin{aligned} q'(x) = \ & (\exists y\ \mathsf{Person}(x) \wedge \mathsf{hasChild}(x, y)) \vee (\exists y\ \mathsf{Parent}(x) \wedge \mathsf{hasChild}(x, y)) \\ & \vee (\exists y\ \mathsf{Mother}(x) \wedge \mathsf{hasChild}(x, y)) \vee (\mathsf{Person}(x) \wedge \mathsf{Parent}(x)) \\ & \vee \mathsf{Parent}(x) \vee (\mathsf{Mother}(x) \wedge \mathsf{Parent}(x)) \end{aligned}$$

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Parent}, \qquad \text{Parent} \sqsubseteq \exists\text{hasChild}, \\ \text{Parent} \sqsubseteq \text{Person}, \qquad \exists\text{isMarriedTo} \sqsubseteq \text{Person} \quad \}$$

$$\mathcal{A} = \{ \text{Mother}(\mathit{mary}), \quad \text{hasChild}(\mathit{alice}, \mathit{john}), \quad \text{isMarriedTo}(\mathit{alice}, \mathit{bob}) \}$$

$$q(x) = \exists y \text{Person}(x) \wedge \text{hasChild}(x, y)$$

$$q'(x) = (\exists y \ \text{Person}(x) \wedge \text{hasChild}(x, y)) \vee (\exists y \ \text{Parent}(x) \wedge \text{hasChild}(x, y)) \\ \vee (\exists y \ \text{Mother}(x) \wedge \text{hasChild}(x, y)) \vee (\text{Person}(x) \wedge \text{Parent}(x)) \\ \vee \text{Parent}(x) \vee (\text{Mother}(x) \wedge \text{Parent}(x)) \vee \text{Mother}(x)$$

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Parent}, \qquad \text{Parent} \sqsubseteq \exists\text{hasChild}, \\ \text{Parent} \sqsubseteq \text{Person}, \qquad \exists\text{isMarriedTo} \sqsubseteq \text{Person} \quad \}$$

$$\mathcal{A} = \{ \text{Mother}(\mathit{mary}), \quad \text{hasChild}(\mathit{alice}, \mathit{john}), \quad \text{isMarriedTo}(\mathit{alice}, \mathit{bob}) \}$$

$$q(x) = \exists y \text{Person}(x) \wedge \text{hasChild}(x, y)$$

$$\begin{aligned} q'(x) = \; & (\exists y \ \text{Person}(x) \wedge \text{hasChild}(x, y)) \vee (\exists y \ \text{Parent}(x) \wedge \text{hasChild}(x, y)) \\ & \vee (\exists y \ \text{Mother}(x) \wedge \text{hasChild}(x, y)) \vee (\text{Person}(x) \wedge \text{Parent}(x)) \\ & \vee \text{Parent}(x) \vee (\text{Mother}(x) \wedge \text{Parent}(x)) \vee \text{Mother}(x) \\ & \vee (\exists yz \ \text{isMarriedTo}(x, z) \wedge \text{hasChild}(x, y)) \end{aligned}$$

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Parent}, \qquad \text{Parent} \sqsubseteq \exists \text{hasChild}, \\ \text{Parent} \sqsubseteq \text{Person}, \qquad \exists \text{isMarriedTo} \sqsubseteq \text{Person} \quad \}$$

$$\mathcal{A} = \{ \text{Mother}(\mathit{mary}), \quad \text{hasChild}(\mathit{alice}, \mathit{john}), \quad \text{isMarriedTo}(\mathit{alice}, \mathit{bob}) \}$$

$$q(x) = \exists y \text{Person}(x) \land \text{hasChild}(x, y)$$

$$\begin{aligned} q'(x) = \; & (\exists y \; \text{Person}(x) \land \text{hasChild}(x, y)) \lor (\exists y \; \text{Parent}(x) \land \text{hasChild}(x, y)) \\ & \lor (\exists y \; \text{Mother}(x) \land \text{hasChild}(x, y)) \lor (\text{Person}(x) \land \text{Parent}(x)) \\ & \lor \text{Parent}(x) \lor (\text{Mother}(x) \land \text{Parent}(x)) \lor \text{Mother}(x) \\ & \lor (\exists yz \; \text{isMarriedTo}(x, z) \land \text{hasChild}(x, y)) \\ & \lor (\exists z \; \text{isMarriedTo}(x, z) \land \text{Parent}(x)) \end{aligned}$$

$$\mathcal{T} = \{ \quad \text{Mother} \sqsubseteq \text{Parent}, \qquad \text{Parent} \sqsubseteq \exists \text{hasChild}, \\ \text{Parent} \sqsubseteq \text{Person}, \qquad \exists \text{isMarriedTo} \sqsubseteq \text{Person} \quad \}$$

$$\mathcal{A} = \{ \text{Mother}(\textit{mary}), \quad \text{hasChild}(\textit{alice}, \textit{john}), \quad \text{isMarriedTo}(\textit{alice}, \textit{bob}) \}$$

$$q(x) = \exists y \text{Person}(x) \wedge \text{hasChild}(x, y)$$

$$\begin{aligned} q'(x) = \; & (\exists y \ \text{Person}(x) \wedge \text{hasChild}(x, y)) \vee (\exists y \ \text{Parent}(x) \wedge \text{hasChild}(x, y)) \\ & \vee (\exists y \ \text{Mother}(x) \wedge \text{hasChild}(x, y)) \vee (\text{Person}(x) \wedge \text{Parent}(x)) \\ & \vee \text{Parent}(x) \vee (\text{Mother}(x) \wedge \text{Parent}(x)) \vee \text{Mother}(x) \\ & \vee (\exists yz \ \text{isMarriedTo}(x, z) \wedge \text{hasChild}(x, y)) \\ & \vee (\exists z \ \text{isMarriedTo}(x, z) \wedge \text{Parent}(x)) \\ & \vee (\exists z \ \text{isMarriedTo}(x, z) \wedge \text{Mother}(x)) \end{aligned}$$

Let g be an atom and I be a positive inclusion. The atom obtained from g by applying I, denoted by $gr(g, I)$, is defined as follows:

- if $g = A(x)$ and $I = A_1 \sqsubseteq A$, then $gr(g, I) = A_1(x)$
- if $g = A(x)$ and $I = \exists R \sqsubseteq A$, then $gr(g, I) = R(x, \_)$
- if $g = A(x)$ and $I = \exists R^- \sqsubseteq A$, then $gr(g, I) = R(\_ , x)$
- if $g = R(x, \_)$ and $I = A \sqsubseteq \exists R$, then $gr(g, I) = A(x)$
- if $g = R(x, \_)$ and $I = \exists R_1 \sqsubseteq \exists R$, then $gr(g, I) = R_1(x, \_)$
- if $g = R(x, \_)$ and $I = \exists R_1^- \sqsubseteq \exists R$, then $gr(g, I) = R_1(\_ , x)$
- if $g = R(\_ , x)$ and $I = A \sqsubseteq \exists R^-$, then $gr(g, I) = A(x)$
- if $g = R(\_ , x)$ and $I = \exists R_1 \sqsubseteq \exists R^-$, then $gr(g, I) = R_1(x, \_)$
- if $g = R(\_ , x)$ and $I = \exists R_1^- \sqsubseteq \exists R^-$, then $gr(g, I) = R_1(\_ , x)$
- if $g = R(x, y)$ and $I = R_1 \sqsubseteq R$ or $I = R_1^- \sqsubseteq R^-$, then $gr(g, I) = R_1(x, y)$
- if $g = R(x, y)$ and $I = R_1 \sqsubseteq R^-$ or $I = R_1^- \sqsubseteq R$, then $gr(g, I) = R_1(y, x)$

Input: a conjunctive query q, a TBox $\mathcal{T}$
Output: a union of conjunctive queries PR

$PR \leftarrow \{q\}$, $PR' \leftarrow \emptyset$
While $PR' \neq PR$

$PR' \leftarrow PR$

for all $q \in PR'$

for all $g \in q$, **for all** $I \in \mathcal{T}$ applicable to g
$PR \leftarrow PR \cup \{q[g \leftarrow gr(g, I)]\}$
for all $g_1, g_2 \in q$
if g_1 and g_2 unify
$PR \leftarrow PR \cup \{\mathrm{reduce}(q, g_1, g_2)\}$

Return PR

reduce: applies to q the most general unifier between g_1 and g_2 then replaces each unbound variable (existentially quantified and not shared between atoms) with _

$$\mathcal{T} = \{ \quad \begin{array}{ll} \mathsf{hasChild} \sqsubseteq \mathsf{parentOf}^-, & \exists \mathsf{hasChild} \sqsubseteq \mathsf{Parent}, \\ \mathsf{Spouse} \sqsubseteq \exists \mathsf{isMarriedTo}, & \exists \mathsf{isMarriedTo} \sqsubseteq \mathsf{Spouse}, \\ \mathsf{sisterOf} \sqsubseteq \mathsf{siblingOf}, & \mathsf{siblingOf} \sqsubseteq \mathsf{siblingOf}^- \end{array} \quad \}$$

$$q_1(x) = \exists yz \ \mathsf{isMarriedTo}(x, y) \wedge \mathsf{siblingOf}(y, z)$$

$$q_2(x) = \exists y \ \mathsf{Parent}(x) \wedge \mathsf{parentOf}(x, y) \wedge \mathsf{Spouse}(y)$$

Complexity

Rewriting

- polynomial time w.r.t. the size of the TBox
- exponential w.r.t. the size of the query

Evaluation of the rewritten query in polynomial time w.r.t. the size of the ABox

Query entailment is NP-complete w.r.t. the size of the KB and query (combined complexity), and in PTIME w.r.t. the size of the ABox (data complexity)

4. Querying inconsistent data

Inconsistency in ontology-mediated query answering

TBox $\mathcal{T}$

$\mathsf{Father} \sqsubseteq \mathsf{Parent}$	*fathers are parents*
$\mathsf{Mother} \sqsubseteq \mathsf{Parent}$	*mothers are parents*
$\mathsf{Father} \sqsubseteq \neg\mathsf{Mother}$	*concepts* Father *and* Mother *are disjoint*

ABox $\mathcal{A}$

$\mathsf{sibling}(\mathit{fred}, \mathit{bob})$	*fred and bob are siblings*
$\mathsf{Mother}(\mathit{fred})$	*fred is a mother*

Conjunctive query q

$$\exists y\, \mathsf{Parent}(x) \land \mathsf{sibling}(x, y)$$

Query certain answer: *fred*

Inconsistency in ontology-mediated query answering

TBox $\mathcal{T}$

$\mathsf{Father} \sqsubseteq \mathsf{Parent}$	*fathers are parents*
$\mathsf{Mother} \sqsubseteq \mathsf{Parent}$	*mothers are parents*
$\mathsf{Father} \sqsubseteq \neg\mathsf{Mother}$	*concepts* Father *and* Mother *are disjoint*

ABox $\mathcal{A}$

$\mathsf{sibling}(fred, bob)$	*fred and bob are siblings*
$\mathsf{Mother}(fred)$	*fred is a mother*
$\mathsf{Father}(fred)$	*fred is a father*

Conjunctive query q

$\exists y\, \mathsf{Parent}(x) \wedge \mathsf{sibling}(x, y)$

Query certain answers: *fred*, *bob*

Inconsistency $\implies$ no model $\implies$ everything is entailed

Inconsistency-tolerant semantics

Repair

maximal subset of the ABox consistent with the TBox

TBox $\mathcal{T}$

Father $\sqsubseteq$ Parent
Mother $\sqsubseteq$ Parent
Father $\sqsubseteq \neg$Mother

ABox $\mathcal{A}$

sibling(fred, bob)
Mother(fred)
Father(fred)

Repair $\mathcal{R}_1$

sibling(fred, bob)
Mother(fred)

Repair $\mathcal{R}_2$

sibling(fred, bob)
Father(fred)

Inconsistency-tolerant semantics

Sure answers semantics (AR semantics)

sure answer $\Leftrightarrow$ entailed by every repair

TBox $\mathcal{T}$

$\text{Father} \sqsubseteq \text{Parent}$
$\text{Mother} \sqsubseteq \text{Parent}$
$\text{Father} \sqsubseteq \neg\text{Mother}$

ABox $\mathcal{A}$

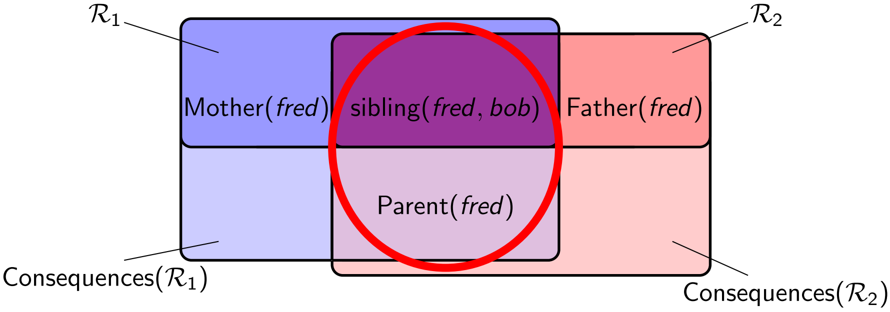
$\text{sibling}(\mathit{fred}, \mathit{bob})$
$\text{Mother}(\mathit{fred})$
$\text{Father}(\mathit{fred})$

Repair $\mathcal{R}_1$

$\text{sibling}(\mathit{fred}, \mathit{bob})$
$\text{Mother}(\mathit{fred})$

Repair $\mathcal{R}_2$

$\text{sibling}(\mathit{fred}, \mathit{bob})$
$\text{Father}(\mathit{fred})$



Inconsistency-tolerant semantics

Sure answers semantics (AR semantics)

sure answer $\Leftrightarrow$ entailed by every repair

TBox $\mathcal{T}$

Father $\sqsubseteq$ Parent
Mother $\sqsubseteq$ Parent
Father $\sqsubseteq \neg$Mother

ABox $\mathcal{A}$

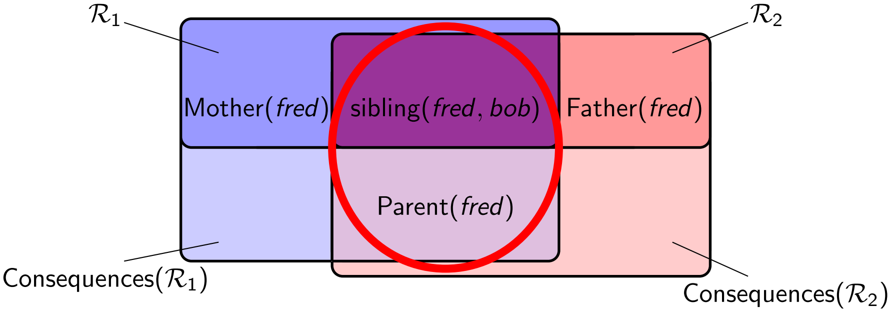
sibling(*fred*, *bob*)
Mother(*fred*)
Father(*fred*)

Repair $\mathcal{R}_1$

sibling(*fred*, *bob*)
Mother(*fred*)

Repair $\mathcal{R}_2$

sibling(*fred*, *bob*)
Father(*fred*)



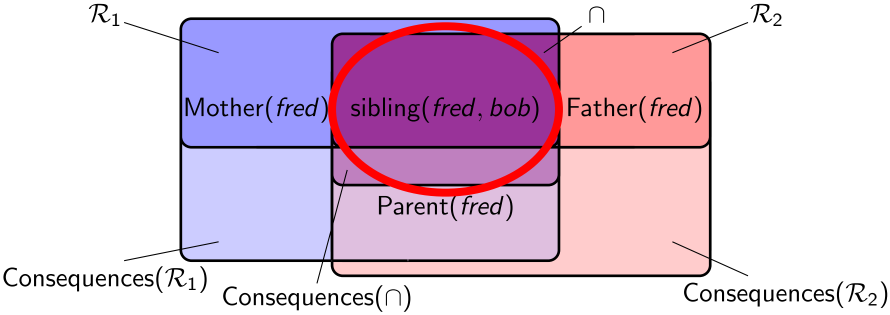
natural but difficult to compute! $\Longrightarrow$ approximations

Inconsistency-tolerant semantics

Intersection answers semantics (IAR semantics)

intersection answer $\Leftrightarrow$ entailed by the intersection of the repairs

TBox $\mathcal{T}$	ABox $\mathcal{A}$	Repair $\mathcal{R}_1$	Repair $\mathcal{R}_2$
$\text{Father} \sqsubseteq \text{Parent}$	sibling(*fred*, *bob*)	sibling(*fred*, *bob*)	sibling(*fred*, *bob*)
$\text{Mother} \sqsubseteq \text{Parent}$	Mother(*fred*)	Mother(*fred*)	
$\text{Father} \sqsubseteq \neg\text{Mother}$	Father(*fred*)		Father(*fred*)

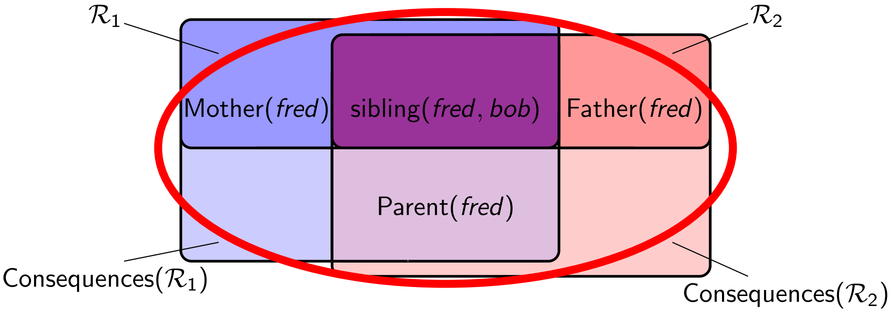


Inconsistency-tolerant semantics

Possible answers semantics (brave semantics)

possible answer $\Leftrightarrow$ entailed by (at least) one repair

TBox $\mathcal{T}$	ABox $\mathcal{A}$	Repair $\mathcal{R}_1$	Repair $\mathcal{R}_2$
Father $\sqsubseteq$ Parent	sibling(*fred*, *bob*)	sibling(*fred*, *bob*)	sibling(*fred*, *bob*)
Mother $\sqsubseteq$ Parent	Mother(*fred*)	Mother(*fred*)	
Father $\sqsubseteq \neg$Mother	Father(*fred*)		Father(*fred*)



Inconsistency-tolerant semantics

intersection answers $\subseteq$ sure answers $\subseteq$ possible answers

Remark: other inconsistency-tolerant semantics exist

Data complexity of query entailment in DL-Lite$_{\mathcal{R}}$:

- intersection / possible / classical semantics: in polynomial time
- sure: coNP-complete

Methods for efficient query answering under sure semantics in practice

References

Franz Baader et al., eds. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

Diego Calvanese et al. "Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family". In: *Journal of Automated Reasoning (JAR)* 39.3 (2007), pp. 385–429.

Boris Motik et al. *OWL 2 Web Ontology Language Profiles*. W3C Recommendation. Available at `http://www.w3.org/TR/owl2-profiles/`. Nov. 2012.

Diego Calvanese et al. "Ontologies and Databases: The DL-Lite Approach". In: *Reasoning Web, Tutorial Lectures*. 2009, pp. 255–356.

Meghyn Bienvenu and Magdalena Ortiz. "Ontology-Mediated Query Answering with Data-Tractable Description Logics". In: *Reasoning Web, Tutorial Lectures*. 2015, pp. 218–307.

Domenico Lembo et al. "Inconsistency-Tolerant Semantics for Description Logics". In: *Proceedings of RR*. 2010.

Meghyn Bienvenu and Camille Bourgaux. "Inconsistency-Tolerant Querying of Description Logic Knowledge Bases". In: *Reasoning Web, Tutorial Lectures*. 2016, pp. 156–202.