Principles of Knowledge Representation and Reasoning

Semantic Networks and Description Logics V: Description Logics – Decidability and Complexity

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel, Stefan Wölfl, and Julien Hué

January 30, 2013

Decidability & Undecidability

L_2 is the fragment of first-order predicate logic using only two different variable names (note: variable names can be reused!).
$L_2^=$: L_2 plus 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: $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, $\exists r.C$, $r \sqsubseteq s$, $r \sqcap s$, $r \sqcup s$, $\neg r$, r^{-1}.

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

Decidability

L_2 is the fragment of first-order predicate logic using only two different variable names (note: variable names can be reused!).
$L_2^=$: L_2 plus 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: $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, $\exists r.C$, $r \sqsubseteq s$, $r \sqcap s$, $r \sqcup s$, $\neg r$, r^{-1}.

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

Decidability

L_2 is the fragment of first-order predicate logic using only two different variable names (note: variable names can be reused!).
$L_2^=$: L_2 plus 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: $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, $\exists r.C$, $r \sqsubseteq s$, $r \sqcap s$, $r \sqcup s$, $\neg r$, r^{-1}.

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

Decidability

L_2 is the fragment of first-order predicate logic using only two different variable names (note: variable names can be reused!).
$L_2^=$: L_2 plus 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: $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, $\exists r.C$, $r \sqsubseteq s$, $r \sqcap s$, $r \sqcup s$, $\neg r$, r^{-1}.

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

Undecidability

- $r \circ s$, $r \sqcap s$, $\neg r$, 1 [Schild 88]
  ... already shown by Tarski (for relation algebras)
- $r \circ s$, $r \doteq s$, $C \sqcap D$, $\forall r.C$ [Schmidt-Schauß 89]
  ... This is, in fact, a fragment of the early description logic KL-ONE, where people had hoped to come up with a complete subsumption algorithm

Undecidability

- $r \circ s$, $r \sqcap s$, $\neg r$, 1 [Schild 88]
  … already shown by Tarski (for relation algebras)
- $r \circ s$, $r \doteq s$, $C \sqcap D$, $\forall r.C$ [Schmidt-Schauß 89]
  … This is, in fact, a fragment of the early description logic KL-ONE, where people had hoped to come up with a complete subsumption algorithm

Polynomial Cases

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:

$$C := A \mid \neg A \mid \top \mid \bot \mid C \sqcap C' \mid \forall r.C \mid (\geq n r) \mid (\leq n r),$$
$$r := t \mid r^{-1}$$

and

$$C := A \mid C \sqcap C' \mid \forall r.C \mid \exists r$$
$$r := t \mid r^{-1} \mid r \sqcap r' \mid r \circ r'$$

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:

$$C := A \,|\, \neg A \,|\, \top \,|\, \bot \,|\, C \sqcap C' \,|\, \forall r.C \,|\, (\geq n r) \,|\, (\leq n r),$$
$$r := t \,|\, r^{-1}$$

and

$$C := A \,|\, C \sqcap C' \,|\, \forall r.C \,|\, \exists r$$
$$r := t \,|\, r^{-1} \,|\, r \sqcap r' \,|\, r \circ r'$$

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:

$$C := A \mid \neg A \mid \top \mid \bot \mid C \sqcap C' \mid \forall r.C \mid (\geq n r) \mid (\leq n r),$$
$$r := t \mid r^{-1}$$

and

$$C := A \mid C \sqcap C' \mid \forall r.C \mid \exists r$$
$$r := t \mid r^{-1} \mid r \sqcap r' \mid r \circ r'$$

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:

$$C := A \mid \neg A \mid \top \mid \bot \mid C \sqcap C' \mid \forall r.C \mid (\geq n\, r) \mid (\leq n\, r),$$
$$r := t \mid r^{-1}$$

and

$$C := A \mid C \sqcap C' \mid \forall r.C \mid \exists r$$
$$r := t \mid r^{-1} \mid r \sqcap r' \mid r \circ r'$$

Complexity of $\mathcal{ALC}$ Subsumption

How hard is $\mathcal{ALC}$ subsumption?

UNI FREIBURG

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 φ over the atoms a_i is mapped to $\pi(\varphi)$:

$$
\begin{aligned}
a_i &\mapsto a_i \\
\psi \wedge \psi' &\mapsto \pi(\psi) \sqcap \pi(\psi') \\
\psi' \vee \psi &\mapsto \pi(\psi) \sqcup \pi(\psi') \\
\neg\psi &\mapsto \neg\pi(\psi)
\end{aligned}
$$

Obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If φ 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 $d \in \pi(\varphi)^{\mathcal{I}}$ and set the truth value of atom a_i according to the fact that $d \in \pi(a_i)^{\mathcal{I}}$. $\square$

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 φ over the atoms a_i is mapped to $\pi(\varphi)$:

$$
\begin{aligned}
a_i &\mapsto a_i \\
\psi \wedge \psi' &\mapsto \pi(\psi) \sqcap \pi(\psi') \\
\psi' \vee \psi &\mapsto \pi(\psi) \sqcup \pi(\psi') \\
\neg\psi &\mapsto \neg\pi(\psi)
\end{aligned}
$$

Obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If φ 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 $d \in \pi(\varphi)^{\mathcal{I}}$ and set the truth value of atom a_i according to the fact that $d \in \pi(a_i)^{\mathcal{I}}$. □

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 φ over the atoms a_i is mapped to $\pi(\varphi)$:

$$a_i \mapsto a_i$$
$$\psi \wedge \psi' \mapsto \pi(\psi) \sqcap \pi(\psi')$$
$$\psi' \vee \psi \mapsto \pi(\psi) \sqcup \pi(\psi')$$
$$\neg\psi \mapsto \neg\pi(\psi)$$

Obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If φ 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 $d \in \pi(\varphi)^{\mathcal{I}}$ and set the truth value of atom a_i according to the fact that $d \in \pi(a_i)^{\mathcal{I}}$. □

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 φ over the atoms a_i is mapped to $\pi(\varphi)$:

$$\begin{aligned} a_i &\mapsto a_i \\ \psi \wedge \psi' &\mapsto \pi(\psi) \sqcap \pi(\psi') \\ \psi' \vee \psi &\mapsto \pi(\psi) \sqcup \pi(\psi') \\ \neg\psi &\mapsto \neg\pi(\psi) \end{aligned}$$

Obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If φ 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 $d \in \pi(\varphi)^{\mathcal{I}}$ and set the truth value of atom a_i according to the fact that $d \in \pi(a_i)^{\mathcal{I}}$. $\square$

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 φ over the atoms a_i is mapped to $\pi(\varphi)$:

$$
\begin{aligned}
a_i &\mapsto a_i \\
\psi \wedge \psi' &\mapsto \pi(\psi) \sqcap \pi(\psi') \\
\psi' \vee \psi &\mapsto \pi(\psi) \sqcup \pi(\psi') \\
\neg\psi &\mapsto \neg\pi(\psi)
\end{aligned}
$$

Obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If φ 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 $d \in \pi(\varphi)^{\mathcal{I}}$ and set the truth value of atom a_i according to the fact that $d \in \pi(a_i)^{\mathcal{I}}$. □

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 φ over the atoms a_i is mapped to $\pi(\varphi)$:

$$
\begin{aligned}
a_i &\mapsto a_i \\
\psi \wedge \psi' &\mapsto \pi(\psi) \sqcap \pi(\psi') \\
\psi' \vee \psi &\mapsto \pi(\psi) \sqcup \pi(\psi') \\
\neg\psi &\mapsto \neg\pi(\psi)
\end{aligned}
$$

Obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If φ 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 $d \in \pi(\varphi)^{\mathcal{I}}$ and set the truth value of atom a_i according to the fact that $d \in \pi(a_i)^{\mathcal{I}}$. $\square$

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 φ over the atoms a_i is mapped to $\pi(\varphi)$:

$$\begin{aligned} a_i &\mapsto a_i \\ \psi \wedge \psi' &\mapsto \pi(\psi) \sqcap \pi(\psi') \\ \psi' \vee \psi &\mapsto \pi(\psi) \sqcup \pi(\psi') \\ \neg\psi &\mapsto \neg\pi(\psi) \end{aligned}$$

Obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (use structural induction). If φ 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 $d \in \pi(\varphi)^{\mathcal{I}}$ and set the truth value of atom a_i according to the fact that $d \in \pi(a_i)^{\mathcal{I}}$. □

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)satisifiability in the modal logic K.
- Satisifiability and unsatisfiability in K is PSPACE-complete.

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)satisifiability in the modal logic K.
- Satisifiability and unsatisfiability in K is PSPACE-complete.

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)satisifiability in the modal logic K.
- Satisifiability 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:

$$\Box\psi \mapsto \forall b.\pi(\psi)$$
$$\Diamond\psi \mapsto \exists b.\pi(\psi)$$

Again, obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (again using structural induction). If φ has a Kripke model, interpret each world w as an object in the universe of discourse, that is, w is an instance 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. $\square$

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:

$$\Box\psi \mapsto \forall b.\pi(\psi)$$
$$\Diamond\psi \mapsto \exists b.\pi(\psi)$$

Again, obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (again using structural induction). If φ has a Kripke model, interpret each world w as an object in the universe of discourse, that is, w is an instance 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. □

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:

$$\Box\psi \mapsto \forall b.\pi(\psi)$$
$$\Diamond\psi \mapsto \exists b.\pi(\psi)$$

Again, obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (again using structural induction). If φ has a Kripke model, interpret each world w as an object in the universe of discourse, that is, w is an instance 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. □

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:

$$\Box\psi \mapsto \forall b.\pi(\psi)$$
$$\Diamond\psi \mapsto \exists b.\pi(\psi)$$

Again, obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (again using structural induction). If φ has a Kripke model, interpret each world w as an object in the universe of discourse, that is, w is an instance 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. □

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:

$$\Box\psi \mapsto \forall b.\pi(\psi)$$
$$\Diamond\psi \mapsto \exists b.\pi(\psi)$$

Again, obviously, φ is satisfiable iff $\pi(\varphi)$ is satisfiable (again using structural induction). If φ has a Kripke model, interpret each world w as an object in the universe of discourse, that is, w is an instance 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. □

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.

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.

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.

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.

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.

Further consequences of the reducibility of K to $\mathcal{ALC}$

- In the reduction we used only one role symbol. Are there modal logics that would require more than one such role symbol?
  - $\leadsto$ The multi-modal logic K_n has n different Box operators $\Box_i$ (for n different agents).
  - $\leadsto$ $\mathcal{ALC}$ (wrt. TBox reasoning) is a notational variant of K_n. [Schild, IJCAI-91]
- Are there other modal logics that correspond to other descriptions logics?
  - $\leadsto$ propositional dynamic logic (PDL), e.g., transitive closure, composition, role inverse, …

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

$\leadsto$ Algorithms and complexity results can be borrowed. Works also the other way around.

Further consequences of the reducibility of K to $\mathcal{ALC}$

- In the reduction we used only one role symbol. Are there modal logics that would require more than one such role symbol?
  - $\rightsquigarrow$ The multi-modal logic K_n has n different Box operators $\Box_i$ (for n different agents).
  - $\rightsquigarrow$ $\mathcal{ALC}$ (wrt. TBox reasoning) is a notational variant of K_n. [Schild, IJCAI-91]
- Are there other modal logics that correspond to other descriptions logics?
  - $\rightsquigarrow$ propositional dynamic logic (PDL), e.g., transitive closure, composition, role inverse, …

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

$\rightsquigarrow$ Algorithms and complexity results can be borrowed. Works also the other way around.

Further consequences of the reducibility of K to $\mathcal{ALC}$

- 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).
  - ⇝ $\mathcal{ALC}$ (wrt. TBox reasoning) is a notational variant of K_n. [Schild, IJCAI-91]
- Are there 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.

Further consequences of the reducibility of K to $\mathcal{ALC}$

- 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).
  - ⇝ $\mathcal{ALC}$ (wrt. TBox reasoning) is a notational variant of K_n. [Schild, IJCAI-91]
- Are there 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.

Further consequences of the reducibility of K to $\mathcal{ALC}$

- 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).
  - ⇝ $\mathcal{ALC}$ (wrt. TBox reasoning) is a notational variant of K_n. [Schild, IJCAI-91]
- Are there 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

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., $\mathcal{FL}^-$ vs. $\mathcal{ALC}$.
- Does it make sense to use languages such as $\mathcal{ALC}$ 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!

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., $\mathcal{FL}^-$ vs. $\mathcal{ALC}$.
- Does it make sense to use languages such as $\mathcal{ALC}$ 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!

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., $\mathcal{FL}^-$ vs. $\mathcal{ALC}$.
- Does it make sense to use languages such as $\mathcal{ALC}$ 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!

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., $\mathcal{FL}^-$ vs. $\mathcal{ALC}$.
- Does it make sense to use languages such as $\mathcal{ALC}$ 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!

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., $\mathcal{FL}^-$ vs. $\mathcal{ALC}$.
- Does it make sense to use languages such as $\mathcal{ALC}$ 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!

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., $\mathcal{FL}^-$ vs. $\mathcal{ALC}$.
- Does it make sense to use languages such as $\mathcal{ALC}$ 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!

The Complexity of Subsumption in TBoxes

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 the following example unfolding leads to an exponential blowup:

$$
\begin{aligned}
C_1 &\doteq \forall r.C_0 \sqcap \forall s.C_0 \\
C_2 &\doteq \forall r.C_1 \sqcap \forall s.C_1 \\
&\vdots \\
C_n &\doteq \forall r.C_{n-1} \sqcap \forall s.C_{n-1}
\end{aligned}
$$

- 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?

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 the following example unfolding leads to an exponential blowup:

$$C_1 \doteq \forall r.C_0 \sqcap \forall s.C_0$$
$$C_2 \doteq \forall r.C_1 \sqcap \forall s.C_1$$
$$\vdots$$
$$C_n \doteq \forall r.C_{n-1} \sqcap \forall s.C_{n-1}$$

- 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?

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 the following example unfolding leads to an exponential blowup:

$$C_1 \doteq \forall r.C_0 \sqcap \forall s.C_0$$
$$C_2 \doteq \forall r.C_1 \sqcap \forall s.C_1$$
$$\vdots$$
$$C_n \doteq \forall r.C_{n-1} \sqcap \forall s.C_{n-1}$$

- 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?

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 the following example unfolding leads to an exponential blowup:

$$C_1 \doteq \forall r.C_0 \sqcap \forall s.C_0$$
$$C_2 \doteq \forall r.C_1 \sqcap \forall s.C_1$$
$$\vdots$$
$$C_n \doteq \forall r.C_{n-1} \sqcap \forall s.C_{n-1}$$

- 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?

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 the following example unfolding leads to an exponential blowup:

$$C_1 \doteq \forall r.C_0 \sqcap \forall s.C_0$$
$$C_2 \doteq \forall r.C_1 \sqcap \forall s.C_1$$
$$\vdots$$
$$C_n \doteq \forall r.C_{n-1} \sqcap \forall s.C_{n-1}$$

- 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?

TBox subsumption for small languages

- 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 + strucural subsumption gives us an exponential algorithm.

TBox subsumption for small languages

- 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 + strucural subsumption gives us an exponential algorithm.

TBox subsumption for small languages

- 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 + strucural subsumption gives us an exponential algorithm.

TBox subsumption for small languages

- 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 $+$ strucural subsumption gives us an exponential algorithm.

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 π as follows:

$$\begin{aligned} \text{automaton } A &\mapsto \text{terminology } \mathcal{T}_A \\ \text{state } q &\mapsto \text{concept name } q \\ \text{terminal state } q_f &\mapsto \text{concept name } q_f \\ \text{input symbol } r &\mapsto \text{role name } r \end{aligned}$$

r-transition from q to $q' \mapsto q \doteq \ldots \sqcap \forall r : q' \sqcap \ldots$. $\square$

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 π as follows:

$$\text{automaton } A \mapsto \text{terminology } \mathcal{T}_A$$
$$\text{state } q \mapsto \text{concept name } q$$
$$\text{terminal state } q_f \mapsto \text{concept name } q_f$$
$$\text{input symbol } r \mapsto \text{role name } r$$

r-transition from q to $q' \mapsto q \doteq \ldots \sqcap \forall r : q' \sqcap \ldots$. $\square$

Complexity of TBox subsumption

Theorem (Complexity of TBox subsumption)

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

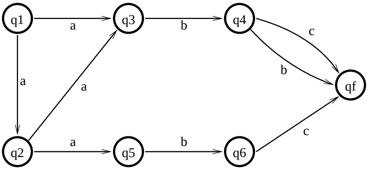
Proof sketch.

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 π as follows:

$$\begin{aligned} \text{automaton } A &\mapsto \text{terminology } \mathcal{T}_A \\ \text{state } q &\mapsto \text{concept name } q \\ \text{terminal state } q_f &\mapsto \text{concept name } q_f \\ \text{input symbol } r &\mapsto \text{role name } r \end{aligned}$$

r-transition from q to $q' \mapsto q \doteq \ldots \sqcap \forall r : q' \sqcap \ldots$. $\square$

"Proof" by example



$$q_1 \doteq \forall a.q_3 \sqcap \forall a.q_2$$

$$q_2 \doteq \forall a.q_3 \sqcap \forall a.q_5$$

$$q_3 \doteq \forall b.q_4$$

$$q_4 \doteq \forall b.q_f \sqcap \forall c.q_f$$

$$q_5 \doteq \forall b.q_6$$

$$q_6 \doteq \forall b.q_f$$

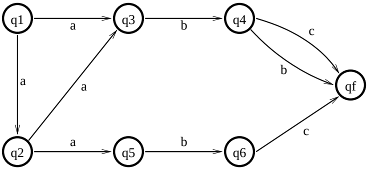
$$q_1 \equiv \forall abc.q_f \sqcap \forall abb.q_f \sqcap$$
$$\forall aabc.q_f \sqcap \forall aabb.q_f$$

$$q_2 \equiv \forall abb.q_f \sqcap \forall abc.q_f$$

$$q_1 \sqsubseteq_{\mathcal{T}} q_2 \quad \text{and} \quad \mathcal{L}(q_2) \subseteq \mathcal{L}(q_1)$$

In general, we have: $\mathcal{L}(q) \subseteq \mathcal{L}(q')$ iff $q' \sqsubseteq_{\mathcal{T}} q$, from which the correctness of the reduction and the complexity result follows.

"Proof" by example



$$q_1 \doteq \forall a.q_3 \sqcap \forall a.q_2$$

$$q_2 \doteq \forall a.q_3 \sqcap \forall a.q_5$$

$$q_3 \doteq \forall b.q_4$$

$$q_4 \doteq \forall b.q_f \sqcap \forall c.q_f$$

$$q_5 \doteq \forall b.q_6$$

$$q_6 \doteq \forall b.q_f$$

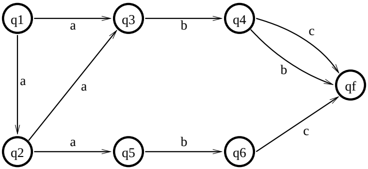
$$q_1 \equiv \forall abc.q_f \sqcap \forall abb.q_f \sqcap \forall aabc.q_f \sqcap \forall aabb.q_f$$

$$q_2 \equiv \forall abb.q_f \sqcap \forall abc.q_f$$

$$q_1 \sqsubseteq_{\mathcal{T}} q_2 \quad \text{and} \quad \mathcal{L}(q_2) \subseteq \mathcal{L}(q_1)$$

In general, we have: $\mathcal{L}(q) \subseteq \mathcal{L}(q')$ iff $q' \sqsubseteq_{\mathcal{T}} q$, from which the correctness of the reduction and the complexity result follows.

"Proof" by example



$$q_1 \doteq \forall a.q_3 \sqcap \forall a.q_2$$
$$q_2 \doteq \forall a.q_3 \sqcap \forall a.q_5$$
$$q_3 \doteq \forall b.q_4$$
$$q_4 \doteq \forall b.q_f \sqcap \forall c.q_f$$
$$q_5 \doteq \forall b.q_6$$
$$q_6 \doteq \forall b.q_f$$
$$q_1 \equiv \forall abc.q_f \sqcap \forall abb.q_f \sqcap \forall aabc.q_f \sqcap \forall aabb.q_f$$
$$q_2 \equiv \forall abb.q_f \sqcap \forall abc.q_f$$
$$q_1 \sqsubseteq_{\mathcal{T}} q_2 \quad \text{and} \quad \mathcal{L}(q_2) \subseteq \mathcal{L}(q_1)$$

In general, we have: $\mathcal{L}(q) \subseteq \mathcal{L}(q')$ iff $q' \sqsubseteq_{\mathcal{T}} q$, 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 $\mathcal{ALC}$, 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 $\mathcal{ALC}$ concept descriptions, one notices that they are usually very well behaved.

What does this complexity result mean?

- Note that for expressive languages such as $\mathcal{ALC}$, 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 $\mathcal{ALC}$ concept descriptions, one notices that they are usually very well behaved.

What does this complexity result mean?

- Note that for expressive languages such as $\mathcal{ALC}$, 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 $\mathcal{ALC}$ concept descriptions, one notices that they are usually very well behaved.

What does this complexity result mean?

- Note that for expressive languages such as $\mathcal{ALC}$, 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 $\mathcal{ALC}$ concept descriptions, one notices that they are usually very well behaved.

What does this complexity result mean?

- Note that for expressive languages such as $\mathcal{ALC}$, 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 $\mathcal{ALC}$ concept descriptions, one notices that they are usually very well behaved.

What does this complexity result mean?

- Note that for expressive languages such as $\mathcal{ALC}$, 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 $\mathcal{ALC}$ concept descriptions, one notices that they are usually very well behaved.

Outlook

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.
- Nowadays tools can handle KBs with up to 160,000 concepts (example from unified medical language system) in reasonable time.
- Description logics are used as the semantic backbone for OWL (a Web-language extending RDF).

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.
- Nowadays tools can handle KBs with up to 160,000 concepts (example from unified medical language system) in reasonable time.
- Description logics are used as the semantic backbone for OWL (a Web-language extending RDF).

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.
- Nowadays tools can handle KBs with up to 160,000 concepts (example from unified medical language system) in reasonable time.
- Description logics are used as the semantic backbone for OWL (a Web-language extending RDF).

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.
- Nowadays tools can handle KBs with up to 160,000 concepts (example from unified medical language system) in reasonable time.
- Description logics are used as the semantic backbone for OWL (a Web-language extending RDF).

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.
- Nowadays tools can handle KBs with up to 160,000 concepts (example from unified medical language system) in reasonable time.
- Description logics are used as the semantic backbone for OWL (a Web-language extending RDF).

Literature I

Bernhard Nebel and Gert Smolka.

Attributive description formalisms . . . and the rest of the world.

In: Otthein Herzog and Claus-Rainer Rollinger, editors, **Text Understanding in LILOG**, pages 439–452. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt.

Tractable concept languages.

In: **Proceedings of the 12th International Joint Conference on Artificial Intelligence**, pages 458–465, Sydney, Australia, August 1991. Morgan Kaufmann.

Literature II

Klaus Schild.

A correspondence theory for terminological logics: Preliminary report.

In **Proceedings of the 12th International Joint Conference on Artificial Intelligence**, pages 466–471, Sydney, Australia, August 1991. Morgan Kaufmann.

I. Horrocks, U. Sattler, and S. Tobies.

Reasoning with Individuals for the Description Logic SHIQ.

In: David MacAllester, ed., **Proceedings of the 17th International Conference on Automated Deduction (CADE-17)**, Germany, 2000. Springer Verlag.

B. Nebel.

Terminological Reasoning is Inherently Intractable,

Artificial Intelligence, **43**: 235-249, 1990.