Description Logics

Franz Baader
Theoretical Computer Science
TU Dresden
Germany
Reasoning makes implicitly represented knowledge explicit, provided as service by the DL system, e.g.:

**Subsumption:** Is $C$ a subconcept of $D$?

\[ C \sqsubseteq_{\mathcal{T}} D \iff C^\mathcal{I} \subseteq D^\mathcal{I} \text{ for all models } \mathcal{I} \text{ of the TBox } \mathcal{T}. \]

**Satisfiability:** Is the concept $C$ non-contradictory?

\[ C \text{ is satisfiable w.r.t. } \mathcal{T} \iff C^\mathcal{I} \neq \emptyset \text{ for some model } \mathcal{I} \text{ of } \mathcal{T}. \]

**Consistency:** Is the ABox $\mathcal{A}$ non-contradictory?

\[ \mathcal{A} \text{ is consistent w.r.t. } \mathcal{T} \iff \text{it has a model that is also a model of } \mathcal{T}. \]

**Instantiation:** Is $e$ an instance of $C$?

\[ \mathcal{A} \models_{\mathcal{T}} C(e) \iff e^\mathcal{I} \in C^\mathcal{I} \text{ for all models } \mathcal{I} \text{ of } \mathcal{T} \text{ and } \mathcal{A}. \]
Reductions between inference problems

Subsumption to satisfiability:

\[ C \subseteq_{\mathcal{T}} D \ \text{iff} \ C \cap \neg D \text{ is unsatisfiable w.r.t. } \mathcal{T} \]

Satisfiability to subsumption:

\[ C \text{ is satisfiable w.r.t. } \mathcal{T} \ \text{iff} \ \neg C \not\subseteq_{\mathcal{T}} \bot \]

Satisfiability to consistency:

\[ C \text{ is satisfiable w.r.t. } \mathcal{T} \ \text{iff} \ \{ C(a) \} \text{ is consistent w.r.t. } \mathcal{T} \]

Instance to consistency:

\[ a \text{ is an instance of } C \text{ w.r.t. } \mathcal{T} \text{ and } \mathcal{A} \ \text{iff} \ \mathcal{A} \cup \{ \neg C(a) \} \text{ is inconsistent w.r.t. } \mathcal{T} \]

Consistency to instance:

\[ \mathcal{A} \text{ is consistent w.r.t. } \mathcal{T} \ \text{iff} \ a \text{ is not an instance of } \bot \text{ w.r.t. } \mathcal{T} \text{ and } \mathcal{A} \]
Reduction

getting rid of the TBox

Since TBoxes are acyclic, expansion always terminates,
but the expanded concept may be exponential in the size of $\mathcal{T}$.

\[
\begin{align*}
A_0 & \equiv \forall r. A_1 \sqcap \forall s. A_1 \\
A_1 & \equiv \forall r. A_2 \sqcap \forall s. A_2 \\
& \vdots \\
A_{n-1} & \equiv \forall r. A_n \sqcap \forall s. A_n
\end{align*}
\]

The size of $\mathcal{T}$ is linear in $n$,
but the expansion $A_0^T$ contains $A_n$ $2^n$ times.

Reductions:

- $C$ is satisfiable w.r.t. $\mathcal{T}$ iff $C^T$ is satisfiable w.r.t. the empty TBox $\emptyset$.
- $C \sqsubseteq_\mathcal{T} D$ iff $C^T \sqsubseteq_{\emptyset} D^T$. 
Classification

Computing the subsumption hierarchy of all concept names occurring in the TBox.

\[
\begin{align*}
\text{Man} & \equiv \text{Person} \sqcap \neg \text{Female} \\
\text{Woman} & \equiv \text{Person} \sqcap \text{Female} \\
\text{MaleSpeaker} & \equiv \text{Man} \sqcap \exists \text{gives.Talk} \\
\text{FemaleSpeaker} & \equiv \text{Woman} \sqcap \exists \text{gives.Talk} \\
\text{Speaker} & \equiv \text{FemaleSpeaker} \sqcup \text{MaleSpeaker} \\
\text{BusySpeaker} & \equiv \text{Speaker} \sqcap (\geq 3 \text{ gives.Talks})
\end{align*}
\]
**Realization**

Computing the most specific concept names in the TBox to which an ABox individual belongs.

\[
\begin{align*}
\text{Man} & \equiv \text{Person} \sqcap \neg \text{Female} \\
\text{Woman} & \equiv \text{Person} \sqcap \text{Female} \\
\text{MaleSpeaker} & \equiv \text{Man} \sqcap \exists \text{gives.Talk} \\
\text{FemaleSpeaker} & \equiv \text{Woman} \sqcap \exists \text{gives.Talk} \\
\text{Speaker} & \equiv \text{FemaleSpeaker} \sqcup \text{MaleSpeaker} \\
\text{BusySpeaker} & \equiv \text{Speaker} \sqcap (\geq 3 \text{ gives.Talks})
\end{align*}
\]

\[
\begin{align*}
\text{Man(FRANZ)}, & \quad \text{gives(FRANZ, T1)}, \\
\text{Talk(T1)}
\end{align*}
\]

**FRANZ is an instance of Man, Speaker, MaleSpeaker.**

most specific
Complexity of reasoning in Description Logics

A commonly held belief in the 1980ies:

reasoning in KR systems should be tractable, i.e., of polynomial time complexity

- **KL-ONE** and its early successor systems (BACK, MESON, K-Rep, ...) employed polynomial-time algorithms

- even in rather inexpressive DLs, reasoning may be intractable [Brachman&Levesque, 1987]

- reasoning in **KL-ONE** is undecidable [Schmidt-Schauß, 1989]

- reasoning w.r.t. a TBox is intractable even in the minimal DL $\mathcal{FL}_0$ (value-restriction, conjunction) [Nebel, 1990]
Ways out of this dilemma

- **Expressive DL**
  - Sound, but incomplete
  - Tractable algorithms

- **Inexpressive DL**
  - Sound and complete
  - Intractable algorithms

Make bug into a feature?

Made possible by the development of tableau algorithms
Reasoning procedures

requirements

- The procedure should be a decision procedure for the problem:
  - soundness: positive answers are correct
  - completeness: negative answers are correct
  - termination: always gives an answer in finite time

- The procedure should be as efficient as possible:
  preferably optimal w.r.t. the (worst-case) complexity of the problem

- The procedure should be practical:
  easy to implement and optimize, and behave well in applications

Example
- Satisfiability in first-order logic does not have a decision procedure.
- Satisfiability in propositional logic has a decision procedure, but the problem is NP-complete.
Tableau algorithm for $\mathcal{ALC}$

It is sufficient to design a decision procedure for consistency of an ABox without a TBox:

- TBoxes can be eliminated by expanding concept descriptions
- satisfiability, subsumption, and the instance problem can be reduced to consistency

The tableau-based consistency algorithm tries to generate a finite model for the input ABox $\mathcal{A}_0$:

- applies tableau rules to extend the ABox \textit{one rule per constructor}
- checks for obvious contradictions
- an ABox that is complete (no rule applies) and open (no obvious contradictions) describes a model
Tableau algorithm

\[ \mathcal{T} \text{ GoodStudent} \equiv \text{Smart} \sqcap \text{Studious} \]

Subsumption question:
\[ \exists \text{attended.} \text{Smart} \sqcap \exists \text{attended.} \text{Studious} \sqsubseteq_{\mathcal{T}} \exists \text{attended.} \text{GoodStudent} \]

Reduction to satisfiability: is the following concept unsatisfiable w.r.t. \( \mathcal{T} \)?
\[ \exists \text{attended.} \text{Smart} \sqcap \exists \text{attended.} \text{Studious} \sqcap \neg \exists \text{attended.} \text{GoodStudent} \]

Reduction to consistency: is the following ABox inconsistent w.r.t. \( \mathcal{T} \)?
\[ \{ (\exists \text{attended.} \text{Smart} \sqcap \exists \text{attended.} \text{Studious} \sqcap \neg \exists \text{attended.} \text{GoodStudent})(a) \} \]

Expansion: is the following ABox inconsistent?
\[ \{ (\exists \text{attended.} \text{Smart} \sqcap \exists \text{attended.} \text{Studious} \sqcap \neg \exists \text{attended.} (\text{Smart} \sqcap \text{Studious}))(a) \} \]

Negation normal form: is the following ABox inconsistent?
\[ \{ (\exists \text{attends.} \text{Smart} \sqcap \exists \text{attends.} \text{Studious} \sqcap \forall \text{attends.} (\neg \text{Smart} \sqcup \neg \text{Studious}))(a) \} \]
Is the following ABox inconsistent?

\{ (\exists \text{attended. Smart} \cap \exists \text{attended. Studious} \cap \forall \text{attended. (\neg \text{Smart} \cup \neg \text{Studious})})(a) \} \\
\exists r. A \cap \exists r. B \cap \forall r. (\neg A \cup \neg B) \\
\exists r. A, \exists r. B, \forall r. (\neg A \cup \neg B)

\begin{itemize}
\item \text{a}
\item \text{b}
  \begin{itemize}
  \item A
  \item \neg A \cup \neg B
  \item \neg A \quad \neg B
  \end{itemize}
\item \text{c}
  \begin{itemize}
  \item B
  \item \neg A \cup \neg B
  \item \neg A
  \end{itemize}
\end{itemize}

and thus a counterexample to the subsumption relationship
Tableau algorithm

more formal description

Input: An $\mathcal{ALC}$-ABox $\mathcal{A}_0$

Output: “yes” if $\mathcal{A}_0$ is consistent
“no” otherwise

Preprocessing:

transform all concept descriptions in $\mathcal{A}_0$ into negation normal form (NNF)
by applying the following rules:

$\neg(C \sqcap D) \leadsto \neg C \sqcup \neg D$

$\neg(C \sqcup D) \leadsto \neg C \sqcap \neg D$

$\neg\neg C \leadsto C$

$\neg(\exists r.C) \leadsto \forall r.\neg C$

$\neg(\forall r.C) \leadsto \exists r.\neg C$

The NNF can be computed in polynomial time, and it does not change the semantics of the concept.
Tableau algorithm

more formal description

Data structure:
finite set of ABoxes rather than a single ABox: start with \( \mathcal{A}_0 \)

Application of tableau rules:
the rules take one ABox from the set and replace it by finitely many new ABoxes

Termination:
if no more rules apply to any ABox in the set

Answer:
“yes” if the set contains an open ABox, i.e., an ABox not containing an obvious contradiction of the form

\[ A(a) \text{ and } \neg A(a) \quad \text{for some individual name } a \]
### Tableau rules

one for every constructor (except for negation)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The (\sqcap)-rule</strong></td>
<td>(\mathcal{A}) contains ((C \sqcap D)(a)), but not both (C(a)) and (D(a))</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {C(a), D(a)})</td>
</tr>
<tr>
<td><strong>The (\sqcup)-rule</strong></td>
<td>(\mathcal{A}) contains ((C \sqcup D)(a)), but neither (C(a)) nor (D(a))</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {C(a)}) and (\mathcal{A}'' := \mathcal{A} \cup {D(a)})</td>
</tr>
<tr>
<td><strong>The (\exists)-rule</strong></td>
<td>(\mathcal{A}) contains ((\exists r. C)(a)), but there is no (c) with ({r(a, c), C(c)} \subseteq \mathcal{A})</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {r(a, b), C(b)}) where (b) is a new individual name</td>
</tr>
<tr>
<td><strong>The (\forall)-rule</strong></td>
<td>(\mathcal{A}) contains ((\forall r. C)(a)) and (r(a, b)), but not (C(b))</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {C(b)})</td>
</tr>
</tbody>
</table>
**Tableau algorithm**

1. **local correctness:** rules preserve consistency

2. **termination:** no infinite paths

3. **soundness:** any complete and open ABox has a model

   **completeness:** closed ABoxes do not have a model

   **trivial**
Local correctness

rules preserve consistency

The $\exists$-rule

**Condition:** $\mathcal{A}$ contains $(\exists r.C)(a)$, but there is no $c$ with \{\(r(a, c), C(c)\)\} $\subseteq \mathcal{A}$

**Action:** \(\mathcal{A}' := \mathcal{A} \cup \{r(a, b), C(b)\}\) where $b$ is a new individual name

To show: $\mathcal{A}$ has a model iff $\mathcal{A}'$ has a model

⇒ Let $\mathcal{I}$ be a model of $\mathcal{A}$.

Since $(\exists r.C)(a) \in \mathcal{A}$, there is a $d \in \Delta^\mathcal{I}$ such that $(a^\mathcal{I}, d) \in r^\mathcal{I}$ and $d \in C^\mathcal{I}$.

Let $\mathcal{I}'$ be the interpretation that coincides with $\mathcal{I}$, with the exception that $b^\mathcal{I}' = d$.

Since $b$ does not occur in $\mathcal{A}$, $\mathcal{I}'$ is a model of $\mathcal{A}$.

By definition of $b^\mathcal{I}'$, it is also a model of \{\(r(a, b), C(b)\)\}.

⇐ trivial since $\mathcal{A} \subseteq \mathcal{A}'$. 
Termination is an easy consequence of the following facts:

The label $\mathcal{L}(a)$ of an individual name consists of the concepts in concept assertions for $a$.

1. rule application is monotonic: every application of a rule extends the label of an individual, and does not remove anything;

2. concepts in labels are subdescriptions of concepts occurring in the input ABox $\mathcal{A}_0$;

$\Rightarrow$ finite number of rule applications per individual

3. the number of new individuals that are $r$-successors of an individual is bounded by the number of existential restrictions in $\mathcal{A}_0$;

4. the length of successor chains of new individuals is bounded by the maximal size of the concepts in $\mathcal{A}_0$:
   - if $x$ is a new individual, then it has a unique predecessor $y$
   - the maximal size of concepts in $\mathcal{L}(x)$ is strictly smaller than in $\mathcal{L}(y)$

$\Rightarrow$ finitely many new individuals
Soundness

any complete and open ABox has a model

Let $\mathcal{A}$ be a complete and open ABox.

The canonical interpretation $\mathcal{I}_\mathcal{A}$ induced by $\mathcal{A}$ is defined as follows:

- $\Delta^\mathcal{I}_\mathcal{A} := \{ x \mid x \text{ is an individual name occurring in } \mathcal{A} \}$
- $x^\mathcal{I}_\mathcal{A} := x$ for all individual names occurring in $\mathcal{A}$
- $A^\mathcal{I}_\mathcal{A} := \{ x \mid A(x) \in \mathcal{A} \}$ for all $A \in N_C$
- $r^\mathcal{I}_\mathcal{A} := \{ (x, y) \mid r(x, y) \in \mathcal{A} \}$ for all $r \in N_R$

Claim

$\mathcal{I}_\mathcal{A}$ is a model of $\mathcal{A}$. 
Tableau algorithm is a decision procedure for consistency

1. Started with a finite ABox $\mathcal{A}_0$ in NNF
   the algorithm always terminates with
   a finite set of complete ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_n$

2. Local correctness: $\mathcal{A}_0$ consistent iff
   one of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ consistent

3. Answer “no”:
   none of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ open
   $\mathcal{A}_1, \ldots, \mathcal{A}_n$ inconsistent
   $\mathcal{A}_0$ inconsistent

4. Answer “yes”:
   one of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ open
   $\mathcal{A}_0$ consistent
Adding number restrictions

Number restrictions: \((\geq n \cdot r. C), (\leq n \cdot r. C)\) with semantics

\[
\begin{align*}
(\geq n \cdot r. C)^I & := \{d \in \Delta^I \mid \text{card}\{e \mid (d, e) \in r^I \land e \in C^I\} \geq n\} \\
(\leq n \cdot r. C)^I & := \{d \in \Delta^I \mid \text{card}\{e \mid (d, e) \in r^I \land e \in C^I\} \leq n\}
\end{align*}
\]

Negation normal form:

\[
\begin{align*}
\neg(\geq n + 1 \cdot r. C) & \iff (\leq n \cdot r. C) \\
\neg(\geq 0 \cdot r. C) & \iff \perp \\
\neg(\leq n \cdot r. C) & \iff (\geq n + 1 \cdot r. C)
\end{align*}
\]

Extension of algorithm:

- new rules: \(\geq\)-rule and \(\leq\)-rule
- new assertions: inequality assertions of the form \(x \neq y\) with obvious semantics \(x^I \neq y^I\) viewed as symmetric
- new obvious contradictions
Adding number restrictions

The $\geq$-rule

**Condition:** $\mathcal{A}$ contains $(\geq n \mathit{r.C})(a)$, but there are no $c_1, \ldots, c_n$ with
\[
\{r(a, c_1), C(c_1), \ldots, r(a, c_n), C(c_n)\} \cup \{c_i \neq c_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}
\]

**Action:**

\[
\mathcal{A}' := \mathcal{A} \cup \{r(a, b_1), C(b_1), \ldots, r(a, b_n), C(b_n)\} \cup \{b_i \neq b_j \mid 1 \leq i < j \leq n\}
\]

where $b_1, \ldots, b_n$ are new individual names

The $\leq$-rule

**Condition:** $\mathcal{A}$ contains $(\leq n \mathit{r.C})(a)$, and there are $b_1, \ldots, b_{n+1}$ with
\[
\{r(a, b_1), C(b_1), \ldots, r(a, b_{n+1}), C(b_{n+1})\} \subseteq \mathcal{A},
\]
but $\{b_i \neq b_j \mid 1 \leq i < j \leq n + 1\} \not\subseteq \mathcal{A}$

**Action:**

for all $i < j$ with $b_i \neq b_j \not\in \mathcal{A}$

\[
\mathcal{A}_{i,j} := \mathcal{A}[b_i \leftarrow b_j]
\]

$b_i$ replaced by $b_j$
Adding number restrictions

- $A$ contains $(\leq n r.C)(a)$, and there are $b_1, \ldots, b_{n+1}$ with
  $$\{r(a, b_1), C(b_1), \ldots, r(a, b_{n+1}); C(b_{n+1})\} \subseteq A$$ and
  $$\{b_i \neq b_j \mid 1 \leq i < j \leq n + 1\} \subseteq A$$

- $A$ contains $a \neq a$ for some individual name $a$
Adding GCIs

\( C \subseteq D \) with semantics \( C^I \subseteq D^I \)

A finite set of GCIs can be encoded into one GCI of the form \( \top \subseteq C \):

\[
\{ C_1 \subseteq D_1, \ldots, C_n \subseteq D_n \} \quad \rightarrow \quad \{ \top \subseteq (\neg C_1 \cup D_1) \cap \ldots \cap (\neg C_n \cup D_n) \}
\]

Consider a GCI \( \top \subseteq C \) where \( C \) is in NNF.

The GCI-rule for \( \top \subseteq C \)

**Condition:** \( \mathcal{A} \) contains the individual name \( a \), but not \( C(a) \)

**Action:** \( \mathcal{A}' := \mathcal{A} \cup \{ C(a) \} \)
Task

What do you think could be a problem with GCIs?
Adding GCIs

does this yield a decision procedure?

- local correctness, completeness, and soundness are easy to show

- termination does not hold:

Test consistency of \( \{ P(a) \} \) w.r.t. the GCI \( \top \subseteq \exists r. P \)

\[
\begin{align*}
& a \xrightarrow{r} P \xrightarrow{r} P \xrightarrow{r} P \\
& \exists r. P \quad \exists r. P \quad \exists r. P
\end{align*}
\]

Solution: blocking

- \( y \) is blocked by \( x \) iff \( \mathcal{L}(y) \subseteq \mathcal{L}(x) \)

- to avoid cyclic blocking we fix an enumeration of the individual names, and add to the blocking condition that \( y \) comes after \( x \) in the enumeration

- generating rules are not applied to blocked individuals
Complexity of Reasoning

- For \( \text{ALC} \), the subsumption problem and the instance problem are \( \text{PSpace} \)-complete.

  The tableau algorithm as described needs exponential space, but it can be modified such that it needs only polynomial space.

- Both TBoxes and number restrictions can be added without increasing the complexity.

- W.r.t. general TBoxes, the subsumption and the instance problem are \( \text{ExpTime} \)-complete.

  Tableau algorithms do not “easily” yield this upper bound, but they are more practical than the worst-case optimal automata-based algorithms.

- The tableau algorithms implemented in systems like FaCT and Racer are highly optimized, and behave quite well on large knowledge bases.