Introduction to Logic in Computer Science: Autumn 2007

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Description Logics

Description logics (DLs) have been developed in the late 1980s and early 1990s to provide sound logical foundations for some of the semi-formal knowledge representation (KR) languages developed in AI. Nowadays one of the main applications is the Semantic Web.

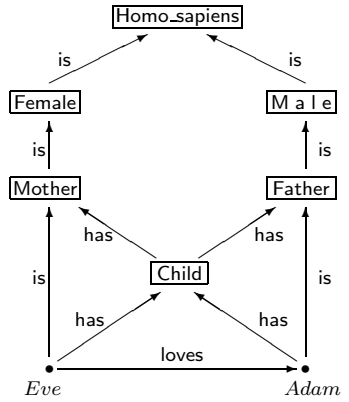
Today we will

- introduce the basic DL $\mathcal{ALC}$ and some of its variants;
- discuss connections to modal and to first-order logic;
- discuss typical inference problems and their complexity; and
- present some tableau-based decision procedures for DLs.

Much of the material on these slides is taken from the handbook chapter by Baader and Nutt (2003).

F. Baader and W. Nutt. Basic Description Logics. In *The Description Logic Handbook*. Cambridge University Press, 2003.

Knowledge Representation



We would like to be able to ask a computer questions regarding this knowledge.

But what is the precise meaning of this "semantic network", e.g. what is the difference between *Adam* and Male or what is the difference between a loves- and an is-arrow?

Picture credits: D. Gabbay *et al.*, *Many-Dimensional Modal Logics*. Elsevier, 2003.

Formalising Semantic Networks

It seems useful to distinguish the following:

- Specific *objects* (or *individuals*) such as *Adam* and *Eve*.
- *Classes* or *sets* of objects, such as Female (later called *concepts*).
- The use of is as a *subset-relation* between concepts (like between Mother and Female).
- The use of is as a *membership-relation* between individuals and concepts (like between *Adam* and Father).
- Other binary *relations* (which we will call *roles*) between individuals, such as loves between *Eve* and *Adam*.
- The use of the same kinds of *relations (roles) between individuals and concepts* (e.g. has). In that case we may want to distinguish (at least) whether the relation is meant to hold for *all* of the individuals belonging to the concept or just for *some* of them.

Description Logic Philosophy

- Develop logics with features that seem useful for KR.
- Be modular: include or exclude various features, develop reasoning algorithms for all variations, understand the complexity of all variations.
- Stay decidable (so don't use FOL).
- Distinguish *terminological* and *assertional* knowledge:
  - TBox: knowledge about *concepts* and how they relate to each other
  - ABox: knowledge about *individuals*, to what concepts they belong, how they relate to each other

  We will first focus on languages for concept descriptions ...

The Language $\mathcal{AL}$

One of the most basic DLs is $\mathcal{AL}$ ("attributive language").

Let A stand for *atomic concepts* and R for *atomic roles*.

Syntax for *concept* descriptions C in the basic language $\mathcal{AL}$:

$$C \quad ::= \quad \top \mid \bot \mid A \mid \neg A \mid C \sqcap C \mid \forall R.C \mid \exists R.\top$$

Note that negation is restricted to atomic concepts and existential quantification is limited. The universal quantification is also known as *value restriction*. Examples:

- $\texttt{Person} \sqcap \neg\texttt{Female}$
- $\texttt{Person} \sqcap \exists\texttt{hasChild}.\top \sqcap \forall\texttt{hasChild}.(\texttt{Female} \sqcap \texttt{Student})$
- $\texttt{Person} \sqcap \forall\texttt{hasChild}.\bot$

Semantics of $\mathcal{AL}$

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a *domain* $\Delta^\mathcal{I}$ and an *interpretation function* mapping atomic concepts A to sets $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and atomic roles R to binary relations $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.
And:

$$\begin{aligned}
\top^\mathcal{I} &= \Delta^\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset \\
(\neg A)^\mathcal{I} &= \Delta^\mathcal{I} \setminus A^\mathcal{I} \\
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(\forall R.C)^\mathcal{I} &= \{a \in \Delta^\mathcal{I} \mid \{b \in \Delta^\mathcal{I} \mid (a,b) \in R^\mathcal{I}\} \subseteq C^\mathcal{I}\} \\
(\exists R.\top)^\mathcal{I} &= \{a \in \Delta^\mathcal{I} \mid \{b \in \Delta^\mathcal{I} \mid (a,b) \in R^\mathcal{I}\} \neq \emptyset\}
\end{aligned}$$

Extensions of $\mathcal{AL}$

$\mathcal{AL}$ may be extended with any of the following:

- Union of concepts: $C \sqcup D$ with $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$
- Full negation: $\neg C$ with $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$
- Full existential quantification: $\exists R.C$ with $(\exists R.C)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \{b \in \Delta^\mathcal{I} \mid (a,b) \in R^\mathcal{I}\} \cap C^\mathcal{I} \neq \emptyset\}$
- Number restrictions (*at-least*): $\geq n\, R$ with $(\geq n\, R)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \#\{b \in \Delta^\mathcal{I} \mid (a,b) \in R^\mathcal{I}\} \geq n\}$
- Number restrictions (*at-most*): $\leq n\, R$ with $(\leq n\, R)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \#\{b \in \Delta^\mathcal{I} \mid (a,b) \in R^\mathcal{I}\} \leq n\}$

Naming Conventions

To name an extension of $\mathcal{AL}$ add the letters corresponding to the additional constructs allowed (omitting redundant extensions):

- $\mathcal{U}$ for concept unions
- $\mathcal{C}$ for full negation (*c*omplements)
- $\mathcal{E}$ for full existential quantification
- $\mathcal{N}$ for number restrictions

But the literature knows many more description logics: e.g. $\mathcal{EL}$, $\mathcal{FL}_0$, $\mathcal{FL}^-$, $\mathcal{ALCRI}$, $\mathcal{ALCH}$, $\mathcal{DLR}$, $\mathcal{SIN}$, $\mathcal{SHIF}$, $\mathcal{SHIQ}$, $\mathcal{SHOIQ}$, $\mathcal{SHOQ}(D)$ ... as expected, this is the subject of some occasional ridicule directed at DL researchers. :-)

We will largely concentrate on $\mathcal{ALC}$: $\mathcal{AL}$ with full negation, which also buys us concept unions and full existential quantification.

$\mathcal{ALC}$ and Modal Logic

$\mathcal{ALC}$ is equivalent to the modal logic $\mathbf{K}_n$ (multi-modal $\mathbf{K}$):

$$\begin{aligned} \forall R_i.\varphi &\sim \Box_i\varphi \\ \exists R_i.\varphi &\sim \Diamond_i\varphi \end{aligned}$$

Standard Translation

We can translate $\mathcal{ALC}$ concepts into formulas of classical first-order logic with a single free variable x.

We translate each *atomic concept* A as a unary predicate A' and each *atomic role* R as a binary predicate R'.

The *standard translation* $\cdot^*$ of concepts is defined inductively:

$$\begin{aligned} A^* &= A'(x) \\ (\neg C)^* &= \neg C^* \\ (C \sqcap D)^* &= C^* \wedge D^* \\ (C \sqcup D)^* &= C^* \vee D^* \\ (\forall R.C)^* &= (\forall y)(R'(x,y) \rightarrow C^*[y/x]) \\ (\exists R.C)^* &= (\exists y)(R'(x,y) \wedge C^*[y/x]) \end{aligned}$$

Here y is meant to be a fresh variable symbol each time it is needed.

$\mathcal{ALC}$ and the Two-Variable Fragment of FOL

By renaming bound variables in a clever way, we can translate any $\mathcal{ALC}$ concept description into a FOL formula using only two variable symbols (x and y). Example:

$$\begin{aligned} & (\forall R_1.(C \sqcap \exists R_2.D))^* \\ = \; & (\forall y)[R_1'(x,y) \rightarrow C'(y) \wedge (\exists z)(R_2'(y,z) \wedge D'(z))] \\ = \; & (\forall y)[R_1'(x,y) \rightarrow C'(y) \wedge (\exists x)(R_2'(y,x) \wedge D'(x))] \end{aligned}$$

The *two-variable fragment* of FOL (w/o proper function symbols) is known to be *decidable* an **NEXPTIME**-complete.

This gives us a first (fairly unattractive) upper complexity bound (we'll do better later on).

TBox: Terminological Knowledge

The TBox is used to collect knowledge on concepts.

Syntax: A TBox is a list of concept equalities ($C \equiv D$).
If C is atomic, then $C \equiv D$ may be considered a concept definition.

Examples:

- $\texttt{Woman} \equiv \texttt{Person} \sqcap \texttt{Female}$
- $\texttt{Mother} \equiv \texttt{Woman} \sqcap \exists \texttt{hasChild}.\texttt{Person}$
- $\texttt{Person} \equiv \exists \texttt{hasParent}.\texttt{Person}$

Acyclic TBoxes without multiple definitions are of particular interest (but not only those).

Semantics: An interpretation $\mathcal{I}$ is a model for $C \equiv D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$, and accordingly for an entire TBox.

ABox: Assertional Knowledge

The ABox is used to specify knowledge regarding individuals.

Syntax: An ABox is a list of concept assertions and role assertions.

$$a : C \qquad (a, b) : R$$

Examples:

- $\texttt{alice} : \texttt{Mother}$
- $\texttt{bob} : \neg\texttt{Father} \sqcap \forall \texttt{hasParent}.\texttt{Lawyer}$
- $(\texttt{alice}, \texttt{bob}) : \texttt{hasChild}$

We make the *unique name assumption* (UNA): all individuals in an ABox are pairwise distinct (not really important for $\mathcal{ALC}$).

Semantics: Extend $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ to individuals ($a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$).
$\mathcal{I}$ is a model for $a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and for $(a, b) : R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, and accordingly for an entire ABox (or an ABox wrt. a TBox).

Common Reasoning Tasks

We would like to be able to perform the following types of reasoning tasks in a KR system:

- Concept Satisfiability: given a concept description C, is there an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$?
- Concept Subsumption: given two concept descriptions C and D, is it the case that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $\mathcal{I}$? We may also be interested in the full subsumption *hierarchy* of a set of concepts.
- ABox Consistency: does the given ABox have a model?
- Instance Checking: given an ABox, an individual a, and a concept C, is $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of the ABox?

All of the above can also be defined with respect to a given TBox.

Reduction of Reasoning Tasks

All of the aforementioned reasoning tasks (w/o TBox) can be reduced to ABox consistency checking:

- Satisfiability: take ABox $\{a : C\}$ and show consistency
- Subsumption: take ABox $\{a : C \sqcap \neg D\}$ and show inconsistency
- Instance Checking: add $a : \neg C$ to ABox and show inconsistency

For reasoning with respect to a TBox that is *acyclic* and has *unique definitions*, we can simply *unfold* the defined concepts into the ABox (but note that this can worsen complexity!).

Reasoning with respect to general TBoxes is more difficult.

Tableaux for $\mathcal{ALC}$ ABox Consistency Checking

Input: An $\mathcal{ALC}$ ABox to be checked for consistency. For simplicity, assume all concept descriptions have been translated into NNF.

Rules:

$$\frac{a : C \sqcap D}{\begin{array}{c} a : C \\ a : D \end{array}} \qquad \frac{a : C \sqcup D}{a : C \mid a : D} \qquad \frac{a : \forall R.C \quad (a,b) : R}{b : C} \qquad \frac{a : \exists R.C}{(a,b) : R \quad b : C} \qquad \frac{a : C \quad a : \neg C}{\times}$$

Note: $\forall$-formulas need to be analysed for every b with $(a, b) : R$ on the branch. The b in the $\exists$-rule is a *new* individual name.
The $\exists$-rule need not be applied if there already is a witness.

Soundness, Completeness, Termination

The tableau algorithm on the previous slide gives us a decision procedure for checking the consistency of an ABox in $\mathcal{ALC}$:

- Soundness: easy
- Completeness: similar to what we have seen earlier on
- Termination: First observe that any concept formula added to a branch must be a subformula of a formula appearing in the input ABox.

  So the only potential problem could be that an infinite number of individuals get generated (through some interplay of $\forall$ and $\exists$). But this cannot happen: the "further away" from one of the original individuals we get, the shorter the concept formulas, so at some point there will be no more $\exists$-formulas that could generate new individuals. ✓

Complexity

Consistency checking of $\mathcal{ALC}$-ABox is **PSPACE**-complete. Here are some pointers towards a proof:

- **PSPACE**-membership: While the size of the tableau may be exponential, each branch will be linear wrt. the size of the input. If we use a non-deterministic algorithm, we only need to store one such branch. This gives an **NPSPACE** algorithm. By Savitch's Theorem, we get membership in **PSPACE**.
- **PSPACE**-hardness: By reduction from the satisfiability problem for Quantified Boolean Formulas.

Reasoning wrt. a TBox

We now want to extend our tableau algorithm to be able to check consistency of an ABox wrt. a TBox. The basic algorithm will be the same, but now we have a problem with termination.

Example:

- TBox includes $\top \equiv \neg\texttt{Person} \sqcup \exists\texttt{hasParent}.\texttt{Person}$.
- ABox includes $\texttt{mary} : \texttt{Person}$.
- Basically, we have to label each newly generated individual with $\neg\texttt{Person} \sqcup \exists\texttt{hasParent}.\texttt{Person}$, which results in an infinite `hasParent`-chain.
- This does correspond to the intuitive semantics (so is certainly sound), but we lose termination.

Tableaux for ABox Consistency wrt. a TBox

We want to check whether a given ABox is consistent wrt. a given TBox, i.e. whether there exists a model satisfying both of them.

Preparation: Suppose the TBox $= \{C_1 \equiv D_1, \ldots, C_n \equiv D_n\}$.
Define $\hat{C} = (\neg C_1 \sqcup D_1) \sqcap (C_1 \sqcup \neg D_1) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n) \sqcap (C_n \sqcup \neg D_n)$.
Note that every individual should belong to this concept $\hat{C}$.

Naïve algorithm: Use exactly the same algorithm as before, but add $a : \hat{C}$ to every branch for every individual a (from the input or generated along the way). This works, except for the termination problem.

To get a terminating algorithm we can use the *blocking* technique ...

Blocking

The application of the $\exists$-rule to a formula labelled by individual a is *blocked* by individual b (on branch $\mathcal{A}$) if this condition holds:

$$\{C \mid [a : C] \in \mathcal{A}\} \quad \subseteq \quad \{D \mid [b : D] \in \mathcal{A}\}$$

That is, b labels the same set of concepts as a (and possibly more).

Then anything that the current branch could force upon role successors of a would also be present in the role successors of b. So we can use the successors of b in place of generating new ones for a.

To avoid cyclic blocking (of a by b and b by a, for instance), we impose an (arbitrary) ordering on individuals, and only allow blocking of a by individuals b further down that ordering.

Some careful thinking reveals that blocking will ensure termination without affecting completeness.

Remark: Similar techniques can be used to handle *transitive roles*.

Conclusion

- Description logic have been developed to formlaise early AI approaches to knowledge representation.
- We have discussed the following topics:
  - Family of logics extending $\mathcal{AL}$, including $\mathcal{ALC}$
  - Connections to modal logic and FOL (standard translation)
  - TBox, ABox, common reasoning tasks, complexity issues
  - Tableaux algorithms for $\mathcal{ALC}$ (with blocking for TBox-reasoning)
- Very well understood family of logics with real real-world applications (e.g. Semantic Web) as well as strong impact on research in neighbouring areas of logic (e.g. modal and hybrid logic).
- In DL, tableaux are the predominat reasoning algorithms (unlike for FOL, where resolution is most widely used).
- Testbed for the development of efficient decsion algorithms for problems of very high complexity (**PSPACE** and higher).