Part I: Description Logics

Naouel KARAM
karam@isima.fr

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning
- Non standard reasoning
DL origins

- Semantic Networks

- Problem: missing semantics (complex networks)
- Solution: use a logical formalism rather than a network
DL definition

- Descendents of semantics networks, frame-based systems, and KL-ONE

- Family of logic-based knowledge representation (KR) formalisms well-suited for the representation of and reasoning about
  - terminological knowledge
  - ontologies
  - database schemata
  - ...

Architecture of a DL system

- Terminology of application domain
- Description Language
- TBox
- ABox
- Reasoning
- Facts about specific world
- Built complex descriptions
  Logical formalism
- Application Programs
- Derive implicit knowledge
Overview of the tutorial

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning
- Non standard reasoning
The conceptual knowledge of an application domain is represented by:

- **Concepts**: interpreted as a set of individuals
- **Roles**: interpreted as relations between individuals

Complex concept descriptions can be built from atomic ones using concept constructors: \( \cap, \cup, \forall, \exists, \ldots \):

\[
\text{Person} \cap \text{Male} \cap \exists \text{hasChild}.\text{Person}
\]

**concept names** assign a name to a set of individuals

**role names** assign a name to relations between individuals

**concept constructors** connect concept names and role names
The basic description language AL

• Concept descriptions are formed according to the following syntax rules:

\[ C, D \rightarrow T \mid \bot \mid A \mid \neg A \mid C \cap D \mid \forall r. C \mid \exists r. T \]

- top concept
- bottom concept
- atomic concept
- atomic negation
- conjunction
- value restriction
- limited existential quantification

• Examples of AL-concept descriptions

Person \( \cap \exists \text{hasChild}. T \)  persons that have at least one child
Person \( \cap \forall \text{hasChild}. \neg \text{Male} \)  persons all of whose children are not male
Person \( \cap \forall \text{hasChild}. \bot \)  persons without a child
Formal semantics for AL-concept descriptions

- Semantics based on **interpretation** $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$
  - A non empty set $\Delta^\mathcal{I}$ (the domain of the interpretation)
  - An interpretation function $\cdot^\mathcal{I}$
    - an atomic concept $A$: a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
    - an atomic role $r$: a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$

- Inductive extension to concept descriptions
  $$\top^\mathcal{I} = \Delta^\mathcal{I}$$
  $$\bot^\mathcal{I} = \emptyset$$
  $$(\neg A)^\mathcal{I} = \Delta^\mathcal{I} \setminus A^\mathcal{I}$$
  $$(C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$$
  $$(\forall r. C)^\mathcal{I} = \{ x \in \Delta^\mathcal{I} | \forall y : (x, y) \in r^\mathcal{I} \rightarrow y \in C^\mathcal{I} \}$$
  $$(\exists r. T)^\mathcal{I} = \{ x \in \Delta^\mathcal{I} | \exists y : (x, y) \in r^\mathcal{I} \}$$
The family of AL-languages

- More expressive languages can be obtained by adding further constructors
  - **Union of concepts** (\(U\))
    written \(C \sqcup D\)
    interpreted as \((C \sqcup D)^I = C^I \cup D^I\)
  
  - **Full existential quantification** (\(E\))
    written \(\exists r. C\)
    interpreted as \((\exists r. C)^I = \{x \in \Delta^I \mid \exists y : (x, y) \in r^I \land y \in C^I\}\)
  
  - **Negation** (\(C\))
    written \(\neg C\)
    interpreted as \((\neg C)^I = \Delta^I \setminus C^I\)
The family of AL-languages

- Number restrictions (\( \mathbb{N} \))
  
  written \( \geq n \ r \) (at-least restriction)
  \( \leq n \ r \) (at-most restriction)

  interpreted as

  \[
  (\geq n \ r)_I = \{ x \in \Delta^I \mid \#\{ y \in \Delta^I \mid (x, y) \in r^I \} \geq n \} \\
  (\leq n \ r)_I = \{ x \in \Delta^I \mid \#\{ y \in \Delta^I \mid (x, y) \in r^I \} \leq n \}
  \]

- Extending AL by any subset of the above operators yields a particular language identified by a string of the form

  \[ AL[U][E][N][C] \]
### The family of AL-languages

<table>
<thead>
<tr>
<th>Concept constructors</th>
<th>$AL$</th>
<th>$ALN$</th>
<th>$ALE$</th>
<th>$ALEN$</th>
<th>$ALC$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\neg A$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\neg C$</td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$C \cap D$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$C \cup D$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
<tr>
<td>$\forall r.C$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\exists r.\top$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\exists r.C$</td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\geq n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
<tr>
<td>$\leq n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>
The family of \( \mathcal{ALC} \)-languages

- Based on their semantics, prove the equivalence between the languages:

\[ \mathcal{ALC} \quad \text{and} \quad \mathcal{ALUE} \]

\[ \mathcal{ALCN} \quad \text{and} \quad \mathcal{ALUEN} \]

Union and full existential quantification can be expressed using negation, because of the equivalences:

\[
C \cup D \equiv \neg(\neg C \cap \neg D)
\]

\[
\exists r. C \equiv \neg \forall r. \neg C
\]
Overview of the tutorial

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning
- Non standard reasoning
DL knowledge bases

• Formed by two components: The intentional one, called **TBox** and the extensional one called **ABox**.

• **TBox (T)**
  - Schema describing the concepts of the application domain, their properties and the relations between them.

• **ABox (A)**
  - Partial instantiation of the schema describing assertions on individuals.

• A knowledge base is noted

\[ \Sigma = (T, A) \]
A TBox is a set of terminological axioms having one of the forms:

- **Primitive concept**
  - necessary conditions
  
  \[ A \sqsubseteq C \]  
  
  **Primitive Concept specification**

- **Defined concept**
  - necessary and sufficient conditions
  
  \[ A \equiv C \]  
  
  **Concept definition**

Concepts not appearing in the left-hand side of any terminological axiom are called **atomic concepts**.

A more general kind of TBox, called **free-TBox** is obtained by admitting terminological axioms of the form: \( C \sqsubseteq D \) and \( C \equiv D \).

An example of a TBox from the family domain:

- \( \text{Man} \equiv \text{Human} \sqcap \text{Male} \)
- \( \text{Parent} \equiv \text{Human} \sqcap \exists \text{hasChild.Human} \)
- \( \text{Father} \equiv \text{Man} \sqcap \text{Parent} \)
- \( \text{HappyFather} \equiv \text{Father} \sqcap \forall \text{hasChild.\neg Male} \)
Cycles

- A concept name $A$ **directly uses** a concept $B$ in a TBox $\mathcal{T}$ if $B$ appears on the right-hand side of the definition of $A$.
- We call **uses** the transitive closure of the relation **directly uses**.
- $\mathcal{T}$ is called **acyclic** iff there does not exist a concept name in $\mathcal{T}$ that uses itself.

A cyclic TBox:

$\begin{align*}
A_1 & \equiv A_2 \sqcap \exists r. A_4 \\
A_2 & \equiv \exists r. A_3 \sqcap A_5 \\
A_3 & \equiv A_1
\end{align*}$

**Expansion** of an acyclic TBox

$\begin{align*}
\text{Man} & \equiv \text{Human} \sqcap \text{Male} \\
\text{Parent} & \equiv \text{Human} \sqcap \exists \text{hasChild}. \text{Human} \\
\text{Father} & \equiv \text{Man} \sqcap \text{Parent} \\
\text{HappyFather} & \equiv \text{Father} \sqcap \forall \text{hasChild}. \neg \text{Male}
\end{align*}$

The expansion contains only atomic concepts in the right-hand side of each definition.
TBoxes with primitive specifications

• Primitive specifications are used when we are unable to define completely a concept.
• For example, if the concept Man could not be defined in detail, one can require that every man is a human with the primitive specification:

\[ \text{Man} \prec \text{Human} \]

• A TBox \( \mathcal{T} \) containing primitive specifications can be transformed into a regular TBox \( \mathcal{T}' \) with only definitions by adding to primitive specifications a concept standing for the absent part of the definition.

\[ \text{Man} \equiv \text{Human} \cap \overline{\text{Man}} \]

Qualities that distinguish a man among humans

• \( \mathcal{T}' \) is called the normalization of \( \mathcal{T} \)
Semantics

• An interpretation $I$ satisfies the terminological axiom:

$$A \preceq C \text{ if } A^I \subseteq C^I$$

$$A \equiv C \text{ if } A^I = C^I$$

$$C \preceq D \text{ if } C^I \subseteq D^I$$

$$C \equiv D \text{ if } C^I = D^I$$

• An interpretation $I$ is a model of a TBox $T$ iff it satisfies each terminological axiom in $T$. 
An ABox is a set of assertions having one of the forms:

- Concept assertion: $C(a)$
- Role assertion: $r(a, b)$

An example of an ABox from the family domain:

- $\text{Man(PETER)}$
- $\text{Man(MARC)}$
- $\text{hasChild(PETER, MARC)}$

Semantics:
- Extend interpretations to individual names: an interpretation $I$ maps an individual name $a$ to an element $a^I \in \Delta^I$
- An interpretation $I$ satisfies the assertion:
  - $C(a)$ if $a^I \in C^I$
  - $r(a, b)$ if $(a^I, b^I) \in r^I$
- An interpretation $I$ is a model of an ABox $A$ if it satisfies each assertion in $A$
Individual names in the description language

- Individual names can appear in the TBox
  - The *one-of* constructor ($\mathcal{O}$)
    written $\{a_1, \ldots, a_n\}$
    interpreted as $\{a_1, \ldots, a_n\}^\mathcal{I} = \{a_1^\mathcal{I}, \ldots, a_n^\mathcal{I}\}$
    example: $\{CHINA, FRANCE, RUSSIA, UK, USA\}$
  - In a language with the union constructor, a constructor for singleton sets adds sufficient expressiveness to describe arbitrary sets as
    $$\{a_1, \ldots, a_n\} \text{ is equivalent to } \{a_1\} \cup \ldots \cup \{a_n\}$$
  - The *fills* constructor
    written $\mathcal{r} : a$
    interpreted as $$(\mathcal{r} : a)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid (d, a^\mathcal{I}) \in \mathcal{r}^\mathcal{I}\}$$
  - In a language with singleton sets and full existential quantification "fills" does not add anything new as
    $$\mathcal{r} : a \text{ is equivalent to } \exists \mathcal{r}.\{a\}$$
Overview of the tutorial

- Introduction
- Concept descriptions
- Knowledge bases
- Reasoning
Reasoning tasks for TBoxes

• **Concept satisfiability** (written \( T \not\models C \equiv \bot \))
  - A concept \( C \) is satisfiable with respect to \( T \) if there exists a model \( I \) of \( T \) such that \( C^I \) is nonempty.

• **Subsumption** (written \( T \models C \sqsubseteq D \) or \( C \sqsubseteq_T D \))
  - A concept \( C \) is subsumed by a concept \( D \) with respect to \( T \) if \( C^I \subseteq D^I \) for every model \( I \) of \( T \).
  - Example: **Parent** subsume **Father**

• **Equivalence** (written \( T \models C \equiv D \) or \( C \equiv_T D \))
  - Two concepts \( C \) and \( D \) are equivalent with respect to \( T \) if \( C^I = D^I \) for every model \( I \) of \( T \).

• **Disjointness**
  - Two concepts \( C \) and \( D \) are disjoint with respect to \( T \) if \( C^I \cap D^I = \emptyset \) for every model \( I \) of \( T \).
Reductions

• Reduction to subsumption

  (i) $C$ is unsatisfiable $\iff C$ is subsumed by $\bot$;
  (ii) $C$ and $D$ are equivalent $\iff C$ is subsumed by $D$ and $D$ is
       subsumed by $C$;
  (iii) $C$ and $D$ are disjoint $\iff C \cap D$ is subsumed by $\bot$.

• Reduction to satisfiability (systems allowing negation)

  (i) $C$ is subsumed by $D$ $\iff C \cap \neg D$ is unsatisfiable;
  (ii) $C$ and $D$ are equivalent $\iff$ both $(C \cap \neg D)$ and $(\neg C \cap D)$ are
       unsatisfiable;
  (iii) $C$ and $D$ are disjoint $\iff C \cap D$ is unsatisfiable.
Reasoning tasks for ABoxes

- **Consistency** (written $\sum \not \models$)
  - The problem of checking whether $\sum$ is satisfiable, i.e. it has a model

- **Instance checking** (written $\sum \models C(a)$)
  - The problem of checking whether the assertion $C(a)$ is satisfied in every model of $\sum$.

- Reduction of instance checking to consistency

\[
\sum \models C(a) \Leftrightarrow \sum \cup \{\neg C(a)\} \models
\]
Reasoning tasks of a DL system

- Terminological
  - Classification
    
    compute the subsumption hierarchy

- Assertional
  - Realisation
    
    return the most specific concepts, w.r.t. the subsumption relation, of which a concept $a$ is an instance
  - Retrieval
    
    return all instances of $C$. 
 Reasoning algorithms

• Two types of algorithms are employed to decide inference problems:
  – Structural subsumption algorithms
  – Tableau-based algorithms

state of the art technique to decide inferences for a great variety of very expressive DLs only applicable for DLs not allowing for disjunction and full negation, useful for solving non-standard inferences (c.f. Part II)

• Illustrate the underlying idea for both approach
  – Running example

\[
C_{ex} := \exists r. P \sqcap \forall r. Q \sqcap \forall r. Q', \\
D_{ex} := \exists r. (P \sqcap Q) \sqcap \forall r. Q',
\]
Two phases:
- Turn the given potential subsumee into a normal form (making the implicit knowledge contained in the description explicit),
- syntactically compare the (potential) subsumer with the normal form of the (potential) subsumee.

Normalization
- Uses a set of normalization rules
- For our example we need the following rules:

\[
\forall r. E \sqcap \forall r. F \rightarrow \forall r. (E \sqcap F),
\exists r. E \sqcap \forall r. F \rightarrow \exists r. (E \sqcap F) \sqcap \forall r. F.
\]

- We obtain

\[
C'_{ex} := \exists r. (P \sqcap Q \sqcap Q') \sqcap \forall r. (Q \sqcap Q').
\]

check if for all names and restrictions in the subsumer there exists more specific expressions in the normal form of the subsumee

\[
D_{ex} := \exists r. (P \sqcap Q) \sqcap \forall r. Q',
\]
Normalization rules for ALE

\( \forall r. C \cap \forall r. D \rightarrow \forall r. (C \cap D) \) \hspace{1cm} (1)

\( \forall r. C \cap \exists r. D \rightarrow \forall r. C \cap \exists r. (C \cap D) \) \hspace{1cm} (2)

\( \forall r. \top \rightarrow \top \) \hspace{1cm} (3)

\( C \cap \top \rightarrow C \) \hspace{1cm} (4)

\( P \cap \neg P \rightarrow \bot \), for all \( P \in N_C \) \hspace{1cm} (5)

\( \exists r. \bot \rightarrow \bot \) \hspace{1cm} (6)

\( C \cap \bot \rightarrow \bot \) \hspace{1cm} (7)
Tableau algorithms

• Employed for DLs that allow for negation, the subsumption is reduced to deciding satisfiability of concepts: \( C \sqsubseteq D \iff C \cap \neg D \) is unsatisfiable.

\[
C_{ex} \cap \neg D_{ex} = \exists r. P \cap \forall r. Q \cap \forall r. Q' \cap \neg(\exists r. (P \cap Q) \cap \forall r. Q')
\]
\[
\equiv \exists r. P \cap \forall r. Q \cap \forall r. Q' \cap (\forall r. (\neg P \cup \neg Q) \cup \exists r. \neg Q') =: E_{ex}
\]

Negation normal form

• Build \( \mathcal{I} \) with \( E_{ex}^{\mathcal{I}} \neq \emptyset \)

\[
a_0 \in E_{ex}^{\mathcal{I}}
\]
\[
a_1 \text{ with } (a_0, a_1) \in r^{\mathcal{I}} \text{ and } a_1 \in P^{\mathcal{I}}
\]
\[
a_1 \in P^{\mathcal{I}} \cap Q^{\mathcal{I}} \cap Q'^{\mathcal{I}}
\]
\[
a_0 \in (\forall r. (\neg P \cup \neg Q) \cup \exists r. \neg Q')^{\mathcal{I}}
\]
\[
a_1 \in (\neg P \cup \neg Q)^{\mathcal{I}}
\]
\[
a_0 \in (\exists r. \neg Q')^{\mathcal{I}}
\]
\[
a_2 \text{ with } (a_0, a_2) \in r^{\mathcal{I}} \text{ and } a_2 \in \neg Q'^{\mathcal{I}}
\]
\[
a_2 \in (\neg Q')^{\mathcal{I}} \cap Q^{\mathcal{I}} \cap Q'^{\mathcal{I}}
\]

\( E_{ex} \) is unsatisfiable \( \Rightarrow C_{ex} \sqsubseteq D_{ex} \)
### A tableau algorithm for ALCN

<table>
<thead>
<tr>
<th>( A )</th>
<th>rule</th>
<th>( A' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>((C_1 \land C_2)(x))</td>
<td>(\rightarrow \land)</td>
<td>(C_1(x), \ C_2(x))</td>
</tr>
<tr>
<td>((C_1 \lor C_2)(x))</td>
<td>(\rightarrow \lor)</td>
<td>(C(x)) where (C \in {C_1, C_2})</td>
</tr>
<tr>
<td>((\exists r.C)(x))</td>
<td>(\rightarrow \exists)</td>
<td>(C(y), r(x, y)) where (y) not occurring in (A)</td>
</tr>
<tr>
<td>((\forall r.C)(x), r(x, y))</td>
<td>(\rightarrow \forall)</td>
<td>(C(y))</td>
</tr>
<tr>
<td>((\geq r)(x))</td>
<td>(\rightarrow \geq)</td>
<td>({r(x, y_i) \mid 1 \leq i \leq n} \cup {y_i \neq y_j \mid 1 \leq i \leq j \leq n}) where (y_1, ..., y_n) not occurring in (A)</td>
</tr>
<tr>
<td>((\leq r)(x), \ r(x, y_1), ..., r(x, y_{n+1}))</td>
<td>(\rightarrow \leq)</td>
<td>(<a href="%5Ctext%7Brenaming%7D">y_i/y_j</a>)</td>
</tr>
</tbody>
</table>
A tableau algorithm for ALCN

- Test the satisfiability of an ALCN-concept in negation normal form
  \[
  \neg\neg C \rightarrow C \\
  \neg(C \land D) \rightarrow \neg C \lor \neg D \\
  \neg(\exists r. C) \rightarrow \forall r. \neg C \\
  \neg(\forall r. C) \rightarrow \exists r. \neg C \\
  \neg(\leq nr) \rightarrow (\geq n + 1r) \\
  \neg(\geq 0r) \rightarrow \bot \\
  \neg(\geq nr) \rightarrow (\leq n + 1r) \text{ for } n > 0
  \]

- Start with ABox
  \[\mathcal{A}_0 = \{C_0(x_0)\}\]

- Apply propagation rules until
  - no more rule apply
    \[\mathcal{A}_0 \text{ is consistent, } C_0 \text{ satisfiable}\]
  - A contradiction (called clash) occurs
    \[\mathcal{A}_0 \text{ is inconsistent, } C_0 \text{ insatisfiable}\]

Clashes
(i) \{\bot(x)\} \subseteq \mathcal{A};
(ii) \{A(x), \neg A(x)\} \subseteq \mathcal{A};
(iii) \{((\leq nr)(x)) \cup \{r(x, y_i) \mid 1 \leq i \leq n + 1\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n + 1\}\} \subseteq \mathcal{A}.
An example

• Verify the validity of the subsumption:

\[(\geq 3r) \cap \exists r.(P \cap Q) \subseteq (\geq 2r) \cap \exists r. P\]

\[(((\geq 3r) \cap \exists r.(P \cap Q) \cap ((\leq 1r) \cup \forall r. \neg P))(x)\]

\[\rightarrow \exists r(x, y_1) (P \cap Q)(y_1)\]

\[\rightarrow \forall P(y_1) Q(y_1)\]

\[\rightarrow (\leq 1r)(x) \text{ Clash}\]

\[\rightarrow (\forall r. \neg P)(x)\]

\[\rightarrow \forall P(y_1) \neg P(y_2) \neg P(y_3) \text{ Clash}\]
A philosophical question

• The link between structural subsumption and tableau algorithms