

Principles of Knowledge Representation and Reasoning

Description Logics – Algorithms

Bernhard Nebel, Malte Helmert and Stefan Wöfl

Albert-Ludwigs-Universität Freiburg

July 22, 2008

Description Logics – Algorithms

Motivation

Structural Subsumption Algorithms

Tableau Subsumption Method

Motivation

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 Idea

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

$$D = \text{Human} \sqcap \exists \text{has-child} \sqcap \forall \text{has-child}.\text{Human} \sqcap \\ \forall \text{has-child}.\exists \text{has-child}$$

$$C = \text{Human} \sqcap \text{Female} \sqcap \exists \text{has-child} \sqcap \\ \forall \text{has-child}.\text{(Human} \sqcap \text{Female} \sqcap \exists \text{has-child)}$$

Check: $C \sqsubseteq D$

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

$\rightsquigarrow C$ is subsumed by D !

Subsumption Algorithm

SUB(C, D) algorithm:

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

$$C = \sqcap A_i \sqcap \sqcap \exists r_j \sqcap \sqcap \forall r_k : C_k$$

$$D = \sqcap B_l \sqcap \sqcap \exists s_m \sqcap \sqcap \forall s_n : D_n$$

2. For each B_l in D , is there an A_i in C with $A_i = B_l$?
3. For each $\exists s_m$ 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)

$$\text{SUB}(C, D) \Rightarrow C \sqsubseteq D$$

Proof sketch.

Reordering of terms (1):

- a) Commutativity and associativity are trivial
- b) Value restriction law. We show: $(\forall r.(C \sqcap D))^I = (\forall r.C \sqcap \forall r.D)^I$

Assumption: $d \in (\forall r.(C \sqcap D))^I$

Case 1: $\nexists e : (d, e) \in r^I \quad \checkmark$

Case 2: $\exists e : (d, e) \in r^I \Rightarrow e \in (C \sqcap D)^I \Rightarrow e \in C^I, e \in D^I$

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

Other direction is similar

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

Completeness

Theorem (Completeness)

$$C \sqsubseteq D \Rightarrow \text{SUB}(C, D)$$

Proof idea.

One shows the contrapositive:

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

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

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

Generalizing the Algorithm

Extensions of \mathcal{FL}^- by

- ▶ $\neg A$ (*atomic negation*),
- ▶ $(\leq nr)$, $(\geq nr)$ (*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

Hermaphrodite \doteq Male \sqcap Female

Parents-of-sons-and-daughters \doteq

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

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

Query

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

Parents-of-hermaphrodites

Reductions

1. Unfolding

$$\exists \text{has-child.Male} \sqcap \exists \text{has-child.Female} \sqsubseteq \\ \exists \text{has-child.}(\text{Male} \sqcap \text{Female})$$

2. Reduction to unsatisfiability

Is

$$\exists \text{has-child.Male} \sqcap \exists \text{has-child.Female} \sqcap \\ \neg(\exists \text{has-child.}(\text{Male} \sqcap \text{Female}))$$

unsatisfiable?

3. Negation normal form (move negations inside):

$$\exists \text{has-child.Male} \sqcap \exists \text{has-child.Female} \sqcap \\ \forall \text{has-child.}(\neg \text{Male} \sqcup \neg \text{Female})$$

4. Try to construct a model

Model Construction (1)

- Assumption:** There exists an object x in the interpretation of our concept:

$$x \in (\exists \dots)^{\mathcal{I}}$$

- This implies that x is in the interpretation of all conjuncts:

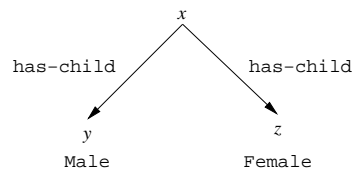
$$x \in (\exists \text{has-child.Male})^{\mathcal{I}}$$

$$x \in (\exists \text{has-child.Female})^{\mathcal{I}}$$

$$x \in (\forall \text{has-child.}(\neg \text{Male} \sqcup \neg \text{Female}))^{\mathcal{I}}$$

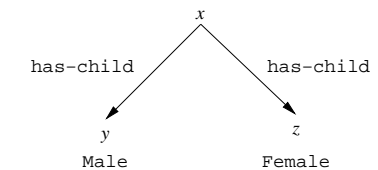
- This implies that there should be objects y and z such that $(x, y) \in \text{has-child}^{\mathcal{I}}$, $(x, z) \in \text{has-child}^{\mathcal{I}}$, $y \in \text{Male}^{\mathcal{I}}$ and $z \in \text{Female}^{\mathcal{I}}$ and ...

Model Construction (2)

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


-

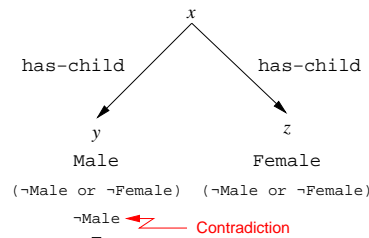
Model Construction (3)

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

 $(\neg \text{Male} \text{ or } \neg \text{Female}) \quad (\neg \text{Male} \text{ or } \neg \text{Female})$

-

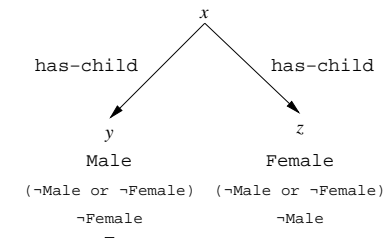
Model Construction (4)

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



Model Construction (5)

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



⇒ Model **constructed!**

Tableau Method (1): NNF

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

Now we have the following equivalences:

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

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

$$\neg\neg C \equiv C$$

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

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

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: $x : C$ or xry , where C is a concept description in NNF, r is a role name and x and y are **variable names**.

Let \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** $x : C$ (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 $\{x : C\}$ is satisfiable.

Tableau Method (3): Transforming Constraint Systems

Transformation rules:

1. $S \rightarrow_{\cap} \{x: C_1, x: C_2\} \cup S$
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. $S \rightarrow_{\sqcup} \{x: D\} \cup S$
if $(x: C_1 \sqcup C_2) \in S$ and neither $(x: C_1) \in S$ nor $(x: C_2) \in S$ and $D = C_1$ or $D = C_2$.
3. $S \rightarrow_{\exists} \{xry, y: C\} \cup S$
if $(x: \exists r.C) \in S$, y is a *fresh variable*, and there is no z s.t. $(xrz) \in S$ 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 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 $x: A$ and $x: \neg A$, 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. \square

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:

$$\exists r.A \cap \exists r.B \cap \forall r. \left(\exists r.A \cap \exists r.B \cap \forall r. (\exists r.A \cap \exists r.B \cap \forall r. (\dots)) \right)$$

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

-  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.