Principles of Knowledge Representation and Reasoning

Semantic Networks and Description Logics IV: Description Logics – Algorithms

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel, Stefan Wölfl, and Julien Hué

January 25, 2013

1 Motivation

Reasoning problems & algorithms

Reasoning problems:

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

Solving techniques presented in this chapter:

- 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
  - Often the methhod of choice

2 Structural Subsumption Algorithms

- Idea
- Example
- Algorithm
- Soundness
- Completeness
- Generalizations
- ABox Reasoning

Structural subsumption algorithms

In what follows we consider the rather small logic $\mathcal{FL}^-$:

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

To solve the subsumption problem for this logic we apply the following 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

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

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

Check: $C \sqsubseteq D$

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

$\rightsquigarrow$ C is subsumed by D !

Subsumption algorithm

SUB(C, D) algorithm:

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

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

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

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 $s_n = r_k$ and $C_k \sqsubseteq D_n$ (i.e., check SUB(C_k, D_n))?

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

Soundness

Theorem (Soundness)

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

Proof sketch.

Reordering of terms step (1):

1. Commutativity and associativity are trivial
2. Value restriction law. We show: $(\forall r.(C \sqcap D))^{\mathcal{I}} = (\forall r.C \sqcap \forall r.D)^{\mathcal{I}}$
   Assume $d \in (\forall r.(C \sqcap D))^{\mathcal{I}}$.
   If there is no $e \in \mathcal{D}$ with $(d,e) \in r^{\mathcal{I}}$ it follows trivially that $d \in (\forall r.C \sqcap \forall r.D)^{\mathcal{I}}$.
   If there is an $e \in \mathcal{D}$ with $(d,e) \in r^{\mathcal{I}}$ it follows $e \in (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$.
   Since e is arbitrary, we have $d \in (\forall r.C)^{\mathcal{I}}$ and $d \in (\forall r.D)^{\mathcal{I}}$,
   i.e., $(\forall r.(C \sqcap D))^{\mathcal{I}} \subseteq (\forall r.C \sqcap \forall r.D)^{\mathcal{I}}$.
   The other direction is similar.

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

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

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

□

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)

do 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.

3 Tableau Subsumption Method

- Example
- Reductions: Unfolding & Unsatisfiability
- Model Construction
- Equivalences & NNF
- Constraint Systems
- Transforming Constraint Systems
- Invariances
- Soundness and Completeness
- Space Complexity
- ABox Reasoning

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

Example

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

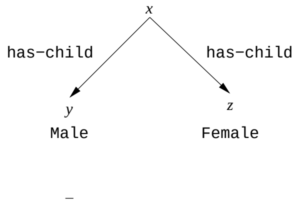
Model construction (1)

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:
$$x \in (\exists \texttt{has-child.Male})^{\mathcal{I}}$$
$$x \in (\exists \texttt{has-child.Female})^{\mathcal{I}}$$
$$x \in \left(\forall \texttt{has-child.}(\neg \texttt{Male} \sqcup \neg \texttt{Female})\right)^{\mathcal{I}}$$
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)

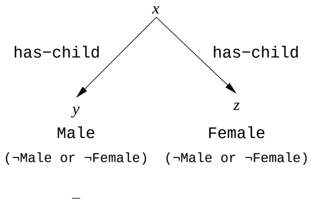
$x : \exists$`has-child.Male`

$x : \exists$`has-child.Female`



Model construction (3)

x : ∃has-child.Male
x : ∃has-child.Female
x : ∀has-child.(¬Male ⊔ ¬Female)



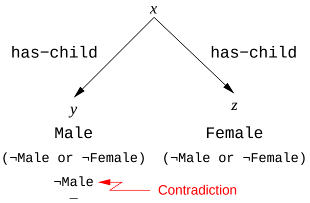
Model construction (4)

x : ∃has-child.Male

x : ∃has-child.Female

x : ∀has-child.(¬Male ⊔ ¬Female)

y : ¬Male



Model construction (5)

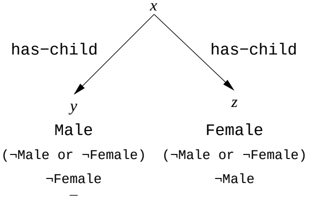
$x : \exists$has-child.Male

$x : \exists$has-child.Female

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

$y : \neg$Female

$z : \neg$Male



⇝ Model constructed!

Tableau method (1): NNF

We write: $C \equiv D$ iff $C \sqsubseteq D$ and $D \sqsubseteq C$. Now we have the following equivalences:

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

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

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

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:

$$x : C \quad \text{or} \quad x\, r\, y,$$

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

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

Definition

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 exist $\mathcal{I}$ and α such that α satisfies S.

Theorem

An $\mathcal{ALC}$ concept C in NNF is satisfiable if and only if the system $\{x : C\}$ is satisfiable.

Tableau method (4): Transforming constraint systems

Transformation rules:

1. $S \rightarrow_{\sqcap} \{x : C_1, x : C_2\} \cup S$
   if $(x : C_1 \sqcap C_2) \in S$ 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 $(y : C) \notin S$.

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

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

Tableau method (5): 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 if and only of 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 if and only if 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 (6): 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 if and only 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, applying the rules on a single constraint system can lead to constraint systems of exponential size.

Example

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

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

Idea: Generate a y only for one $\exists r.C$ and then proceed 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 I

Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider.
The Description Logic Handbook: Theory, Implementation, Applications,
Cambridge University Press, Cambridge, UK, 2003.

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.

Literature II

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.