Semantic Networks and Description Logics
Description Logics – Decidability and Complexity

Knowledge Representation and Reasoning

December XXX, 2005
Description Logics – Decidability and Complexity

Decidability & Undecidability

Polynomial Cases

Complexity of $\mathcal{ALC}$ Subsumption

Expressive Power vs. Complexity

The Complexity of Subsumption in TBoxes

Outlook
Decidability

$L_2$ is the fragment of first-order predicate logic using only two different variable names (note: variable names can be reused!).
$L_2^=$ the same including equality.

Theorem
$L_2^=$ is decidable.

Corollary
Subsumption and satisfiability of concept descriptions is decidable in description logics using only the following concept and role forming operators: $C \cap D, C \cup D, \neg C, \forall r.C, \exists r.C, r \sqsubseteq s, r \sqcap s, r \sqsupset s, \neg r, r^{-1}$.

Potential problems: Role composition and cardinality restrictions for role fillers. Cardinality restrictions, however, are not a real problem.
Undecidability

- $r \circ s$, $r \cap s$, $\neg r$, 1 [Schild 88]
- not relevant; Tarski had shown that already! – for relation algebras
- $r \circ s$, $r \sqsupseteq s$, $C \cap D$, $\forall r.C$ [Schmidt-Schauß 89]
- This is in fact a fragment of the early description logic \textit{KL-ONE}, where people had hoped to come up with a complete subsumption algorithm
Decidable, Polynomial-Time Cases

- \( \mathcal{FL}^- \) has obviously a polynomial subsumption problem (in the empty TBox) – the SUB algorithm needs only quadratic time.
- Donini et al [IJCAI 91] have shown that in the following languages subsumption can be decided using only polynomial time (and they are maximal wrt. this property):

\[
C \rightarrow A \mid \neg A \mid C \sqcap C' \mid \forall r.C \mid (\geq nr) \mid (\leq nr), \ r \rightarrow t \mid r^{-1}
\]

and

\[
C \rightarrow A \mid C \sqcap C' \mid \forall r.C \mid \exists r, \ r \rightarrow t \mid r^{-1} \mid r \sqcap r' \mid r \circ r'
\]

**Open:**

\[
C \rightarrow A \mid C \sqcap C' \mid \forall r.C \mid (\geq nr) \mid (\leq nr), \ r \rightarrow t \mid r \circ r'.
\]
How Hard is $\mathcal{ALC}$ Subsumption?

Proposition

$\mathcal{ALC}$ subsumption and unsatisfiability are co-NP-hard.

Proof.

Unsatisfiability and subsumption are reducible to each other. We give a reduction from UNSAT.

A propositional formula $\varphi$ over the atoms $a_i$ is mapped to $\pi(\varphi)$:

\[
\begin{align*}
a_i & \quad \mapsto \quad a_i \\
\psi \land \psi' & \quad \mapsto \quad \pi(\psi) \cap \pi(\psi') \\
\psi' \lor \psi & \quad \mapsto \quad \pi(\psi) \cup \pi(\psi') \\
\neg \psi & \quad \mapsto \quad \neg \pi(\psi)
\end{align*}
\]

Obviously, $\varphi$ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If $\varphi$ has a model, construct a model for $\pi(\varphi)$ with just one element $t$ standing for the truth of the atoms and the formula. Conversely, if $\pi(\varphi)$ satisfiable, pick one element $d \in \pi(\varphi)^I$ and set the truth value of atom $a_i$ according to the fact that $d \in \pi(a_i)^I$. \qed
How Hard Does It Get?

- Is $\mathcal{ALC}$ unsatisfiability and subsumption also complete for co-NP?
- Unlikely – since models of a single concept description can already become exponentially large!
- We will show PSPACE-completeness, whereby hardness is proved using a complexity result for (un)satisfiability in the modal logic $K$
- Satisfiability and unsatisfiability in $K$ is PSPACE-complete
Reduction from \( K \)-Satisfiability

Lemma (Lower bound for \( \mathcal{ALC} \))

\( \mathcal{ALC} \) subsumption, unsatisfiability and satisfiability are all PSPACE-hard.

Proof.

Extend the reduction given in the last proof by the following two rules – assuming that \( b \) is a fixed role name:

\[
\begin{align*}
\Box \varphi & \iff \forall b. \pi(\varphi) \\
\Diamond \varphi & \iff \exists b. \pi(\varphi)
\end{align*}
\]

Again, obviously, \( \varphi \) is satisfiable iff \( \pi(\varphi) \) is satisfiable (again using structural induction). If \( \varphi \) has a Kripke model, interpret each world \( w \) as an object in the universe of discourse that is an instances of the primitive concept \( \pi(a_i) \) iff \( a_i \) is true in \( w \). For the converse direction use the interpretation the other way around.
Complexity of \( \mathcal{ALC} \) Subsumption

Computational Complexity of \( \mathcal{ALC} \) Subsumption

Lemma (Upper Bound for \( \mathcal{ALC} \))

\( \mathcal{ALC} \) subsumption, unsatisfiability and satisfiability are all in PSPACE.

Proof.

This follows from the tableau algorithm for \( \mathcal{ALC} \). Although there may be exponentially many closed constraint systems, we can visit them step by step generating only one at a time. When closing a system, we have to consider only one role at a time – resulting in an only polynomial space requirement, i.e., satisfiability can be decided in PSPACE.

Theorem (Complexity of \( \mathcal{ALC} \))

\( \mathcal{ALC} \) subsumption, unsatisfiability and satisfiability are all PSPACE-complete.
In the reduction we used only one role symbol. Are there modal logics that would require more than one such role symbol?

The multi-modal logic $K_n$ has $n$ different Box operators $\Box_i$ (for $n$ different agents).

$\mathcal{ALC}$ is a notational variant of $K_n$ [Schild, IJCAI-91]

Are there perhaps other modal logics that correspond to other descriptions logics?

propositional dynamic logic (PDL), e.g., transitive closure, composition, role inverse, . . .

DL can be thought as fragments of first-order predicate logic. However, they are much more similar to modal logics

Algorithms and complexity results can be borrowed. Works also the other way around.
Expressive Power vs. Complexity

- Of course, one wants to have a description logic with high *expressive power*. However, high expressive power implies usually that the *computational complexity* of the reasoning problems might also be high, e.g., \(FL^-\) vs. \(ALC\).

- Does it make sense to use a language such as \(ALC\) or even extensions (corresponding to PDL) with higher complexity?

- There are three approaches to this problem:
  1. Use only *small* description logics with *complete* inference algorithms
  2. Use *expressive* description logics, but employ *incomplete* inference algorithms
  3. Use *expressive* description logics with *complete* inference algorithms

- For a long time, only options 1 and 2 were studied. Meanwhile, most researcher concentrate on *option 3*!
Is Subsumption in the Empty TBox Enough?

- We have shown that we can *reduce* concept subsumption in a given TBox to concept subsumption in the empty TBox.
- However, it is not obvious that this can be done in *polynomial time*.
- In particular, in the following example *unfolding* leads to an exponential blowup:

\[
\begin{align*}
C_1 &= \forall r.C_0 \sqcap \forall s.C_0 \\
C_2 &= \forall r.C_1 \sqcap \forall s.C_1 \\
&\vdots \\
C_n &= \forall r.C_{n-1} \sqcap \forall s.C_{n-1}
\end{align*}
\]

- Unfolding $C_n$ leads to a concept description with a size $\Omega(2^n)$.
- Is it possible to *avoid* this blowup?
- Can we avoid exponential preprocessing?
Question: Can we decide in polynomial time TBox subsumption for a description logic such as $\mathcal{FL}^-$, for which concept subsumption in the empty TBox can be decided in polynomial time?

Let us consider $\mathcal{FL}_0 : C \sqcap D, \forall r.C$ with terminological axioms.

Subsumption without a TBox can be done easily, using a structural subsumption algorithm.

Unfolding $+$ structural subsumption gives us an exponential algorithm.
Complexity of TBox Subsumption

Theorem (Complexity of TBox subsumption)

$TBox$ subsumption for $\mathcal{FL}_0$ is NP-hard.

Proof sketch.

We use the NDFA-equivalence problem, which is NP-complete for cycle-free automatons and PSPACE-complete for general NDFAs. We transform a cycle-free NDFA to a $\mathcal{FL}_0$-terminology with the mapping $\pi$ as follows:

- automaton $A \mapsto$ terminology $T_A$
- state $q \mapsto$ concept name $q$
- terminal state $q_f \mapsto$ concept name $q_f$
- input symbol $r \mapsto$ role name $r$

$r$-transition from $q$ to $q' \mapsto q \sqsubseteq \ldots \sqsubseteq \forall r : q' \sqsubseteq \ldots$
“Proof” by Example

\[ q_1 = \forall a.q_3 \sqcap \forall a.q_2 \]
\[ q_2 = \forall a.q_3 \sqcap \forall a.q_5 \]
\[ q_3 = \forall b.q_4 \]
\[ q_4 = \forall b.q_f \sqcap \forall c.q_f \]
\[ q_5 = \forall b.q_6 \]
\[ q_6 = \forall b.q_f \]
\[ q_1 \equiv \forall abc.q_f \sqcap \forall abb.q_f \sqcap \forall aabc.q_f \sqcap \forall aabb.q_f \]
\[ q_2 \equiv \forall abb.q_f \sqcap \forall abc.q_f \]
\[ q_1 \sqsubseteq_T q_2 \text{ and } \mathcal{L}(q_2) \subseteq \mathcal{L}(q_1) \]

In general, we have: \( \mathcal{L}(q) \subseteq \mathcal{L}(q') \) iff \( q' \sqsubseteq_T q \), from which the correctness of the reduction and the complexity result follows.
What Does This Complexity Result Mean?

- Note that for expressive languages such as $ALC$, we do not notice any difference!
- The TBox subsumption complexity result for less expressive languages does not play a large role *in practice*
- **Pathological situations** do not happen very often
- In fact, if the definition depth is logarithmic in the size of the TBox, the whole problem vanishes.
- However, in order to protect oneself against such problems, one often uses **lazy unfolding**
- Similarly, also for the $ALC$ concept descriptions, one notices that they are usually very well behaved.
Outlook

- Description logics have a long history (Tarski's relation algebras and Brachman's KL-ONE)
- Early on, either small languages with provably easy reasoning problems (e.g., the system CLASSIC) or large languages with incomplete inference algorithms (e.g., the system Loom) were used.
- Meanwhile, one uses complete algorithms on very large descriptions logics (e.g., SHIQ), e.g. in the systems FaCT and RACER
- RACER can handle KBs with up to 160,000 concepts (example from unified medical language system) in reasonable time (less than one day computing time)
- Description logics are used as the semantic backbone for OWL (a Web-language extending RDF)


