Logik für Informatiker
Logic for computer scientists

Ontologies: Description Logics

Till Mossakowski

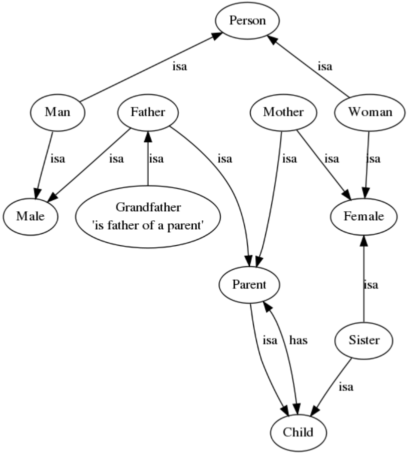
WiSe 2009/10

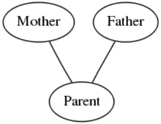
Ontology languages

- description logics (efficiently decidable fragments of first-order logic)
    - used for domain ontologies
    - standardised in web ontology language OWL
- first-order logics
    - used for upper ontologies
    - standardised in Common Logic, CASL

Semantic networks

- used for representation of and reasoning about knowledge
- e.g. KL-ONE: reasoning about concepts, subclassing and their relations





Description Logics

- drawback of semantic networks:
  - often, meaning of arrows is not precisely defined
  - sometimes, full first-order logic is used $\Rightarrow$ undecidable
- Description Logics:
  - completely formal syntax and semantics,
  - decidable fragments of first-order logic
  - efficient reasoning tools available (Pellet, Fact++, Racer)

Description Logics: Concepts

- *Concepts* (in OWL: *classes*) (Mother, Father, etc.)
- *Subsumption* $C \sqsubseteq D$ (read: "C is subsumed by D") means that each C is a D
  - $Woman \sqsubseteq Person$
  - $Father \sqsubseteq Male$
  - ...

Description Logics: Roles

- To relate concepts, we need *roles* (in OWL: *properties*) like 'hasChild'.
  - $Parent \sqsubseteq \exists hasChild.\top$ ($\top$: top concept, includes everything. In OWL: *Thing*)
  - $Parent \sqsubseteq \exists hasChild.Child$
  - $Child \sqsubseteq \exists hasParent.\top$ (Bad, because *hasChild* is converse to *hasParent* which is not expressed here)
  - $Child \sqsubseteq \exists hasChild^{-}.\top$ (Better formalization)
  - $hasParent \equiv hasChild^{-}$ (Alternative, not possible in every DL)
  - $Grandfather \equiv (\exists hasChild.\exists hasChild.\top) \sqcap Male$ ($C \equiv D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$)
  - $Grandfather \equiv (\exists hasChild.Parent) \sqcap Father$ (Alternative formalization)

Description Logics: Signatures

A *DL-signature* $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ consists of

- a set $\mathbf{C}$ of concept names,
- a set $\mathbf{R}$ of role names,
- a set $\mathbf{I}$ of individual names,

Description Logics: Concepts

For a signature $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ the set of $\mathcal{ALC}$-*concepts* over Σ is defined by the following grammar:

		(Hets) Manchester syntax
$C ::=$	A for $A \in \mathbf{C}$	a concept name
	$\mid \top$	`Thing`
	$\mid \bot$	`Nothing`
	$\mid \neg C$	`not C`
	$\mid C \sqcap C$	`C and C`
	$\mid C \sqcup C$	`C or C`
	$\mid \exists R.C$ for $R \in \mathbf{R}$	`R some C`
	$\mid \forall R.C$ for $R \in \mathbf{R}$	`R only C`

$\mathcal{ALC}$ stands for "attributive language with complement"

Description Logic: Sentences

The set of $\mathcal{ALC}$-Sentences over Σ (Sen(Σ)) is defined as

- $C \sqsubseteq D$, where C and D are $\mathcal{ALC}$-concepts over Σ.

  ```
  Class: C SubclassOf: D
  ```

- $a : C$, where $a \in \mathbf{I}$ and C is a $\mathcal{ALC}$-concept over Σ.

  ```
  Individual: a Types: C
  ```

- $R(a_1, a_2)$, where $R \in \mathbf{R}$ and $a_1, a_2 \in \mathbf{I}$.

  ```
  Individual: a1 Facts: R a2
  ```

TBoxes and ABoxes

Description logics axioms are generally split up in two sets:

- *TBox*: subsumptions and definitions involving concepts and roles
  - e.g. $Woman \sqsubseteq Person$
- *ABox*: individuals and their membership in concepts and roles
  - e.g. $john : Father$, $hasChild(john, harry)$

Given $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$, a Σ-*model* is of form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

- $\Delta^{\mathcal{I}}$ is a non-empty set
- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each $A \in \mathbf{C}$
- $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each $R \in \mathbf{R}$
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each $a \in \mathbf{I}$

Description Logic: semantics of concepts

We can extend $\cdot^{\mathcal{I}}$ to all concepts as follows:

$$
\begin{array}{lcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} | \exists y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} | \forall y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}
\end{array}
$$

Description Logic: Satisfaction of sentences in a model

$$
\begin{array}{lll}
\mathcal{I} \models C \sqsubseteq D & \text{iff} & C^{\mathcal{I}} \subseteq D^{\mathcal{I}}. \\
\mathcal{I} \models a : C & \text{iff} & a^{\mathcal{I}} \in C^{\mathcal{I}}. \\
\mathcal{I} \models R(a_1, a_2) & \text{iff} & (a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in R^{\mathcal{I}}.
\end{array}
$$

Description Logic: Logical Consequence

For $\Gamma \subseteq \text{Sen}(\Sigma), \phi \in \text{Sen}(\Sigma)$, ϕ is a *logical consequence* of Γ (written: $\Gamma \models_\Sigma \phi$), if for each Σ-model $\mathcal{I}$

$$\mathcal{I} \models \Gamma \text{ implies } \mathcal{I} \models \phi.$$

If Γ contains only subsumptions, Γ is written as $\mathcal{T}$ (TBox).
If Γ contains only sentences $a : C$ and $R(a_1, a_2)$, Γ is written as $\mathcal{A}$ (ABox).

Example: a pizza ontology

$$
\begin{array}{lll}
\text{VegetarianPizza} & \sqsubseteq & \text{Pizza} \\
\text{MagheritaPizza} & \sqsubseteq & \text{Pizza} \\
\text{TomatoTopping} & \sqsubseteq & \text{VegetableTopping} \\
\text{MozzarellaTopping} & \sqsubseteq & \text{CheeseTopping} \\
\text{VegetarianPizza} & \equiv & \forall\ \text{hasTopping}\ (\text{VegetableTopping} \sqcup \text{CheeseTopping}) \\
\text{MagheritaPizza} & \sqsubseteq & \exists\ \text{hasTopping}\ \text{MozarellaTopping} \sqcap \\
 & & \exists\ \text{hasTopping}\ \text{TomatoTopping} \sqcap \\
 & & \forall\ \text{hasTopping} \\
 & & \quad (\text{MozzarellaTopping} \sqcup \text{TomatoTopping})
\end{array}
$$

Logical consequence: $\text{MagheritaPizza} \sqsubseteq \text{VegetarianPizza}$

TBox reasoning

Usually, satisfiability of concepts is tested. A concept C is *satisfiable* in a TBox iff there is a model of the TBox that leads to a non-empty interpretation of C.

Satisfiability and subsumption are inter-reducible:

$$\mathcal{T} \models C \sqsubseteq D \quad \text{iff} \quad \mathcal{T} \models unsat(C \sqcap \neg D)$$
$$\mathcal{T} \models unsat(C) \quad \text{iff} \quad \mathcal{T} \models C \sqsubseteq \bot$$

Complexity of TBox reasoning for $\mathcal{ALC}$:

- general TBoxes: EXPTIME complete
- empty or acyclic TBoxes: PSPACE complete[1].

Acyclic TBoxes contain only definitions $A \equiv C$, such that concept dependency is acyclic (A depends on all concepts occuring in C).

[1] We know that P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME and that P $\subset$ EXPTIME, so it is possible that PSPACE $\subset$ EXPTIME.

ABox-Reasoning

For example: Instance checking:

$$\mathcal{T}, \mathcal{A} \models a : C \text{ } \textit{iff} \text{ } \mathcal{T} \cup \mathcal{A} \cup \{ \textit{ not } a : C \} \textit{ inconsistent}$$

Complexity of deciding ABox consistency may be harder than TBox reasoning, but it usually is not.
For $\mathcal{ALC}$ it is PSPACE/EXPTIME complete.