Principles of Knowledge Representation and Reasoning

Description Logics – Algorithms

Bernhard Nebel, Malte Helmert and Stefan Wölfl

Albert-Ludwigs-Universität Freiburg

July 22, 2008

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

Structural Subsumption Algorithms

- **Small Logic** $\mathcal{FL}^-$
  - $C \sqcap D$
  - $\forall r.C$
  - $\exists r$ (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

$$
\begin{aligned}
D &= \texttt{Human} \sqcap \exists\texttt{has-child} \sqcap \forall\texttt{has-child}.\texttt{Human} \sqcap \\
&\quad \forall\texttt{has-child}.\exists\texttt{has-child} \\
C &= \texttt{Human} \sqcap \texttt{Female} \sqcap \exists\texttt{has-child} \sqcap \\
&\quad \forall\texttt{has-child}.(\texttt{Human} \sqcap \texttt{Female} \sqcap \exists\texttt{has-child})
\end{aligned}
$$

Check: $C \sqsubseteq D$

1. Collect value restrictions in D: $\ldots\forall\texttt{has-child}.(\texttt{Human} \sqcap \exists\texttt{has-child})$
2. Compare:
    - 2.1 For $\texttt{Human}$ in D, we have $\texttt{Human}$ in C
    - 2.2 For $\exists\texttt{has-child}$ in D, we have ...
    - 2.3 For $\forall\texttt{has-child}.(\ldots)$ in D, we have ...
        - 2.3.1 For $\texttt{Human}$ ...
        - 2.3.2 For $\exists\texttt{has-child}$ ...

$\leadsto$ C is subsumed by D!

Subsumption Algorithm

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

1. Reorder terms (*commutativity*, *associativity* and *value restriction law*):

$$\begin{aligned} C &= \bigsqcap A_i \sqcap \bigsqcap \exists r_j \sqcap \bigsqcap \forall r_k : C_k \\ D &= \bigsqcap B_l \sqcap \bigsqcap \exists s_m \sqcap \bigsqcap \forall s_n : D_n \end{aligned}$$

2. For each B_l in D, is there an A_i in C with $A_i = B_l$?
3. For each $\exists s_m$ in D, is there an $\exists r_j$ in C with $s_m = r_j$?
4. For each $\forall s_n : D_n$ in D, is there a $\forall r_k : C_k$ in C such that $C_k \sqsubseteq D_n$ and $s_n = r_k$?

$\rightsquigarrow$ $C \sqsubseteq D$ iff all questions are answered positively

Soundness

Theorem (Soundness)

$SUB(C, D) \Rightarrow C \sqsubseteq D$

Proof sketch.

Reordering of terms *(1)*:

a) Commutativity and associativity are trivial

b) Value restriction law. We show: $\left(\forall r.(C \sqcap D)\right)^{\mathcal{I}} = \left(\forall r.C \sqcap \forall r.D\right)^{\mathcal{I}}$

Assumption: $d \in \left(\forall r.(C \sqcap D)\right)^{\mathcal{I}}$

Case 1: $\nexists e : (d, e) \in r^{\mathcal{I}} \quad \surd$

Case 2: $\exists e : (d, e) \in r^{\mathcal{I}} \Rightarrow e \in (C \sqcap D)^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}},\ e \in D^{\mathcal{I}}$
Since e is arbitrary: $d \in (\forall r.C)^{\mathcal{I}}$, $d \in (\forall r.D)^{\mathcal{I}}$ then d must also be conjunction, i.e., $\left(\forall r.(C \sqcap D)\right)^{\mathcal{I}} \subseteq (\forall r.C \sqcap \forall r.D)^{\mathcal{I}}$

Other direction is similar

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

Completeness

Theorem (Completeness)

$C \sqsubseteq D \Rightarrow SUB(C, D)$

Proof idea.

One shows the contrapositive:

$$\neg \mathrm{SUB}(C, D) \Rightarrow C \not\sqsubseteq D$$

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

$$d \in C^{\mathcal{I}}, \text{ but } d \notin D^{\mathcal{I}}$$

$\square$

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 Method

- **Logic** $\mathcal{ALC}$
  - $C \sqcap D$
  - $C \sqcup D$
  - $\neg C$
  - $\forall r.C$
  - $\exists r.C$
- **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

$\texttt{Hermaphrodite} \doteq \texttt{Male} \sqcap \texttt{Female}$

$\texttt{Parents-of-sons-and-daughters} \doteq$

$\exists\texttt{has-child.Male} \sqcap \exists\texttt{has-child.Female}$

$\texttt{Parents-of-hermaphrodite} \doteq \exists\texttt{has-child.Hermaphrodite}$

Query

$\texttt{Parents-of-sons-and-daughters} \sqsubseteq_{\mathcal{T}}$

$\texttt{Parents-of-hermaphrodites}$

Reductions

1. *Unfolding*
   $\exists\texttt{has-child.Male} \sqcap \exists\texttt{has-child.Female} \sqsubseteq$
   $\exists\texttt{has-child.}(\texttt{Male} \sqcap \texttt{Female})$
2. *Reduction to unsatisfiability*
   Is
   $\exists\texttt{has-child.Male} \sqcap \exists\texttt{has-child.Female} \sqcap$
   $\neg(\exists\texttt{has-child.}(\texttt{Male} \sqcap \texttt{Female}))$
   unsatisfiable?
3. *Negation normal form* (move negations inside):
   $\exists\texttt{has-child.Male} \sqcap \exists\texttt{has-child.Female} \sqcap$
   $\forall\texttt{has-child.}(\neg\texttt{Male} \sqcup \neg\texttt{Female})$
4. *Try to construct a model*

Model Construction (1)

1. **Assumption**: There exists an object x in the interpretation of our concept:
$$x \in (\exists \ldots)^{\mathcal{I}}$$

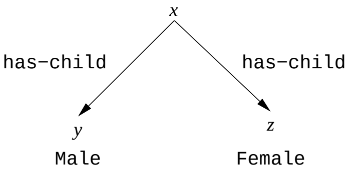
2. This implies that x is in the interpretation of all conjuncts:
$$\begin{array}{lcl} x & \in & (\exists \texttt{has-child}.\texttt{Male})^{\mathcal{I}} \\ x & \in & (\exists \texttt{has-child}.\texttt{Female})^{\mathcal{I}} \\ x & \in & \big(\forall \texttt{has-child}.(\neg \texttt{Male} \sqcup \neg \texttt{Female})\big)^{\mathcal{I}} \end{array}$$

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

Model Construction (2)

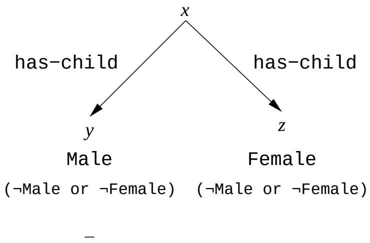
x : $\exists$has-child.Male

x : $\exists$has-child.Female



Model Construction (3)

x : $\exists$has-child.Male
x : $\exists$has-child.Female
x : $\forall$has-child.($\neg$Male $\sqcup$ $\neg$Female)



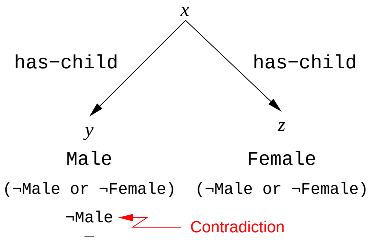
Model Construction (4)

x : $\exists$has-child.Male

x : $\exists$has-child.Female

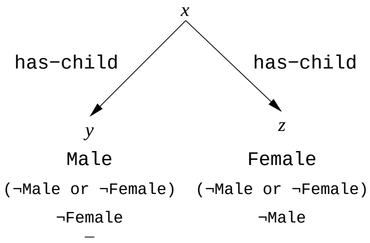
x : $\forall$has-child.($\neg$Male $\sqcup$ $\neg$Female)

y : $\neg$Male



Model Construction (5)

x : $\exists$has-child.Male
x : $\exists$has-child.Female
x : $\forall$has-child.($\neg$Male $\sqcup$ $\neg$Female)
y : $\neg$Female
z : $\neg$Male



$\rightsquigarrow$ Model **constructed**!

Tableau Method (1): NNF

$C \equiv D$ iff $C \sqsubseteq D$ and $D \sqsubseteq C$.

Now we have the following equivalences:

$$\begin{aligned}
\neg(C \sqcap D) &\equiv \neg C \sqcup \neg D \\
\neg(C \sqcup D) &\equiv \neg C \sqcap \neg D \\
\neg\neg C &\equiv C \\
\neg(\forall r.C) &\equiv \exists r.\neg C \\
\neg(\exists r.C) &\equiv \forall r.\neg C
\end{aligned}$$

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 an $\mathcal{ALC}$ concept can be computed in polynomial time.

Tableau Method (2): Constraint Systems

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

Let $\mathcal{I}$ be an interpretation. An **$\mathcal{I}$-assignment** α is a function that maps each variable symbol to an object of the universe $\mathcal{D}$.

A *constraint* $x\colon C$ (xry) is **satisfied** by an $\mathcal{I}$-assignment α, if $\alpha(x) \in C^{\mathcal{I}}$ ($(\alpha(x), \alpha(y)) \in r^{\mathcal{I}}$).

A **constraint system** S is a finite, non-empty set of constraints. An $\mathcal{I}$-assignment α satisfies S if α satisfies each constraint in S. S is **satisfiable** if there exists $\mathcal{I}$ and α such that α satisfies S.

Theorem

An $\mathcal{ALC}$ concept C in NNF is satisfiable iff the system $\{x\colon C\}$ is satisfiable.

Tableau Method (3): Transforming Constraint Systems

Transformation rules:

1. $S \rightarrow_{\sqcap} \{x\colon C_1, x\colon C_2\} \cup S$
   if $(x\colon C_1 \sqcap C_2) \in S$ and either $(x\colon C_1)$ or $(x\colon C_2)$ or both are not in S.
2. $S \rightarrow_{\sqcup} \{x\colon D\} \cup S$
   if $(x\colon C_1 \sqcup C_2) \in S$ and neither $(x\colon C_1) \in S$ nor $(x\colon C_2) \in S$ and $D = C_1$ *or* $D = C_2$.
3. $S \rightarrow_{\exists} \{xry, y\colon C\} \cup S$
   if $(x\colon \exists r.C) \in S$, y is a *fresh variable*, and there is no z s.t. $(xrz) \in S$ and $(z\colon C) \in S$.
4. $S \rightarrow_{\forall} \{y\colon C\} \cup S$
   if $(x\colon \forall r.C), (xry) \in S$ and $(y\colon C) \notin S$.

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

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 $x\colon A$ and $x\colon \neg A$, where A is a concept name.

Theorem (Soundness and Completeness)

A closed constraint system is satisfiable iff it does not contain a clash.

Proof idea.

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

Space Requirements

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

Example:

$$\begin{array}{c} \exists r.A \sqcap \exists r.B \sqcap \\ \forall r.\Big(\exists r.A \sqcap \exists r.B \sqcap \\ \quad \forall r.(\exists r.A \sqcap \exists r.B \sqcap \\ \qquad \forall r.(\ldots))\Big) \end{array}$$

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 $\boxed{x \neq y}$ (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

- Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
- Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.
- Bernhard Hollunder and Werner Nutt. Subsumption Algorithms for Concept Languages. DFKI Research Report RR-90-04. DFKI, Saarbrücken, 1990. Revised version of paper that was published at ECAI-90.
- F. Baader and U. Sattler. An Overview of Tableau Algorithms for Description Logics. *Studia Logica*, 69:5-40, 2001.
- I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239-264, May 2000.