Description Logics

Carsten Lutz and Ulrike Sattler

TU Dresden, Germany
Representation of conceptual knowledge is subfield of Artificial Intelligence

Early days of AI: KR through obscure pictures (semantic networks)

Problems: missing semantics (reasoning!), complex pictures
Remedy: Use a logical formalism for KR rather than pictures
Defining elephants using DLs:

- \( \text{Mammal} \sqcap \exists \text{bodypart}. \text{Trunk} \sqcap \forall \text{color}. \text{Grey} \)

- \( \text{Mammal} \sqcap \exists \text{bodypart}. \text{Trunk} \sqcap \left( = 1 \text{ color} \right) \sqcap \forall \text{color}. \text{Grey} \)
  \( \sqcap \left( = 1 \text{ weight} \right) \sqcap \left( \forall \text{weight}. \text{Heavy} \sqcup \left( \text{Dumbo} \sqcap \forall \text{weight}. \text{Light} \right) \right) \)

A concept language does not solve all problems...

- Do these concepts describe necessary or sufficient conditions?
- How can we describe specific elephants such as Dumbo?
- How do I avoid losing track when constructing large knowledge bases?
Foci of “modern” DL research:

1. Identify interesting Description Logics and study their properties
   Main topics: decidability, computational complexity, expressivity

2. Implement Description Logics in highly-optimized reasoning systems
   Fast and powerful systems available: e.g. FaCT and RACER

3. Apply Description Logics in several application areas
   - Reasoning about Entity Relationship (ER) diagrams
   - Representation of Ontologies for the Semantic Web
**The Description Logic $\mathcal{ALC}$: Syntax**

$\mathcal{ALC}$ is the smallest propositionally closed Description Logic.

**Atomic types:** concept names $A, B, \ldots$ (unary predicates)
role names $R, S, \ldots$ (binary predicates)

**Constructors:**
- $\neg C$ (negation)
- $C \cap D$ (conjunction)
- $C \cup D$ (disjunction)
- $\exists R.C$ (existential restriction)
- $\forall R.C$ (universal restriction)

**For example:**

$\neg (A \cup \exists R. (\forall S.B \cap \neg A))$

Mammal $\cap \exists$bodypart.trunk $\cap \forall$color.Grey
Semantics of $\mathcal{ALC}$

Semantics based on interpretations $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\cdot^\mathcal{I}$ maps

- each concept name $A$ to a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$.
- each role name $R$ to a binary relation $R^\mathcal{I}$ over $\Delta^\mathcal{I}$.

Semantics of complex concepts:

\[ (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I} \quad (C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I} \quad (C \cup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I} \]

\[ (\exists R.C)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid (d, e) \in R^\mathcal{I} \text{ and } e \in C^\mathcal{I}\} \]

\[ (\forall R.C)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid (d, e) \in R^\mathcal{I} \text{ implies } e \in C^\mathcal{I}\} \]

An interpretation $\mathcal{I}$ is a model for a concept $C$ if $C^\mathcal{I} \neq \emptyset$. 
Two main reasoning tasks:

1. **Concept satisfiability** — does there exist a model of $C$?
2. **Concept subsumption** — does $C^\mathcal{I} \subseteq D^\mathcal{I}$ hold for all $\mathcal{I}$? (written $C \sqsubseteq D$)

Why subsumption?

$$\implies$$ Can be used to compute a concept hierarchy:

```
  mammal
   / \   / \\
  predator elephant
    /  \\
   lion
```

In propositionally closed DLs, these can be mutually reduced to one another.
Expressive Power vs. Computational Complexity

In many cases, the expressive power of $\mathcal{ALC}$ does not suffice:

- an elephant has precisely four legs
- every elephant has a bodypart which is a trunk
  and every trunk is a bodypart of an elephant

Many extensions of $\mathcal{ALC}$ have been developed, for example:

- qualified number restrictions ($\leq n \ R \ C$) and ($\geq n \ R \ C$)
- inverse roles $R^{-}$ to be used in existential and universal restriction

But: Increasing expressivity also increases computational complexity

$\implies$ !! tradeoff between expressivity and computational complexity !!
Description Logics should be decidable. But what complexity is “ok”?

**Development of DL Systems**

- **Undecidable**: KL-ONE, NIKL
- **ExpTime**: Fact, DLP, Race
- **PSpace**: Crack, Kris
- **NP**: Classic (AT&T)
- **PTime**: late ’80s, early ’90s, mid ’90s, late ’90s
DLs are more than a Concept Language

Knowledge base

TBox
Terminological Knowledge
Background Knowledge

ABox
Knowledge about individuals

Element Formulated in DL

Elephant = Mammal \sqcap \exists bodypart. Trunk

\top = (Mammal \sqcap \exists bodypart. Hunch) → (Camel \sqcup Dromedary)

dumbo : Elephant
(dumbo, lisa) : child
General TBoxes

There exist several kinds of TBoxes.

**General TBox:** finite set of concept equations $C \equal{} D$

An interpretation $\mathcal{I}$ is a **model** of a TBox $\mathcal{T}$ if

$$C^{\mathcal{I}} = D^{\mathcal{I}} \text{ for all } C \equiv D \in \mathcal{T}.$$  

$$\{ \mathcal{T} \models (\text{Mammal} \sqcap \exists \text{bodypart.Hunch}) \rightarrow (\text{Camel} \sqcup \text{Dromedary}) \}$$

Reasoning tasks with TBoxes:

1. **Concept satisfiability w.r.t. TBoxes**
   Given $C$ and $\mathcal{T}$, does there exist a common model of $C$ and $\mathcal{T}$?

2. **Concept subsumption w.r.t. TBoxes**
   Given $C, D$, and $\mathcal{T}$, does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold in all models of $\mathcal{T}$?

   (written $C \sqsubseteq_{\mathcal{T}} D$)
Concept definition: expression $A \doteq C$ with $A$ concept name and $C$ concept

$\text{Elephant} \doteq \text{Mammal} \cap \exists \text{bodypart}. \text{Trunk}$

A finite set $\mathcal{T}$ of concept definitions is an acyclic TBox if

a) the left-hand sides of concept definitions in $\mathcal{T}$ are unique

b) it contains no "cycles"

not an acyclic TBox: $\{ A_0 \doteq A_1 \cap C$

$A_1 \doteq \exists R. A_2$

$A_2 \doteq A_0 \}$

Acyclic TBoxes can be conceived as macro definitions.
Fix a set of individual names.

An ABox is a finite set of assertions

\[
\begin{align*}
  a : C & \quad (a \text{ individual name, } C \text{ concept}) \\
  (a, b) : R & \quad (a, b \text{ individual names, } R \text{ role name}) \\
  \{\text{dumbo : Elephant} \quad , \quad \text{(dumbo, lisa) : child}\}
\end{align*}
\]

Interpretations \( \mathcal{I} \) map each individual name \( a \) to an element of \( \Delta^\mathcal{I} \).

\( \mathcal{I} \) satisfies an assertion

\[
\begin{align*}
  a : C & \quad \text{iff} \quad a^\mathcal{I} \in C^\mathcal{I} \\
  (a, b) : R & \quad \text{iff} \quad (a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}
\end{align*}
\]

\( \mathcal{I} \) is a model for an ABox \( \mathcal{A} \) if \( \mathcal{I} \) satisfies all assertions in \( \mathcal{A} \).
Reasoning with ABoxes

Reasoning tasks with ABoxes:

1. **ABox consistency**
   
   Given an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, do they have a common model?

2. **Instance checking**

   Given an ABox $\mathcal{A}$, a TBox $\mathcal{T}$, an individual name $a$, and a concept $C$
   
   does $a^\mathcal{I} \in C^\mathcal{I}$ hold in all models of $\mathcal{A}$ and $\mathcal{T}$?

   (written $\mathcal{A}, \mathcal{T} \models a : C$)

   Instance checking can be reduced to ABox consistency.
   
   Concept satisfiability can be reduced to ABox consistency.
Description Logics and First-order Logic

concept names $A \iff$ unary predicates $P_A$
role names $R \iff$ binary predicates $P_R$
concepts $\iff$ formulas with one free variable

\[
\begin{align*}
\varphi^x(A) &= P_A(x) \\
\varphi^x(\neg C) &= \neg \varphi^x(C) \\
\varphi^x(C \cap D) &= \varphi^x(C) \land \varphi^x(D) \\
\varphi^x(C \cup D) &= \varphi^x(C) \lor \varphi^x(D) \\
\varphi^x(\exists R.C) &= \exists y. P_R(x, y) \land \varphi^y(C) \\
\varphi^x(\forall R.C) &= \forall y. P_R(x, y) \to \varphi^y(C)
\end{align*}
\]

Note: - two variables suffices (no "=" , no constants, no function symbols)
- formulas obtained by translation have “guarded” structure
- not all DLs are purely first-order (transitive closure, etc.)
TBoxes:

Let $C$ be a concept and $\mathcal{T}$ a (general or acyclic) TBox.

$$\varphi(C, \mathcal{T}) = \varphi^x(C) \land \forall x. \bigwedge_{D \models E \in \mathcal{T}} \varphi^x(D) \leftrightarrow \varphi^x(E)$$

ABoxes:

**individual names $a$** \iff **constants $c_a$**

$$\varphi(a : C) = \varphi^x(C)[c_a]$$

$$\varphi((a, b) : R) = P_R(c_a, c_b)$$

$$\varphi(\mathcal{A}) = \bigwedge_{\beta \in \mathcal{A}} \varphi(\beta)$$
Description Logics and Modal Logics

**Obvious translation:**

- concept names \( \iff \) propositional variables
- role names \( \iff \) modal parameters
- concepts \( \exists R.C \) \( \iff \) formulas \( \lozenge \psi \)
- concepts \( \forall R.C \) \( \iff \) formulas \( \Box \psi \)

**Notes:**
- Interpretations can be viewed as Kripke structures
- \( \mathcal{ALC} \) is a notational variant of modal \( K_\omega \)
- TBoxes related to universal modality: \( \Box_u \bigwedge_{D \vdash E \in T} D \leftrightarrow E \)
- ABoxes related to nominals / hybrid modal logic
- Extensions of \( \mathcal{ALC} \) are related to graded modal logic, PDL, etc.
Overview of the Course

- Introduction and Tableau Algorithm for $\mathcal{ALCN}$
- Tableau algorithms for expressive Description Logics
- Automata-based decision procedures for expressive Description Logics
- Computational complexity
- Applications, System demonstration, other topics of DL research
**The Description Logic $\mathcal{ALCN}$**

$\mathcal{ALCN}$: $\mathcal{ALC}$

+ unqualified number restrictions ($\leq n \ R$) and ($\geq n \ R$)

Semantics:

$\leq n \ R \Updownarrow = \{ d \in \Delta^I \mid \# \{(d, e) \mid (d, e) \in R^I\} \leq n\}$

$\geq n \ R \Updownarrow = \{ d \in \Delta^I \mid \# \{(d, e) \mid (d, e) \in R^I\} \geq n\}$

Mother of many children: $\text{Female} \sqcap \forall \text{has-children.Human} \sqcap (\geq 4 \text{ has-children})$

Chinese mother: $\text{Female} \sqcap (\leq 1 \text{ has-children}) \sqcup \exists \text{pays-tax.Expensive}$

Note:

Less expressive than qualified number restrictions ($\leq n \ R \ C$) and ($\geq n \ R \ C$)

$\Rightarrow$ decidability/complexity of $\mathcal{ALCN}$-concept satisfiability (without TBoxes)
Appropriate tool: **Tableau Algorithms**

- Frequently used to prove decidability/complexity bounds of DLs
- All state-of-the-art DL reasoners are based on tableau algorithms

**Strategy:**
- Try to construct a model for the input concept $C_0$
- Represent models by **completion trees**
- To decide satisfiability of $C_0$, start with initial completion tree $T_{C_0}$
- Repeatedly apply completion rules and check for obvious contradictions
- Return “satisfiable” iff a complete and contradiction-free completion tree was found
A concept $C$ is in negation normal form (NNF) if

negation occurs only in front of concept names.

**Transformation rules:**

\[
\neg \neg C \quad \iff \quad C
\]

\[
\neg (C \cap D) \quad \iff \quad \neg C \cup \neg D
\]

\[
\neg (C \cup D) \quad \iff \quad \neg C \cap \neg D
\]

\[
\neg (\exists R.C) \quad \iff \quad \forall R.\neg C
\]

\[
\neg (\forall R.C) \quad \iff \quad \exists R.\neg C
\]

\[
\neg (\leq n \ R) \quad \iff \quad (\geq n + 1 \ R)
\]

\[
\neg (\geq 0 \ R) \quad \iff \quad \bot
\]

\[
\neg (\geq n \ R) \quad \iff \quad (\leq n - 1 \ R) \quad \text{if } n > 0
\]
Completion tree:

Finite tree $T = (V, E, \mathcal{L})$ where $\mathcal{L}$ labels

- each node $x \in V$ with a set $\mathcal{L}(x) \subseteq \text{sub}(C_0)$
- each edge $(x, y) \in E$ with a role $\mathcal{L}(x, y)$ occurring in $C_0$.

Initial completion tree for concept $C_0$: $(\{x_0\}, \emptyset, \mathcal{L})$ where $\mathcal{L}(x_0) = \{C_0\}$

Apply completion rules until

- the completion tree is **complete**.

or - there exists a node $x \in V$ such that

Clash \[
\left\{
\begin{array}{l}
1. \{A, \neg A\} \subseteq \mathcal{L}(x) \text{ for some concept name } A \\
2. \{(\leq n \ R), (\geq m \ R)\} \subseteq \mathcal{L}(x) \text{ with } m > n.
\end{array}
\right.
\]
<table>
<thead>
<tr>
<th>Rule</th>
<th>Side 1</th>
<th>Side 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \cdot { C_1 \cap C_2, \ldots } \rightarrow \prod )</td>
<td>( x \cdot { C_1 \cap C_2, C_1, C_2, \ldots } )</td>
<td></td>
</tr>
<tr>
<td>( x \cdot { C_1 \cup C_2, \ldots } \rightarrow \bigcup )</td>
<td>( x \cdot { C_1 \cup C_2, C, \ldots } ) for ( C \in { C_1, C_2 } )</td>
<td></td>
</tr>
<tr>
<td>( x \cdot { \exists R.C, \ldots } \rightarrow \exists )</td>
<td>( x \cdot { \exists R.C, \ldots } )</td>
<td></td>
</tr>
<tr>
<td>( x \cdot { \forall R.C, \ldots } \rightarrow \forall )</td>
<td>( x \cdot { \forall R.C, \ldots } )</td>
<td></td>
</tr>
</tbody>
</table>

\( y \bullet \{ \ldots \} \)

\( y \bullet \{ C \} \)
### Completion Rules II

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \cdot { (\geq n \ R), \ldots } \rightarrow_{\geq} \ x \cdot { (\geq n \ R), \ldots }$</td>
<td>$x$ has no $R$-succ.</td>
</tr>
</tbody>
</table>

$y^* \{ \}$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \cdot { (\leq n \ R), \ldots } \rightarrow_{\leq} \ x \cdot { (\leq n \ R), \ldots }$</td>
<td>$R$</td>
</tr>
</tbody>
</table>

$y^* \{ \}$

merge two $R$-succs.

### Example: blackboard
Correctness of the Algorithm

Lemma

1. The algorithm terminates on any input
2. If the algorithm returns “satisfiable”, then the input concept has a model.
3. If the input concept has a model, then the algorithm returns “satisfiable”.

Corollary

1. $\mathcal{ALCN}$-concept satisfiability and subsumption are decidable
2. $\mathcal{ALCN}$ has the tree model property
3. $\mathcal{ALCN}$ has the finite model property
Role depth of concepts:

\[ d(A) = d(\leq n \ R) = 0 \quad d(\geq n \ R) = 1 \]
\[ d(\neg C) = d(C) \]
\[ d(C \cap D) = d(C \cup D) = \max\{d(C), d(D)\} \]
\[ d(\exists R.C) = d(\forall R.C) = d(C) + 1 \]

The algorithm terminates since:

1. depth of the completion tree bounded by \( d(C_0) \).
2. for each node, at most \( \#\text{sub}(C_0) \) successors are generated
3. each node label contains at most \( \#\text{sub}(C_0) \) concepts
4. concepts are never deleted from node labels
5. nodes may be deleted (via identification), but 1 and 2 is independent from this
A PSPACE upper bound for \( \mathbf{ALCN} \)

Modify \textsc{ExpSpace} tableau algorithm:

1. Construct completion tree in a depth-first manner:

   \[ \text{Diagram of a tree with nodes 1, 2, 3, 4, 5} \]
   
2. Keep only paths of the tree in memory!

   \textbf{Yields a PSpace algorithm:} - paths are of length polynomial in \( |C_0| \)  
   - the outdegree is polynomial in \( |C_0| \).

\textbf{PSPACE lower bound will be proved later!}
Naive approach: unfolding

\[ \rightarrow \] reduce satisfiability w.r.t. TBoxes to satisfiability without TBoxes

Let \( C_0 \) be concept, \( \mathcal{T} \) acyclic TBox

1. replace concept names on right hand sides of definitions \( A \dot{=} C \)
   with their defining concept

2. replace each concept name in \( C_0 \) defined in \( \mathcal{T} \) with its definition.

Terminates due to acyclicity!

**But:** exponential blowup in the worst case

\[
\begin{align*}
A_0 & \dot{=} \forall R.A_1 \cap \forall S.A_1 \\
A_1 & \dot{=} \forall R.A_2 \cap \forall S.A_2 \\
& \vdots \\
A_{k-1} & \dot{=} \forall R.A_k \cap \forall S.A_k
\end{align*}
\]
Unfolding on the Fly

Idea:

Modify existing tableau algorithm to directly deal with acyclic TBoxes

Roadmap:
- convert concept definitions into one of the forms
  \[ A \eqdef \neg X, \ A \eqdef B_1 \sqcap B_2, \ A \eqdef B_1 \sqcup B_2, \ A \eqdef \forall R.B, \ A \eqdef \exists R.B \]
  with \( A, B, B_1, B_2 \) concept names and \( X \) primitive concept name
- restrict node labels to concept names
- make on the fly TBox lookups for rule application

Result: Satisfiability of \( \mathcal{ALC} \)-concepts w.r.t. acyclic TBoxes is \( \text{PSPACE-complete} \).
That’s it

More on tableau algorithms tomorrow!