Description Logics

Knowledge Bases in Description Logics

Enrico Franconi

`franconi@cs.man.ac.uk`

`http://www.cs.man.ac.uk/~franconi`

Department of Computer Science, University of Manchester

Understanding Knowledge Bases

$$\boxed{\Sigma = \langle \text{TBox}, \text{Abox} \rangle}$$

- **Terminological Axioms:** $C \sqsubseteq D$
- **Assertional Axioms:** $C(a), R(a, b)$

- An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I}$ satisfies $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $\mathcal{I}$ satisfies $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is said to be a *model* of Σ if every axiom of Σ is satisfied by $\mathcal{I}$. Σ is said to be *satisfiable* if it admits a model.

TBox statements

(1)	$A \sqsubseteq C$	Primitive concept definition
(2)	$A \doteq C$	Concept definition
(3)	$C \sqsubseteq D$	Concept inclusion
(4)	$C \doteq D$	Concept equation

Acyclic simple TBox

Simple TBox:	(1)	$A \sqsubseteq C$	Primitive concept definition
	(2)	$A \doteq C$	Concept definition

Acyclic simple TBox: well-founded definitions.

A concept name A *directly uses* a concept name B in a TBox Σ iff the definition of A mentions B. A concept name A *uses* a concept name B_n iff there is a chain of concept names $\langle A, B_1, \ldots, B_n \rangle$ such that B_i directly uses B_{i+1}. A TBox is *acyclic* iff no concept name uses itself.

Acyclic simple TBox

Subsumtpion in acyclic simple TBoxes ($\Sigma \models C \sqsubseteq D$) can be reduced in subsumtpion in an empty TBox ($\models \widehat{C} \sqsubseteq \widehat{D}$).

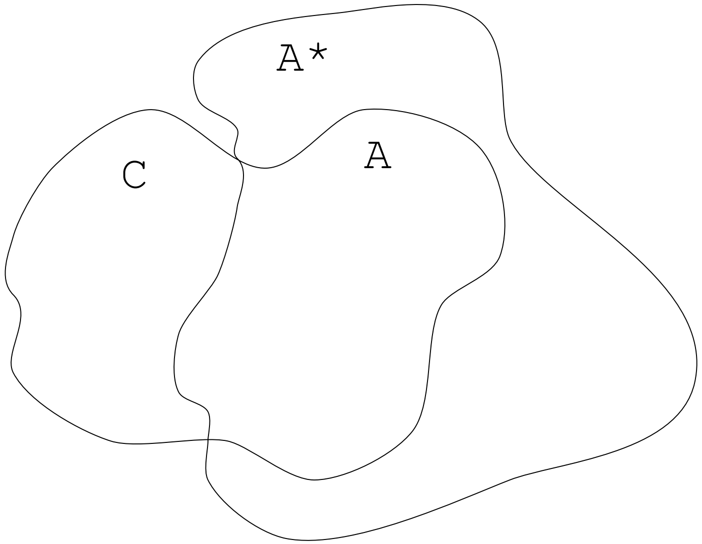
In order to get $\widehat{C}$ (and $\widehat{D}$):

1) Transform the TBox Σ into a new TBox Σ', by replacing every primitive concept definition in Σ of the form $A \sqsubseteq C$ with a concept definition $A \doteq C \sqcap A^*$ – where A^* is a freshly new generated concept name (called *primitive component* of A). Now Σ' contains only (acyclic) concept definitions.

2) Iteratively substitute every occurrence of any defined concept name in C (and D) by the corresponding definition in Σ'. Since Σ' is still acyclic, the process terminates in a finite number of iterations. This process is called *unfolding* or *expansion*.

Theorems

- For each interpretation of Σ there exists an interpretation of Σ' (and viceversa) such that $C^{\mathcal{I}} = C^{\mathcal{I}'}$ for each concept name C in Σ.



$$A \sqsubseteq C \quad \leadsto \quad A \doteq C \sqcap A^*$$

A^* denotes the *unexpressed* part of meaning implicitly contained in the primitive concept definition.

- $\Sigma \models C \sqsubseteq D$ iff $\Sigma' \models C \sqsubseteq D$
- $\Sigma' \models C \sqsubseteq D$ iff $\models \widehat{C} \sqsubseteq \widehat{D}$

Necessary and Sufficient conditions

- A primitive concept definition $A \sqsubseteq C$ states a necessary but not sufficient condition for membership in the class A. Having the property C is necessary for an object in order to be in the class A; however, this condition alone is not sufficient in order to conclude that the object is in the class A.

- A concept definition $A \doteq C$ states necessary and sufficient condition for membership in the class A. Having the property C is necessary for an object in order to be in the class A; moreover, this condition alone is sufficient in order to conclude that the object is in the class A.

Necessary and Sufficient conditions

When transforming primitive concept definitions into concept definitions we get necessary and sufficient conditions for membership in the primitive class A. However, the condition of being in the primitive component A^* can never be satisfied, since the concept name A^* can never be referred to by any other concept.

A concept is subsumed by a primitively defined concept if and only if it refers to its name in its (unfolded) definition.

Inheritance

Unfolding realizes what is usually called *inheritance* in Object-Oriented frameworks.

$$\mathtt{Person} \doteq \exists \mathtt{NAME}.\mathtt{String} \sqcap \exists \mathtt{ADDRESS}.\mathtt{String}$$

$$\mathtt{Parent} \doteq \mathtt{Person} \sqcap \exists \mathtt{CHILD}.\mathtt{Person}$$

$$\widehat{\mathtt{Parent}} \doteq \exists \mathtt{NAME}.\mathtt{String} \sqcap \exists \mathtt{ADDRESS}.\mathtt{String} \sqcap \exists \mathtt{CHILD}.(\exists \mathtt{NAME}.\mathtt{String} \sqcap \exists \mathtt{ADDRESS}.\mathtt{String})$$

$$\mathtt{Female} \doteq \neg \mathtt{Male}$$

$$\mathtt{Man} \doteq \mathtt{Person} \sqcap \forall \mathtt{SEX}.\mathtt{Male}$$

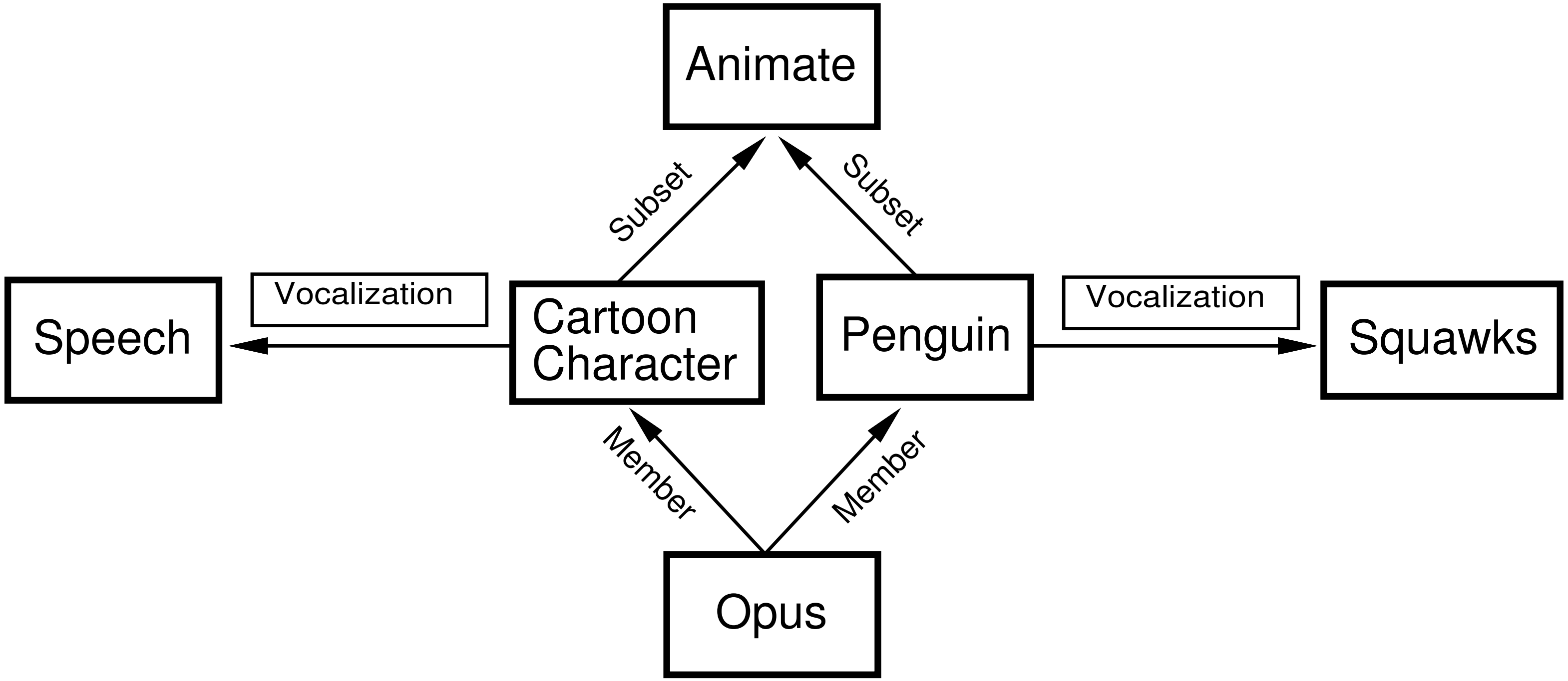
$$\mathtt{Woman} \doteq \mathtt{Person} \sqcap \forall \mathtt{SEX}.\mathtt{Female}$$

$$\mathtt{Transexual} \doteq \mathtt{Man} \sqcap \mathtt{Woman}$$

$$\widehat{\mathtt{Transexual}} \doteq \exists \mathtt{NAME}.\mathtt{String} \sqcap \exists \mathtt{ADDRESS}.\mathtt{String} \sqcap \forall \mathtt{SEX}.\bot$$

Inheritance in O-O

Problems in O-O frameworks: overriding strategies for multiple inheritance.



Complexity of Unfolding

$$C_1 \doteq \forall R_1.C_0 \sqcap \forall R_2.C_0 \sqcap \ldots \sqcap \forall R_m.C_0$$

$$C_2 \doteq \forall R_1.C_1 \sqcap \forall R_2.C_1 \sqcap \ldots \sqcap \forall R_m.C_1$$

$$\ldots$$

$$C_n \doteq \forall R_1.C_{n-1} \sqcap \forall R_2.C_{n-1} \sqcap \ldots \sqcap \forall R_m.C_{n-1}$$

- The size of the TBox is $\mathcal{O}(n \times m)$.
- The size of the unfolded concept $\widehat{C_n}$ is $\mathcal{O}(m^n)$.
- The complexity of the subsumtpion problem in $\mathcal{FL}^-$ with empty TBox ($\models C \sqsubseteq D$) is P.
- The complexity of the subsumtpion problem in $\mathcal{FL}^-$ with an acyclic simple TBox ($\Sigma \models C \sqsubseteq D$) is co-NP-complete.

Efficiency of Subsumption in practice

The exponential worst case is unlikely to occurr in real knowledge bases.

- Let n be the *depth* of a TBox, i.e., the max number of iterations while unfolding every concept definition.
- Let m be the size of the largest definition.
- Let s be the size of the TBox, i.e., m times the number of concept definitions.

The size of an unfolded concept is $\mathcal{O}(m^n)$.

If $n \leq \log_m s$ the size of an unfolded concept becomes polynomial $\mathcal{O}(s)$ with respect to the size of the TBox.

This is a reasonable assumption, since the depth of concept definitions is usually much smaller than the size of the knowledge base. This is why systems behave well in practice.

Definitions

- Definitions are intended to provide an exact account for the concept name being defined.
- Given an initial interpretation of the primitive concept names there exists a unique way determine the interpretation of defined concept names; indeed, that's why they are called *definitions*.
- This justifies the correctness of unfolding: we can always replace a concept name with its definition, since it doesn't add anything to the theory.
- However, if the (simple) TBox is cyclic, this is no more true.

Example of recursive definition

$$\texttt{Bird} \doteq \texttt{Animal} \sqcap \forall \texttt{SKIN}.\texttt{Feather}$$

$$\Delta^{\mathcal{I}} = \{\texttt{tweety}, \texttt{goofy}, \texttt{fea1}, \texttt{fur1}\}$$
$$\texttt{Animal}^{\mathcal{I}} = \{\texttt{tweety}, \texttt{goofy}\}$$
$$\texttt{Feather}^{\mathcal{I}} = \{\texttt{fea1}\}$$
$$\texttt{SKIN}^{\mathcal{I}} = \{\langle\texttt{tweety}, \texttt{fea1}\rangle, \langle\texttt{goofy}, \texttt{fur1}\rangle\}$$
$$\Longrightarrow \quad \texttt{Bird}^{\mathcal{I}} = \{\texttt{tweety}\}$$

$$\texttt{Quiet}-\texttt{Person} \doteq \texttt{Person} \sqcap \forall \texttt{FRIEND}.\texttt{Quiet}-\texttt{Person}$$

$$\Delta^{\mathcal{I}} = \{\texttt{john}, \texttt{sue}, \texttt{andrea}, \texttt{bill}\}$$
$$\texttt{Person}^{\mathcal{I}} = \{\texttt{john}, \texttt{sue}, \texttt{andrea}, \texttt{bill}\}$$
$$\texttt{FRIEND}^{\mathcal{I}} = \{\langle\texttt{john}, \texttt{sue}\rangle, \langle\texttt{andrea}, \texttt{bill}\rangle, \langle\texttt{bill}, \texttt{bill}\rangle\}$$
$$\Longrightarrow \quad \texttt{Quiet}-\texttt{Person}^{\mathcal{I}} = \{\texttt{john}, \texttt{sue}\}$$
$$\Longrightarrow \quad \texttt{Quiet}-\texttt{Person}^{\mathcal{I}} = \{\texttt{john}, \texttt{sue}, \texttt{andrea}, \texttt{bill}\}$$

Descriptive semantics

(It is the one we have introduced before.)

- An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* the concept definition $A \doteq C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$.

- The definition is a *constraint* stating a restriction on the valid models of the knowledge base, and in particular on the possible interpretations of A, where A is no better specified.

- It allows both models of the previous cyclic definition.

Fixpoint Semantics

We associate to a cyclic concept definition an operator $F : 2^{\Delta^{\mathcal{I}}} \mapsto 2^{\Delta^{\mathcal{I}}}$, such that the interpretation of A correspond to the fixpoints of the operator F.

Thus, we associate the equation

$$A = F(A)$$

to a cyclic concept definition of the type

$$A \doteq C$$

where C mentions A.

Least Fixpoint Semantics

The LFS interprets a recursive definition

$$A = F(A)$$

by assigning to A the *smallest* possible extension in each interpretation $\mathcal{I}$ – if it exists – among those satisfying

$$A^{\mathcal{I}} = F(A)^{\mathcal{I}}$$

i.e., the least fixpoint of the corresponding operator.

If the operator is monotonic, then the equation above singles out a *unique* interpretation (subset of $\Delta^{\mathcal{I}}$), hence it *defines* the concept A.

Example

$$\mathtt{Quiet{-}Person} \doteq \mathtt{Person} \sqcap \forall \mathtt{FRIEND}.\mathtt{Quiet{-}Person}$$

$$F \;=\; \lambda A.\{\mathtt{x} \in \Delta^{\mathcal{I}} \mid \mathtt{Person}^{\mathcal{I}}(\mathtt{x}) \wedge \forall \mathtt{y}.\, \mathtt{FRIEND}^{\mathcal{I}}(\mathtt{x},\mathtt{y}) \rightarrow A(\mathtt{y})\}$$

$$A = F(A)$$

$$\Delta^{\mathcal{I}} = \{\mathtt{john}, \mathtt{sue}, \mathtt{andrea}, \mathtt{bill}\}$$
$$\mathtt{Person}^{\mathcal{I}} = \{\mathtt{john}, \mathtt{sue}, \mathtt{andrea}, \mathtt{bill}\}$$
$$\mathtt{FRIEND}^{\mathcal{I}} = \{\langle \mathtt{john}, \mathtt{sue}\rangle, \langle \mathtt{andrea}, \mathtt{bill}\rangle, \langle \mathtt{bill}, \mathtt{bill}\rangle\}$$

$$\Longrightarrow \quad \mathtt{Quiet{-}Person}^{\mathcal{I}} = \{\mathtt{john}, \mathtt{sue}\}$$

Problems

$\mathtt{Human} \doteq \mathtt{Mammal} \sqcap \exists \mathtt{PARENT} \sqcap \forall \mathtt{PARENT}.\mathtt{Human}$

$\mathtt{Horse} \doteq \mathtt{Mammal} \sqcap \exists \mathtt{PARENT} \sqcap \forall \mathtt{PARENT}.\mathtt{Horse}$

Under the fixpoint semantics

$$\mathtt{Human} \equiv \mathtt{Horse}$$

i.e., in any interpretation $\mathcal{I}$ satisfying the above definitions

$$\mathtt{Human}^{\mathcal{I}} = \mathtt{Horse}^{\mathcal{I}}$$

Inductive definitions

- An Empty-List is a List.
- A Node, that has exactly one SUCCESSOR that is a List, is a List.
- Nothing else is a LIST.

$$\mathtt{Node} \doteq \neg\mathtt{Empty{-}List}$$

$$\mathtt{List} \doteq \mathtt{Empty{-}List} \sqcup (\mathtt{Node} \sqcap \leqslant 1\mathtt{SUCCESSOR} \sqcap \exists\mathtt{SUCCESSOR.List})$$

$$\Delta^{\mathcal{I}} = \{\mathsf{a}, \mathsf{b}, \mathtt{nil}\}$$

$$\mathtt{Node}^{\mathcal{I}} = \{\mathsf{a}, \mathsf{b}\}$$

$$\mathtt{Empty{-}List}^{\mathcal{I}} = \{\mathtt{nil}\}$$

$$\mathtt{SUCCESSOR}^{\mathcal{I}} = \{\langle \mathsf{a}, \mathtt{nil}\rangle, \langle \mathsf{b}, \mathsf{b}\rangle\}$$

With descriptive semantics: $\mathtt{List}^{\mathcal{I}} = \{\mathsf{a}, \mathsf{b}, \mathtt{nil}\}$

With least fixpoint semantics: $\mathtt{List}^{\mathcal{I}} = \{\mathsf{a}, \mathtt{nil}\}$

Inductive definitions

- Compare with Logic Programming, where inductive definitions come for free.
- Descriptive semantics is expressible in FOL.
- Least fixpoint semantics (and inductive definitions) go beyond First Order.

Free TBox

(3) $C \sqsubseteq D$ Concept inclusion

(4) $C \doteq D$ Concept equation

(There is no syntactic constraint on the left hand side of the axiom).

Concept inclusions make sense only with descriptive semantics – we will ignore here the extensions of Descritpion Logics where it is possible to specify explicitly the semantics to be given to a knowledge base.

Theorem:

Descritpion logics with simple TBoxes (with general concept definitions $A \doteq D$) and free TBoxes (with concept inclusions $C \sqsubseteq D$) have the same expressive power.

Simple TBoxes and Free TBoxes

Satisfiability in a knowledge base Σ with a free TBox can be reduced into satisfiability in a knowledge base Σ' with a simple TBox.

$$C_i \doteq D_i \quad \rightsquigarrow \quad A_i \sqsubseteq D_i,\ C_i \sqsubseteq A_i$$

$$C_j \sqsubseteq D_j \quad \rightsquigarrow \quad \begin{array}{l} A \doteq (\neg C_1 \sqcup D_1) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n)\ \sqcap \\ \qquad A^* \sqcap \forall R_1.A \sqcap \cdots \sqcap \forall R_m.A \end{array}$$

where A is a new concept name not appearing in Σ and R_i are all the role names appearing in Σ.

The process of eliminating general axioms is called *internalization*. The above simple TBox emulates the general axiom

$$(\neg C_1 \sqcup D_1) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n) \doteq \top$$