Semantic Networks and Description Logics

Description Logics – Algorithms

Motivation

Structural Subsumption Algorithms

Tableau Subsumption Method

Reasoning Problems & Algorithms

- **Satisfiability** or **subsumption** of concept descriptions
- **Satisfiability** or **instance relation** in ABoxes

**Structural subsumption algorithms**

- *Normalization* of concept descriptions and *structural comparison*
- very fast, but can only be used for small DLs

**Tableau algorithms**

- Similar to modal tableau methods
- Meanwhile the method of choice

**Small Logic** $\mathcal{FL^-}$

- $\forall r. C$
- $\exists r$ (simple existential quantification)

**Idea**

1. In the conjunction, collect all *universally quantified expressions* (also called *value restrictions*) with the same role and build *complex value restriction*:

   $$\forall r. C \sqcap \forall r. D \rightarrow \forall r. (C \sqcap D).$$

2. Compare all conjuncts with each other. For each conjunct in the subsuming concept there should be a *corresponding one* in the subsumed one.
Example

\[ D = \text{Human} \land \exists \text{has-child} \land \forall \text{has-child} \land \text{Human} \\land \forall \text{has-child} \\exists \text{has-child} \]

\[ C = \text{Human} \land \text{Female} \land \exists \text{has-child} \land \forall \text{has-child} (\text{Human} \land \text{Female} \land \exists \text{has-child}) \]

Check: \( C \subseteq D \)

1. Collect value restrictions in \( D \):
   \( \exists \text{has-child}(\text{Human} \land \exists \text{has-child}) \)

2. Compare:
   2.1 For \text{Human} in \( D \), we have \text{Human} in \( C \)
   2.2 For \exists \text{has-child} in \( D \), we have ...
   2.3 For \forall \text{has-child}(\_\_\_) in \( D \), we have ...
      2.3.1 For \text{Human} ...
      2.3.2 For \exists \text{has-child} ...

\( \Rightarrow C \) is subsumed by \( D \)

Subsumption Algorithm

**SUB \((C, D)\) algorithm:**

1. Reorder terms (commutativity, associativity and value restriction law):
   \[ C = \bigcap A_i \land \exists r_j \land \forall r_k : C_k \]
   \[ D = \bigcap B_l \land \exists s_m \land \forall s_n : D_n \]

2. For each \( B_l \) in \( D \), is there an \( A_i \) in \( C \) with \( A_i = B_l \)?
3. For each \( \exists r_j \) in \( D \), is there an \( \exists r_j \) in \( C \) with \( r_j = s_m \)?
4. For each \( \forall s_n : D_n \) in \( D \), is there a \( \forall r_k : C_k \) in \( C \) such that \( C_k \subseteq D_n \) and \( s_n = r_k \)?
   \( \Rightarrow C \subseteq D \) if all questions are answered positively

Soundness

**Theorem (Soundness)**

\( SUB(C, D) \Rightarrow C \subseteq D \)

**Proof sketch.**

Reordering of terms \((1)\):

a) Commutativity and associativity are trivial
b) Value restriction law. We show: \( (\forall r.(C \land D))^T = (\forall r.C \land \forall r.D)^T \)
   
   Assumption: \( d \in (\forall r.(C \land D))^T \)
   
   Case 1: \( \exists e : (d,e) \in r^T \) \quad \checkmark
   
   Case 2: \( \exists e : (d,e) \in r^T \Rightarrow e \in (C \land D)^T \Rightarrow e \in C^T, e \in D^T \)

   Since \( e \) is arbitrary: \( d \in (\forall r.C)^T, d \in (\forall r.D)^T \) then \( d \) must also be conjunction, i.e., \( (\forall r.(C \land D))^T \subseteq (\forall r.C \land \forall r.D)^T \)

   Other direction is similar

\((2+3+4)\): Induction on the nesting depth of \( \forall \)-expressions

Completeness

**Theorem (Completeness)**

\( C \subseteq D \Rightarrow SUB(C, D) \)

**Proof idea.**

One shows the contrapositive:

\( \neg SUB(C, D) \Rightarrow C \nsubseteq D \)

**Idea:** If one of the rules leads to a negative answer, we use this to construct an interpretation with a special element \( d \) such that

\( d \in C^T \), but \( d \notin D^T \)
Generalizing the Algorithm

Extensions of $\mathcal{FL}^-$ by

- $\neg A$ (atomic negation),
- $(\leq n r), (\geq n r)$ (cardinality restrictions),
- $r \circ s$ (role composition)

does not lead to any problems. **However:** If we use full existential restrictions, then it is very unlikely that we can come up with a *simple* structural subsumption algorithm – having the same flavor as the one above. **More precisely:** There is (most probably) no algorithm that uses polynomially many reorderings and simplifications and allows for a simple structural comparison.

**Reason:** Subsumption for $\mathcal{FL}^- + \exists r.C$ is NP-hard (Nutt).

ABox Reasoning

**Idea:** abstraction + classification

- Complete ABox by propagating value restrictions to role fillers
- Compute for each object its *most specialized concepts*
- These can then be handled using the ordinary subsumption algorithm

Tableau Subsumption Method

**Logic** $\mathcal{ALC}$

- $C \cap D$
- $C \cup D$
- $\neg C$
- $\forall r.C$
- $\exists r.C$

**Idea:** Decide (un-)satisfiability of a concept description $C$ by trying to systematically construct a model for $C$. If that is successful, $C$ is satisfiable. Otherwise $C$ is unsatisfiable.

Example: Subsumption in a TBox

**TBox**

- Hermaphrodite $= \text{Male} \cap \text{Female}$
- Parents-of-sons-and-daughters $= \exists \text{has-child}. \text{Male} \cap \exists \text{has-child}. \text{Female}$
- Parents-of-hermaphrodite $= \exists \text{has-child}. \text{Hermaphrodite}$

**Query**

- Parents-of-sons-and-daughters $\subseteq$ Parents-of-hermaphrodites
Reductions

1. **Unfolding**
   \[ \exists \text{has-child}. \text{Male} \land \exists \text{has-child}. \text{Female} \subseteq \exists \text{has-child}. (\text{Male} \land \text{Female}) \]

2. **Reduction to unsatisfiability**
   Is
   \[ \exists \text{has-child}. \text{Male} \land \exists \text{has-child}. \text{Female} \land \neg (\exists \text{has-child}. (\text{Male} \land \text{Female})) \]
   unsatisfiable?

3. **Negation normal form** (move negations inside):
   \[ \exists \text{has-child}. \text{Male} \land \exists \text{has-child}. \text{Female} \land \forall \text{has-child}. (\neg \text{Male} \lor \neg \text{Female}) \]

4. **Try to construct a model**

---

Model Construction (1)

1. **Assumption**: There exists an object \( x \) in the interpretation of our concept:
   \[ x \in (\exists \ldots)^I \]

2. This implies that \( x \) is in the interpretation of all conjuncts:
   \[ x \in (\exists \text{has-child}. \text{Male})^I \]
   \[ x \in (\exists \text{has-child}. \text{Female})^I \]
   \[ x \in (\forall \text{has-child}. (\neg \text{Male} \lor \neg \text{Female}))^I \]

3. This implies that there should be objects \( y \) and \( z \) such that
   \( (x, y) \in \text{has-child}^I, (x, z) \in \text{has-child}^I, y \in \text{Male}^I \) and
   \( z \in \text{Female}^I \) and...

---

Model Construction (2)

\[ x : \exists \text{has-child}. \text{Male} \]
\[ x : \exists \text{has-child}. \text{Female} \]

---

Model Construction (3)

\[ x : \exists \text{has-child}. \text{Male} \]
\[ x : \exists \text{has-child}. \text{Female} \]
\[ x : \forall \text{has-child}. (\neg \text{Male} \lor \neg \text{Female}) \]
Tableau Subsumption Method  
Model Construction

Model Construction (4)

\[
\begin{align*}
  x & : \exists \text{has-child}.\text{Male} \\
  x & : \exists \text{has-child}.\text{Female} \\
  x & : \forall \text{hat-child}.(\neg \text{Male} \sqcup \neg \text{Female}) \\
  y & : \neg \text{Male}
\end{align*}
\]

Model Construction (5)

\[
\begin{align*}
  x & : \exists \text{has-child}.\text{Male} \\
  x & : \exists \text{has-child}.\text{Female} \\
  x & : \forall \text{hat-child}.(\neg \text{Male} \sqcup \neg \text{Female}) \\
  y & : \neg \text{Female} \\
  z & : \neg \text{Male}
\end{align*}
\]

\[\implies \text{Model constructed!}\]

Tableau Subsumption Method  
Equivalences & NNF

Tableau Method (1): NNF

\[C \equiv D \iff C \subseteq D \text{ and } D \subseteq C.\]
Now we have the following equivalences:

\[
\begin{align*}
  \neg (C \cap D) & \equiv \neg C \cup \neg D \\
  \neg (C \cup D) & \equiv \neg C \cap \neg D \\
  \neg \neg C & \equiv C \\
  \neg (\forall r.C) & \equiv \exists r.\neg C \\
  \neg (\exists r.C) & \equiv \forall r.\neg C
\end{align*}
\]

These equivalences can be used to move all negations signs to the inside, resulting in concept description where only concept names are negated: **negation normal form (NNF)**

**Theorem**

(NNF) The negation normal form of a \(\mathcal{ALC}\) concept can be computed in polynomial time.

Tableau Subsumption Method  
Constraint Systems

Tableau Method (2): Constraint Systems

A **constraint** is a syntactical object of the form: \(x: C\) or \(xry\), where \(C\) is a concept description in NNF, \(r\) is a role name and \(x\) and \(y\) are **variable names**.

Let \(I\) be an interpretation. An **I-assignment** \(\alpha\) is a function that maps each variable symbol to an object of the universe \(D\).

A constraint \(x: C\) \(xry\) is **satisfied** by an \(I\)-assignment \(\alpha\), if \(\alpha(x) \in C^I\) \(((\alpha(x), \alpha(y)) \in r^I)\).

Ein **constraint system** \(S\) is a finite, non-empty set of constraints. A \(I\)-assignment \(\alpha\) satisfies \(S\) if \(\alpha\) satisfies each constraint in \(S\). \(S\) is **satisfiable** if there exists \(I\) and \(\alpha\) such that \(\alpha\) satisfies \(S\).

**Theorem**

An \(\mathcal{ALC}\) concept \(C\) in NNF is satisfiable iff the system \{\(x: C\)\} is satisfiable.
Tableau Subsumption Method

Tableau Method (3): Transforming Constraint Systems

Transformation rules:

1. $S \rightarrow \exists \{x: C_1, x: C_2\} \cup S$
   if $(x: C_1 \cap C_2) \in S$ and either $(x: C_1)$ or $(x: C_2)$ or both are not in $S$.

2. $S \rightarrow \exists \{x: D\} \cup S$
   if $(x: C_1 \cap C_2) \in S$ and neither $(x: C_1)$ nor $(x: C_2)$ and $D = C_1$ or $D = C_2$.

3. $S \rightarrow_3 \{xry, y: C\} \cup S$
   if $(x: \exists r.C) \in S$, $y$ is a fresh variable, and there is no $z$ s.t.
   $(zr) \in S$ and $(z: C) \in S$.

4. $S \rightarrow_\forall \{y: C\} \cup S$
   if $(x: \forall r.C), (xry) \in S$ and $(y: C) \notin S$.

Deterministic rules (1,3,4) vs. non-deterministic (2).

Generating rules (3) vs. non-generating (1,2,4).

Tableau Method (4): Invariances

Theorem (Invariance)
Let $S$ and $T$ be constraint systems:

1. If $T$ has been generated by applying a deterministic rule to $S$, then $S$ is satisiable iff $T$ is satisiable.

2. If $T$ has been generated by applying a non-deterministic rule to $S$, then $S$ is satisiable if $T$ is satisiable. Furthermore, if a non-deterministic rule can be applied to $S$, then it can be applied such that $S$ is satisiable iff the resulting system $T$ is satisiable.

Theorem (Termination)
Let $C$ be an $\mathcal{ALC}$ concept description in NNF. Then there exists no infinite chain of transformations starting from the constraint system $\{x: C\}$.

Tableau Method (5): Soundness and Completeness

A constraint system is called closed if no transformation rule can be applied.

A clash is a pair of constraints of the form $x: A$ and $x: \neg A$, where $A$ is a concept name.

Theorem (Soundness and Completeness)
A closed constraint system is satisiable iff it does not contain a clash.

Proof idea.

$\Rightarrow$: obvios.

$\Leftarrow$: Construct a model by using the concept labels.

Space Requirements

Because the tableau method is non-deterministic ($\rightarrow_\exists r$ rule) ... there could be exponentially many closed constraint systems in the end. Interestingly, even one constraint system can have exponential size.

Example:

$\exists r.A \cap \exists r.B \cap$

$\forall r. (\exists r.A \cap \exists r.B \cap$

$\forall r. (\exists r.A \cap \exists r.B \cap$

$\forall r. (\ldots))$

However: One can modify the algorithm so that it needs only poly. space.

Idea: Generating a $y$ only for one $\exists r.C$ and then proceeding into the depth.
ABox Reasoning

ABox satisfiability can also be decided using the tableau method if we can add constraints of the form $x \neq y$ (for UNA):

- **Normalize** and **unfold** and add inequalities for all pairs of objects mentioned in the ABox.
- Strictly speaking, in $\mathcal{ALC}$ we do not need this because we are never forced to identify two objects.

Literature