Declarative Knowledge Processing
Lecture 3: ALC and some of its extensions

Magdalena Ortiz
Knowledge Base Systems Group
Institute of Information Systems
ortiz@kr.tuwien.ac.at
25 October 2010

Outline

1. Some extensions of ALC
2. Reasoning Problems
3. DLs and other logics
   3.1 DLs and FOL
   3.2 DLs and Modal Logics
4. Summary

Reasoning around ALC
1. Some extensions of ALC

Reminder - the DL ALC

- Our vocabulary contains three sets $N_C$ of concept names, $N_R$ of role names, and $N_I$ of individuals.
- Every role name $R \in N_R$ is an ALC role.
- ALC concepts $C_1, C_2$ are defined inductively as follows:
  - $C_1, C_2 \rightarrow A \mid \top \mid \bot \mid \neg C_1 \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C$
  - where $A \in N_C$ is a concept name and $R \in N_R$ is a role.
- An ALC Knowledge Base is a pair $\mathcal{K} = (T, A)$ where:
  - The TBox $T$ is a finite set of GCIs $C_1 \sqsubseteq C_2$, and
  - The ABox $A$ is a finite set of (concept and role) membership assertions $C(a)$, $R(a, b)$

ALC Semantics

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ has a non-empty domain $\Delta^\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$ that maps:

- each concept name $A$ to a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$,
- each role name $R$ to a set of pairs $A^\mathcal{I}$ from $\Delta^\mathcal{I}$, and
- each individual $a$ to an element $a^\mathcal{I}$,

and it is extended to all concepts as follows:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>top/verum</td>
<td>$\top$</td>
<td>$\top^\mathcal{I} = \Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>bottom/falsum</td>
<td>$\bot$</td>
<td>$\bot^\mathcal{I} = \emptyset$</td>
</tr>
<tr>
<td>negation</td>
<td>$\neg C$</td>
<td>$\Delta^\mathcal{I} \setminus C^\mathcal{I}$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C_1 \sqcap C_2$</td>
<td>$C_1^\mathcal{I} \cap C_2^\mathcal{I}$</td>
</tr>
<tr>
<td>disjunction</td>
<td>$C_1 \sqcup C_2$</td>
<td>$C_1^\mathcal{I} \cup C_2^\mathcal{I}$</td>
</tr>
<tr>
<td>universal rest.</td>
<td>$\forall R.C$</td>
<td>${d_1 \mid \forall d_2 \in \Delta^\mathcal{I}.((d_1, d_2) \in R^\mathcal{I} \rightarrow d_2 \in C^\mathcal{I})}$</td>
</tr>
<tr>
<td>existential rest.</td>
<td>$\exists R.C$</td>
<td>${d_1 \mid \exists d_2 \in \Delta^\mathcal{I}.((d_1, d_2) \in R^\mathcal{I} \land d_2 \in C^\mathcal{I})}$</td>
</tr>
</tbody>
</table>
**Reasoning around ALC**

1. Some extensions of ALC

**ALC Semantics (cont’d)**

For an interpretation $I$, we say that

- $I$ satisfies a concept $C$ if $C^I \neq \emptyset$
- $I$ satisfies a GCI $C_1 \sqsubseteq C_2$ if $C_1^I \subseteq C_2^I$
- $I$ satisfies a TBox $T$ if it satisfies every GCI in $T$
- $I$ satisfies a concept membership assertion $C(a)$ if $a^I \in C^I$
- $I$ satisfies a role membership assertion $R(a, b)$ if $(a^I, b^I) \in R^I$

**ALC Semantics (cont’d)**

Definition: Model of a concept

An interpretation $I$ is called a model of a concept $C$ if $C^I \neq \emptyset$. We write $I \models C$.

Definition: Model of a KB

An interpretation $I$ is called a model of a knowledge base $(T, A)$ if it satisfies the TBox $T$ and the ABox $A$. We write $I \models \mathcal{K}$.

Now we will introduce some other DLs that are extensions of ALC.

**Adding Inverse Roles**

- **Inverse roles** are the most popular DL role constructor

**Inverse Roles**

**Syntax**

If $R \in \mathbb{N}_R$ is role name, then $R$ and $R^-$ are roles

**Semantics**

$(R^-)^I = \{(d_2, d_1) \mid (d_1, d_2) \in R^I\}$

- What is the intuitive meaning of the following relations?
  - hasParent
  - isMarried
  - teaches
  - likes

- To indicate that a logic supports inverse roles, we add the letter $I$ to its name
  - In particular, $\text{ALCI}$ denotes the extension of $\text{ALC}$ with inverse roles

**Adding Number Restrictions**

- **Counting and numerical constraints** are important for KR& R

  For example, a hand has at most five fingers
  - A person has exactly two (biological) parents
  - A person can be married to at most one other person
  - An employee may not work for more than three projects

  Most popular DLs support some form of counting

- **Qualified number restrictions** (QNRs) are a popular concept constructor

  **Qualified number restrictions**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R \geq n$ if $C$ is a concept, $R$ is a role and $n$ is a natural number, then $\geq n \ R.C$ and $\leq n \ R.C$ are concepts</td>
<td></td>
</tr>
<tr>
<td>$(\geq n \ R.C)^I = {d_1 \mid {(d_2 \mid (d_1, d_2) \in R^I \land d_2 \in C^I)} \geq n}$</td>
<td></td>
</tr>
<tr>
<td>$(\leq n \ R.C)^I = {d_1 \mid {(d_2 \mid (d_1, d_2) \in R^I \land d_2 \in C^I)} \leq n}$</td>
<td></td>
</tr>
</tbody>
</table>
Adding Number Restrictions (cont’d)

- Example concepts:
  - At most five fingers  \( \leq 5 \) hasPart.Finger
  - Married to at most one other person  \( \leq 1 \) marriedTo.Person
  - Work for at least four projects  \( \geq 4 \) worksFor.Project
  - Exactly two parents  \( \geq 2 \) hasParent.Person \( \cap \leq 2 \) hasParent.Person

- \( Q \) indicates the presence of this constructor in a DL
  - In particular, \( ALCQ \) and \( ALCIQ \) (or \( ALCQI \)) respectively denote the extensions of \( ALC \) and \( ALCI \) with QNRs

- There are other (less general) concept constructors for counting, like:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>( \geq n ) ( R )</td>
<td>( (\geq n ) ( R ))^t = (\geq n ) ( R ) ( T )^t</td>
</tr>
<tr>
<td>( F )</td>
<td>( \leq n ) ( R )</td>
<td>( (\leq n ) ( R ))^t = (\leq n ) ( R ) ( T )^t</td>
</tr>
</tbody>
</table>

Plan

For \( ALC \), \( ALCI \), \( ALCQ \) and \( ALCIQ \), we will discuss

- Reasoning Problems
- Relationship to FOL and modal logic
- Reasoning algorithms
- Complexity of reasoning

Afterwards, we will review a few other prominent extensions of \( ALC \)
Simple Examples of (Un)satisfiability

- Adult $\sqcap$ Male is satisfiable
- $\forall$ hasChild.(Adult $\sqcap$ Male) $\sqcap$ $\forall$ hasChild.(Rich) is satisfiable
- $\forall$ hasChild.(Adult $\sqcap$ Male) $\sqcap$ $\exists$ hasChild.$(-$Adult$)$ is unsatisfiable
- $\exists$ hasChild.Adult $\sqcap$ $\exists$ hasChild.$(-$Adult$)$ is satisfiable
- $\forall$ hasChild.Adult $\sqcap$ $\forall$ hasChild.$(-$Adult$)$ is satisfiable
  (satisfied in interpretations with $\text{hasChild}^I = \{\})$

Examples of Subsumption $\models C \sqsubseteq D$

- $\models$ Adult $\sqcap$ Male $\sqsubseteq$ Adult
- $\models$ $\forall$ hasChild.(Adult $\sqcap$ Male) $\sqsubseteq$ $\forall$ hasChild.Adult
  $C^I = \{d \in \Delta^I \mid \forall d'(d, d') \in \text{hasChild}^I \rightarrow d' \in \text{Adult}^I \sqcap \text{Male}^I\}$
  $D^I = \{d \in \Delta^I \mid \forall d'(d, d') \in \text{hasChild}^I \rightarrow d' \in \text{Adult}^I\}$
  $\models C^I \sqsubseteq D^I$

- $\not\models$ $\exists$ hasChild.Adult $\sqsubseteq$ $\forall$ hasChild.Adult
  $C^I = \{d \in \Delta^I \mid \exists d'(d, d') \in \text{hasChild}^I \land d' \in \text{Adult}^I\}$
  $D^I = \{d \in \Delta^I \mid \exists d'(d, d') \in \text{hasChild}^I \rightarrow d' \in \text{Adult}^I\}$
  $\not\models C^I \sqsubseteq D^I$ why?

Interreducibility of reasoning problems

In (every extension of) $\mathcal{ALC}$, we can:

- reduce concept subsumption to unsatisfiability
  $\models C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable

- reduce concept satisfiability to non-subsumption
  $C$ is satisfiable iff $\not\models C \sqsubseteq \bot$

In what follows, we write $\not\models C \sqsubseteq \bot$ to indicate that $C$ is satisfiable.

Hence, to reason about $\mathcal{ALC}$ concepts, we only need an algorithm to decide concept (un)satisfiability.
### Reasoning with KBs

Of course, we are also interested in reasoning about KBs

**Satisfiable (aka consistent) KB**

A KB $K$ is *satisfiable* if it has some model; we write $K \not\vdash \bot$.

**KB satisfiability problem**

| Input:   | KB $K$ |
| Question: | does $K \not\vdash \bot$ hold? |

---

### Reasoning with KBs (cont’d)

### Satisfiable concept w.r.t. a TBox / KB

A concept $C$ is *satisfiable w.r.t. a TBox $T$* if $C^T \not= \emptyset$ in some model of $T$; we write $T \not\models C \subseteq \bot$.

**Concept satisfiability problem w.r.t. a TBox**

| Input:   | Concept $C$, TBox $T$ |
| Question: | does $T \not\models C \subseteq \bot$ hold? |

---

### Satisfiable concept w.r.t. a KB

A concept $C$ is *satisfiable w.r.t. a KB $K$* if $C^K \not= \emptyset$ in some model of $K$; we write $K \not\models C \subseteq \bot$.

**Concept satisfiability problem w.r.t. a KB**

| Input:   | Concept $C$, KB $K$ |
| Question: | does $K \not\models C \subseteq \bot$ hold? |

---

### Reasoning with KBs (cont’d)

### Concept Subsumption w.r.t. a TBox / KB

D subsumes $C$ w.r.t. $T$ ($T \models C \sqsubseteq D$) if $C^T \subseteq D^T$ in every model of $T$.

D subsumes $C$ w.r.t. $K$ ($K \models C \sqsubseteq D$) if $C^K \subseteq D^K$ in every model of $K$.

**Concept subsumption problem w.r.t. a TBox**

| Input: Concepts $C$, $D$, TBox $T$ |
| Question: does $T \models C \sqsubseteq D$ hold? |

**Concept subsumption problem w.r.t. a KB**

| Input: Concepts $C$, $D$, KB $K$ |
| Question: does $K \models C \sqsubseteq D$ hold? |

---

### Reasoning with KBs (cont’d)

### Reasoning about instances in the presence of a full KB

**Instances of concepts and roles in a KB**

- An individual $a$ is an instance of a concept $C$ in a KB $K$ ($K \models C(a)$) if $a$ is an instance of $C$ in every model of $K$.
- A pair of individuals $(a, b)$ is an instance of a role $R$ in a KB $K$ ($K \models R(a, b)$) if $(a, b)$ is an instance of $R$ in every model of $K$.

### Instance checking

- **For a concept**
  - **Input:** Concept $C$, individual $a$, KB $K$
  - **Question:** does $K \models C(a)$ hold?

- **For a role**
  - **Input:** Role $R$, pair of individuals $(a, b)$, KB $K$
  - **Question:** does $K \models R(a, b)$ hold?
Notes on the Reasoning Problems

To think:

- When deciding concept satisfiability and subsumption w.r.t. a KB, what is the role of the ABox?
- Which problems make sense w.r.t. a TBox only?
- And w.r.t. an A-box only?

These are all decision problems

- We will briefly discuss some non-decision problems later
- It is usually possible to reduce a non-decision problems to several instances of a decision problem (more later)
- In practice, this may not always be the most efficient approach
- The notions and techniques form traditional complexity theory that we consider in this course are only defined for decision problems

Reduction of Reasoning Problems to KB (un)Satisfiability

Concept satisfiability and subsumption w.r.t. a TBox can be trivially reduced to the analogous problems w.r.t. a KB

\[
T \not\models C \subseteq D \quad \text{iff} \quad (T, \emptyset) \not\models C \subseteq \bot
\]

\[
T \models C \subseteq D \quad \text{iff} \quad (T, \emptyset) \models C \subseteq D
\]

Now we show that all mentioned problems for reasoning about concepts and instances can be reduced to KB (un)satisfiability:

- Concept satisfiability w.r.t. a KB \( \mathcal{K} \) to KB satisfiability

\[
(T, A) \not\models C \subseteq \bot \quad \text{iff} \quad (T, A \cup \{C(a)\}) \not\models \bot \quad \text{for some } a \text{ not in } A
\]

We have seen that it is enough to have:

- an algorithm for concept satisfiability, to solve the problems that reason with concepts only
- an algorithm for KB satisfiability, to solve the problems that take a TBox or KB into account

It is not hard to see that the latter one suffices for all tasks, as concept satisfiability can be easily reduced to KB satisfiability

\[
\emptyset \not\models C \subseteq \bot \quad \text{iff} \quad (\emptyset, C(a)) \not\models \bot
\]

However, it is not such a good idea to always reduce everything to KB satisfiability, and we will study two reasoning algorithms why?
Some final remarks

For simplicity, when reasoning about a KB \((T, A)\), we may make some assumptions w.l.o.g.:

1. Concept membership assertions in \(A\) are only of the form \(A(a)\), for a concept name \(A\).

Replace each \(C(a) \in A\) with \(A_C(a)\) for a fresh concept name \(A_C\), and add to the TBox \(A_C \sqsubseteq C\).

2. The ABox contains at least one individual.

An empty ABox can be replaced by \(\top(a)\), and answers to all reasoning problems are preserved.

Outline

1. Some extensions of \(ALC\)
2. Reasoning Problems
3. DLs and other logics
   3.1 DLs and FOL
3.2 DLs and Modal Logics
4. Summary

\(ALC\) and FOL

Every \(ALC\) concept \(C\) can be rewritten as a FOL formula \(\varphi_x(C)\) with one free variable \(x\), as follows:

\[
\begin{align*}
\varphi_x(A) & := A(x) & \text{for a concept name } A \\
\varphi_x(\neg C) & := \neg\varphi_x(C) \\
\varphi_x(C_1 \sqcap C_2) & := \varphi_x(C_1) \land \varphi_x(C_2) \\
\varphi_x(C_1 \sqcup C_2) & := \varphi_x(C_1) \lor \varphi_x(C_2) \\
\varphi_x(\forall R.C) & := \forall y.R(x, y) \rightarrow \varphi_y(C) \\
\varphi_x(\exists R.C) & := \exists y.R(x, y) \land \varphi_y(C')
\end{align*}
\]

\(C\) and \(\varphi_x(C)\) have the same extension in every interpretation. That is, for every interpretation \(I\) and concept \(C\), we have

\[
C^I = \{ d \in \Delta^I : I \models \varphi_x(C)[d]\}
\]

\(ALC\) and FOL (cont’d)

- Any given TBox \(T\) is equivalent to a FOL formula conjunction of FOL formulas of the form \(\forall x.\varphi(x) \rightarrow \psi(x)\), which we denote \(T^\#\)

\[
T^\# = \bigwedge_{C \subseteq D \in T} \forall x.\varphi_x(C) \rightarrow \varphi_x(D)
\]

- An ABox \(A\) (with only concept names) can be rewritten as a conjunction \(A^\#\) if simple ground formulas \(A(a)\), \(R(a, b)\)

\[
A^\# = \bigwedge_{A(a) \in A} A(a) \land \bigwedge_{R(a, b) \in A} R(a, b)
\]
**ALC and FOL (cont’d)**

**Theorem.**
For every interpretation $I$, $I \models (T, A)$ if and only if $I \models T^\# \land A^\#$.

- Hence $(T, A)$ is (DL) satisfiable if $T^\# \land A^\#$ is (FOL) satisfiable, and in ALC, all reasoning tasks can be reduced to FOL satisfiability.
- Furthermore, only two variable symbols occur in $T^\# \land A^\#$.
- Since satisfiability in the fragment of FOL that allows for only two variables is known to be decidable, it follows that ALC is decidable.
- However, this does not give us tight complexity bounds and efficient decision algorithms.

**Number restrictions in FOL**

- Since number restrictions are a concept constructor, we only need to extend these concepts our translation $\varphi_x$ that rewrites each concept into a formula with one free variable.
- This is easy to do, but we need equality and more than two variables!
- We are not in the 2-variable fragment anymore, but we are still within a fragment that is known to be decidable.
  - Namely, the extension of the two variable fragment with counting quantifiers of the form $\exists \leq n y$ and $\exists \geq n y$, which have the same semantics as number restrictions.

**ALC and Modal Logic**

- Each ALC concept corresponds to a formula in multi-modal K with a different syntax, and vice-versa.
  - Each concept name corresponds to a propositional variable.
  - Each role name corresponds to an accessibility relation.
  - The connectives $\neg$, $\land$, $\lor$ correspond to $\neg$, $\land$, $\lor$ in ML.
  - $\forall R.C$ corresponds to $\Box R.C$ (also written $[R]C$).
  - $\exists R.C$ corresponds to $\Diamond R.C$ (also written $[R]C$).
- A DL interpretation $I$ is a Kripke structure, and the extension $C^f$ of $C$ coincides with the worlds where the corresponding formula is true. Since satisfiability in multi-modal K is known to be PSpace complete, we obtain:

**Theorem.**
Concept satisfiability and subsumption in ALC are PSpace-complete.
**ALC and Modal Logic (cont’d)**

Many features of DLs are absent from traditional MLs, but are can be expressed or closely resemble some extensions of MLs.

- **Number restrictions** can not be expressed in traditional MLs, but have a precise counterpart in the extensions of MLs that allow for graded modalities: $\langle n \rangle C$, $[n] C$, $\langle n \rangle R C$, $[n] R C$.

- **Inverse roles** are also absent from traditional MLs, but have a counterpart in the MLs with converse modalities: $\langle R \rangle C$, $[R] C$.

- The notions of TBox and ABox do not really have a counterpart in the ML setting.
  - However, they are closely related to the extensions of ML known as hybrid logics.

- Some extensions of MLs have been influenced by DLs, and the exchange of techniques and results between the two fields has been fruitful.

---

**Some of the Main Ideas of Today**

- We have defined the syntax and semantics of the DL **ALC** and some of its extensions: **ALCI**, **ALCQ**, and **ALCIQ**.

- We have discussed in detail the most important reasoning problems of DLs and see how they can be reduced to each other.

- DLs are closely related to other well-known logics, like
  - **first order logic**, of which most DLs are fragments
  - **modal logics**, of which most DLs are extensions/variations

This allows us to derive some decidability and complexity results. As we will see later, one can also build on techniques known from these logics to achieve efficient DL reasoning algorithms.

---

**Links and References**

- **The Description Logic Handbook, Theory, Implementation and Applications**

  Many links, includes some tutorials & courses.

- **Handbook of Modal Logic**
  Patrick Blackburn, Johan van Benthem, and Frank Wolter, eds. Elsevier 2006
  In particular, *Part 3: Extensions* contains:
  - A great chapter on DLs (Ch. 13, by Franz Baader and Carsten Lutz), which discusses in detail the relationship between DLs and MLs
  - Chapters on the mentioned extensions of MLs and other important ‘cousins’ of DLs: Hybrid logics, PDL and modal $\mu$-calculus, Temporal logic, Computational Modal Logic, etc.