Chapter 5 Complexity

Instead of analyzing the complexity of a particular algorithm, we here analyze the complexity of the resoning problem itself:

> How efficient can we expect any reasoning algorithm for a given problem to be, even on very difficult ("worst case") inputs.

We will concentrate on the basic reasoning problems satisfiability and subsumption for the sake of simplicity.

All results established in this chapter also apply to KB consistency.

Complexity classes:

$$\text{PTime} \subseteq \text{NP} \subseteq \text{PSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime}$$

Explain?

Hard for class $\mathcal{C}$: every problem in $\mathcal{C}$ can be reduced to it in polynomial time

Complete for class $\mathcal{C}$: hard for $\mathcal{C}$ and contained in $\mathcal{C}$.

Dresden

5.1 Concept satisfiability in $\mathcal{ALC}$

First, we show that concept satisfiability in $\mathcal{ALC}$

- with acyclic TBoxes is in PSpace;
- without TBoxes is PSpace-hard;

and thus PSpace-complete in both cases.

We can concentrate on the complexity of satisfiability since it immediately yields the complexity of subsumption:

- mutual polynomial-time reductions between satisfiability and non-subsumption (Theorem 2.19);
- for deterministic complexity classes (such as PSpace) subsumption thus has the same complexity as satisfiability.

Note: for nondeterministic complexity classes (NP and NExpTime), subsumption would have the complementary complexity to satisfiability (coNP and coNExpTime).

Dresden

The complexity upper bound

case of acyclic TBox

Without loss of generality we assume that the concept tested for satisfiability is a concept name:

> C is satisfiable w.r.t. a TBox $\mathcal{T}$ iff A is satisfiable w.r.t. $\mathcal{T} \cup \{A \equiv C\}$,
> where A is a fresh concept name

Negation normal form (NNF):

An acyclic TBox $\mathcal{T}$ is in NNF if negation is applied only to primitive concept names in $\mathcal{T}$, but neither to defined concept names nor to compound concepts.

→ In the expanded version of $\mathcal{T}$, all concept descriptions are in NNF.

Proposition 5.1 (NNF)

There is a polynomial time transformation of each acyclic TBox $\mathcal{T}$ into an acyclic TBox $\mathcal{T}'$ in NNF such that for all concept names A occurring in $\mathcal{T}$, A is satisfiable w.r.t. $\mathcal{T}$ iff A is satisfiable w.r.t. $\mathcal{T}'$.

Proof: blackboard.

The complexity upper bound

case of acyclic TBox

Simple TBox:

An acyclic TBox $\mathcal{T}$ is simple if all concept definitions are of the form

$A \equiv P,\ A \equiv \neg P,\ A \equiv B_1 \sqcap B_2,\ A \equiv B_1 \sqcup B_2,\ A \equiv \exists r.B_1,$ or $A \equiv \forall r.B_1$

where P is a primitive concept and B_1, B_2 are defined concept names.

Note: every simple TBox is in NNF.

Lemma 5.2 (simple TBox)

Let A_0 be a concept name. There is a polynomial time transformation of each acyclic TBox $\mathcal{T}$ into a simple TBox $\mathcal{T}'$ such that A_0 is satisfiable w.r.t. $\mathcal{T}$ iff A_0 is satisfiable w.r.t. $\mathcal{T}'$.

Proof: blackboard.

The lemma shows that it is sufficient to design an algorithm that tests satisfiability of a concept name w.r.t. a simple TBox.

The complexity upper bound

case of acyclic TBox

Definition 5.3 (type)

Let $\mathcal{T}$ be a simple TBox and $\mathsf{Def}(\mathcal{T})$ the set of defined concept names in $\mathcal{T}$.
A type for $\mathcal{T}$ is a set $\tau \subseteq \mathsf{Def}(\mathcal{T})$ that satisfies the following conditions:

1. $A \in \tau$ implies $B \notin \tau$, if $A \equiv P$ and $B \equiv \neg P$ in $\mathcal{T}$;
2. $A \in \tau$ implies $B \in \tau$ and $B' \in \tau$, if $A \equiv B \sqcap B' \in \mathcal{T}$;
3. $A \in \tau$ implies $B \in \tau$ or $B' \in \tau$, if $A \equiv B \sqcup B' \in \mathcal{T}$.

- Note the similarity with the S-types of Definition 3.12.
  The restriction to defined concepts rather than subconcepts is possible because the TBox is simple.
- There is also a similarity with the tableau algorithm for $\mathcal{ALC}$.
  Conditions (ii) and (iii) resemble the $\sqcap$- and the $\sqcup$-rule and Condition (i) resembles the clash condition.

Dresden

The complexity upper bound

case of acyclic TBox

The satisfiability algorithm tries to construct a tree models whose depth is bounded by the role depth of the input concept name.

Intuitively, this is the nesting depth of existential and universal restrictions in the unfolded version of the concept name.

Formally, the role depth of a defined concept name A is defined as follows:

- If $A \equiv (\neg)P \in \mathcal{T}$, then $\mathsf{rd}(A) = 0$.
- If $A \equiv B_1 * B_2 \in \mathcal{T}$ with $* \in \{\sqcap, \sqcup\}$, then $\mathsf{rd}(A) = \max(\mathsf{rd}(B_1), \mathsf{rd}(B_2))$.
- If $A \equiv Q\, r.B \in \mathcal{T}$ with $Q \in \{\exists, \forall\}$, then $\mathsf{rd}(A) = \mathsf{rd}(B) + 1$.

Note: this definition is well-founded since $\mathcal{T}$ is acyclic.

For $i \geq 0$, we define $\mathsf{Def}_i(\mathcal{T}) = \{A \in \mathsf{Def}(\mathcal{T}) \mid \mathsf{rd}(A) \leq i\}$.

The satisfiability algorithm

case of acyclic TBox

The following algorithm tests satisfiability of a concept name A_0 w.r.t. a simple TBox $\mathcal{T}$:

define procedure $\mathcal{ALC}$-Worlds$(A_0, \mathcal{T})$
  $i = \mathsf{rd}(A_0)$
  guess a set $\tau \subseteq \mathsf{Def}_i(\mathcal{T})$ with $A_0 \in \tau$
  recurse$(\tau, i, \mathcal{T})$

Corresponds to one application of $\exists$-rule and all possible applications of $\forall$-rule

define procedure recurse$(\tau, i, \mathcal{T})$
  if τ is not a type for $\mathcal{T}$ **then return** false
  if $i = 0$ **then return** true
  for all $A \in \tau$ with $A \equiv \exists r.B \in \mathcal{T}$ **do**
    $S = \{B\} \cup \{B' \mid \exists A' : A' \in \tau \text{ and } A' \equiv \forall r.B' \in \mathcal{T}\}$
    guess a set $\tau \subseteq \mathsf{Def}_{i-1}(\mathcal{T})$ with $S \subseteq \tau$
    if recurse$(\tau, i-1, \mathcal{T})$ = false **then return** false
  return true

Dresden

The satisfiability algorithm

terminates and runs in PSpace

To see that this algorithm always terminates and needs only polynomial space, we consider the recursion tree corresponding to the recursive calls of recurse:

A recursion tree is a tuple $T = (V, E, \ell)$, with

- (V, E) a tree whose nodes correspond to the calls of recurse;
- ℓ a node labelling function that assigns with each node $v \in V$ the arguments $\ell(v) = (\tau, i, \mathcal{T})$ of the recursive call corresponding to v;
- $(v, v') \in E$ if the call corresponding to v' occurred during v.

Termination: depth of recursion tree is bounded by $\mathsf{rd}(A_0) \leq \mathsf{size}(\mathcal{T})$,
outdegree is bounded by the number of concept definitions in $\mathcal{T}$.

In PSpace: #entries in recursion stack is bounded by depth of recursion tree,
size of each entry in recursion stack is bounded by size of $\mathcal{T}$.

Dresden

The satisfiability algorithm

is sound and complete

Lemma 5.4

$\mathcal{ALC}$-Worlds$(A_0, \mathcal{T}) = \mathsf{true}$ iff A_0 is satisfiable w.r.t. $\mathcal{T}$.

Proof: blackboard.

Theorem 5.5

In $\mathcal{ALC}$, concept satisfiability and subsumption w.r.t. acyclic TBoxes are in PSpace.

The complexity lower bound

case without TBox

To show that concept satisfiability in $\mathcal{ALC}$ is PSpace-hard without TBox

- we need to find a problem P that is known to be PSpace-complete,
- and show that P can be reduced to concept satisfiability.

The original PSpace-hardness proof by Schmidt-Schauß and Smolka (1991) reduced QBF (validity of Quantified Boolean Formulae) to concept satisfiability in $\mathcal{ALC}$.

> We use a different PSpace-complete problem for our reduction since a similar ExpTime-complete problem can be used to show ExpTime-hardness for satisfiability w.r.t. a general TBox.

The problem we use is a game played on formulae of propositional logic, called finite Boolean game (FBG).

Finite Boolean games

a known PSpace-complete problem

A finite Boolean game (FBG) is a triple $(\varphi, \Gamma_1, \Gamma_2)$ with

- φ a formula of propositional logic,
- $\Gamma_1 \uplus \Gamma_2$ a partition of the variables used in φ into two sets of identical cardinality.

The game is played by two players with alternating moves that determine the truth values of one propositional variable:

- Player 1 controls the variables in Γ_1 and tries to make the formula true;
- Player 2 controls the variables in Γ_2 and tries to make the formula false.

Decision problem: does Player 1 have a winning strategy,
i.e., can Player 1 force a win no matter what Player 2 does?

Dresden

Winning strategy

for a finite Boolean game $G = (\varphi, \Gamma_1, \Gamma_2)$

Let $n = |\Gamma_1 \uplus \Gamma_2|$, and $\Gamma_1 = \{p_1, p_3, \ldots, p_{n-1}\}$ and $\Gamma_2 = \{p_2, p_4, \ldots, p_n\}$.

Configuration of G: word $t \in \{0,1\}^i$, for some $i \leq n$

- game already played i steps;
- kth letter of t = truth value chosen for p_k.

Initial configuration of G: empty word ε

Move: if the current configuration is t, then a truth value for $p_{|t|+1}$ is selected

- by Player 1 if $|t|$ is even,
- by Player 2 if $|t|$ is odd.

Dresden

Winning strategy

for a finite Boolean game $G = (\varphi, \Gamma_1, \Gamma_2)$

Let $n = |\Gamma_1 \uplus \Gamma_2|$, and $\Gamma_1 = \{p_1, p_3, \ldots, p_{n-1}\}$ and $\Gamma_2 = \{p_2, p_4, \ldots, p_n\}$.

A winning strategy for Player 1 in G is a finite node-labelled tree (V, E, ℓ) of depth n, where

- ℓ assigns to each node $v \in V$ a configuration $\ell(v)$;
- the root is labelled with the initial configuration;
- if v is a node of depth $i < n$ with i even and $\ell(v) = t$, then v has one successor v' with $\ell(v') \in \{t0, t1\}$; *Player 1 must choose an appropriate move*
- if v is a node of depth $i < n$ with i odd and $\ell(v) = t$, then v has two successor v' and v' with $\ell(v') = t0$ and $\ell(v'') = t1$; *Player 1 must react to all possible moves of Player 2*
- if v is a node of depth n and $\ell(v) = t$, then t satisfies φ.

Dresden

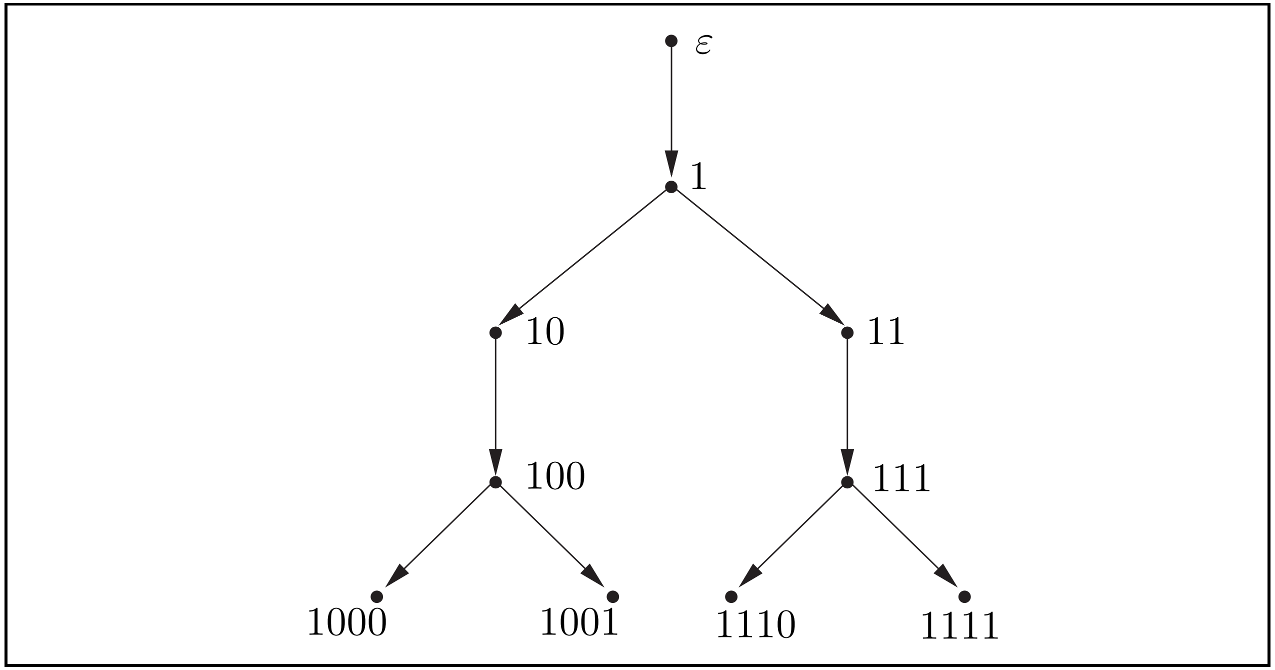
Winning strategy

example

Consider the game $G = (\varphi, \{p_1, p_3\}, \{p_2, p_4\})$, with

$$\varphi = (\neg p_1 \rightarrow p_2) \wedge ((p_1 \wedge p_2) \rightarrow (p_3 \vee p_4)) \wedge (\neg p_2 \rightarrow (p_4 \rightarrow \neg p_3)).$$

The following is a winning strategy for Player 1 in G:



The reduction

from FBG to concept satisfiability

$G = (\varphi, \Gamma_1, \Gamma_2) \xrightarrow{\text{polynomial}} \mathcal{ALC}$ concept C_G

Player 1 has winning strategy iff C_G is satisfiable

Idea: winning strategies are the tree models of C_G

Role names: we use one role name r as edge relation for the tree

Concept names: for each propositional variable p_i $(1 \leq i \leq n)$ a concept name P_i

The reduction

from FBG to concept satisfiability

C_G is a conjunction of the following concept descriptions:

- For each node of odd depth i (i.e., Player 2 is to move), there are two successors, one for each possible truth value of p_{i+1}:

$$C_1 = \bigsqcap_{i \in \{1,3,\ldots,n-1\}} \forall r^i.(\,\exists r.\neg P_{i+1} \sqcap \exists r.P_{i+1}\,)$$

where $\forall r^i.C$ denotes the i-fold nesting $\forall r.\cdots\forall r.C$.

- For each node of even depth i (i.e., Player 1 is to move), there is one successor:

$$C_2 = \bigsqcap_{i \in \{0,2,\ldots,n-2\}} \forall r^i.\exists r.\top$$

Since the generated successor either belongs to P_{i+1} or its negation, a truth value for p_{i+1} is chosen "automatically".

Dresden

The reduction

from FBG to concept satisfiability

C_G is a conjunction of the following concept descriptions:

- Once a truth value is chosen, it remains fixed:

$$C_3 = \bigsqcap_{1 \leq i \leq j < n} \forall r^j . \big((P_i \Rightarrow \forall r . P_i) \sqcap (\neg P_i \Rightarrow \forall r . \neg P_i) \big)$$

Recall: $C \Rightarrow D$ abbreviates $\neg C \sqcup D$

- At the leafs, the formula φ is true:

$$C_4 = \forall r^n . \varphi^*$$

φ^* denotes the result of converting φ into an $\mathcal{ALC}$ concept:

$$p_i \to P_i, \qquad \wedge \to \sqcap, \qquad \vee \to \sqcup$$

$$\boxed{C_G = C_1 \sqcap \cdots \sqcap C_4}$$

Dresden

The reduction

from FBG to concept satisfiability

Fact:

The size of C_1, C_2, C_3 is quadratic in n and the size of C_4 is linear in n plus the size of φ.

Thus, C_G can be constructed in time polynomial in the description of G.

Lemma 5.6

Player 1 has a winning strategy in G iff C_G is satisfiable.

Proof: blackboard.

Theorem 5.7

In $\mathcal{ALC}$, concept satisfiability and subsumption without TBoxes and with acyclic TBoxes are PSpace-complete.

The complexity upper bound

case of general TBox

Without loss of generality we restrict the attention to satisfiability of a concept name A_0 w.r.t. a general TBox $\mathcal{T}$ in which this name occurs:

$$C \text{ is satisfiable w.r.t. } \mathcal{T} \text{ iff } A_0 \text{ is satisfiable w.r.t. } \mathcal{T} \cup \{A_0 \sqsubseteq C\}$$

A_0 new name

In addition, we can assume that the TBox consists of a single GCI of the form $\top \sqsubseteq C_{\mathcal{T}}$:

$$\mathcal{I} \models \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\} \text{ iff } \mathcal{I} \models \{\top \sqsubseteq (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)\}$$

for all interpretations $\mathcal{I}$

We can also assume without loss of generality that the concept $C_{\mathcal{T}}$ in this GCI is in NNF.

Dresden

The complexity upper bound

case of general TBox

We prove an ExpTime upper bound for satisfiability w.r.t. general $\mathcal{ALC}$ TBoxes using a so-called type elimination algorithm.

Definition 5.8 (type)

Let $\mathcal{T}$ be a general TBox satisfying the restrictions described above.
A type for $\mathcal{T}$ is a set $\tau \subseteq \mathsf{sub}(\mathcal{T})$ satisfying the following conditions:

(i) $A \in \tau$ implies $\neg A \notin \tau$, for all $\neg A \in \mathsf{sub}(\mathcal{T})$;

(ii) $C \sqcap D \in \tau$ implies $C \in \tau$ and $D \in \tau$, for all $C \sqcap D \in \mathsf{sub}(\mathcal{T})$;

(iii) $C \sqcup D \in \tau$ implies $C \in \tau$ or $D \in \tau$, for all $C \sqcup D \in \mathsf{sub}(\mathcal{T})$;

(iv) $C_{\mathcal{T}} \in \tau$.

Obiviously, the number of types is at most exponential in the size of $\mathcal{T}$.

The complexity upper bound

case of general TBox

Type elimination starts with the set of all types, and iteratively removes types whose existential restrictions are not realized by the current set of types.

Such types are called bad.

Definition 5.9 (bad type)

Let Γ be a set of types and $\tau \in \Gamma$.

Then τ is bad in Γ if there exists an $\exists r.C \in \tau$ such that the set

$$S = \{C\} \cup \{D \mid \forall r.D \in \tau\}$$

is no subset of any type in Γ.

Type elimination

the algorithm

define procedure $\mathcal{ALC}$-Elim$(A_0, \mathcal{T})$
  set Γ_0 to the set of all types for $\mathcal{T}$
  $i = 0$
  repeat
    $i = i + 1$
    $\Gamma_i = \{\tau \in \Gamma_{i-1} \mid \tau \text{ is not bad in } \Gamma_{i-1}\}$
  until $\Gamma_i = \Gamma_{i-1}$
  if there is $\tau \in \Gamma_i$ with $A_0 \in \tau$ **then return** true
  else return false

Lemma 5.10

$\mathcal{ALC}$-Elim$(A_0, \mathcal{T})$ = true iff A_0 is satisfiable w.r.t. $\mathcal{T}$.

Proof: blackboard.

The complexity upper bound

case of general TBox

It remains to show that this algorithm runs in exponential time:

(i) the number of types for $\mathcal{T}$ is exponential in the size of $\mathcal{T}$;

(ii) in each execution of the repeat loop, at least one type is eliminated;

(iii) computing the set Γ_i inside the repeat loop can be done in time polynomial in the cardinality of Γ_{i-1}
(thus in time exponential in the size of $\mathcal{T}$).

Theorem 5.11

In $\mathcal{ALC}$, concept satisfiability and subsumption w.r.t. general TBoxes are in ExpTime.

The complexity lower bound

case of general TBox

We show ExpTime-hardness by reducing existence of a winning strategy for infinite Boolean games to satisfiability w.r.t. general TBoxes.

An infinite Boolean game (IBG) is a quadruple $(\varphi, \Gamma_1, \Gamma_2), t_0$ with

- φ a formula of propositional logic,
- $\Gamma_1 \uplus \Gamma_2$ a partition of the variables used in φ into two sets,
- t_0 an initial truth value assignment to the variables in φ.

The game is played by two players with alternating moves in which the the truth value of a variable controlled by this player can be flipped or left unchanged:

- Player 1 wins if the formula ever becomes true during the game;
- Player 2 wins if the game runs forever without the formula ever becoming true.

Dresden

Winning strategy

for an infinite Boolean game $G = (\varphi, \Gamma_1, \Gamma_2, t_0)$

Configuration of G: (i, t) with

- $i \in \{1, 2\}$ determining the player to move next;
- t a truth assignment for the variables in $\Gamma_1 \uplus \Gamma_2$.

Initial configuration of G: $(1, t_0)$

A truth assignment t' is a p-variation of a truth assignment t, for $p \in \Gamma_1 \cup \Gamma_2$,

- if $t' = t$
- or t' is obtained from t by flipping the truth value of p.

It is a Γ_i-variation of t if it is a p-variation of t for some $p \in \Gamma_i$, $i \in \{1, 2\}$.

Dresden

Winning strategy

for an infinite Boolean game $G = (\varphi, \Gamma_1, \Gamma_2, t_0)$

A winning strategy for Player 2 in G is an infinite node-labelled tree (V, E, ℓ), where

- ℓ assigns to each node $v \in V$ a configuration $\ell(v)$;
- the root is labelled with the initial configuration;
- if $\ell(v) = (2, t)$, then v has one successor v' with $\ell(v') = (1, t')$ for a Γ_2-variation t' of t.

  Player 2 must choose an appropriate move
- if $\ell(v) = (1, t)$, then v has successors $v_0, \ldots, v_{|\Gamma_1|}$ with labels $\ell(v_i) = (2, t_i)$ $(0 \leq i \leq |\Gamma_1|)$ such that $t_0, \ldots, t_{|\Gamma_1|}$ are all Γ_1-variations of t;

  Player 2 must react to all possible moves of Player 1
- if v is a node with $\ell(v) = (i, t)$, then t does not satisfy φ.

Dresden

The reduction

from IBG to concept satisfiability w.r.t. a general TBox

$G = (\varphi, \Gamma_1, \Gamma_2, t_0)$	$\xrightarrow{\text{polynomial}}$	$\mathcal{ALC}$ TBox $\mathcal{T}_G$ and concept name I
Player 2 has winning strategy	iff	I is satisfiable w.r.t. $\mathcal{T}_G$

Idea: winning strategies are the tree models of I w.r.t. $\mathcal{T}_G$

Role names: we use one role name r as edge relation for the tree

Concept names: for each propositional variable p_i $(1 \leq i \leq n)$ a concept name P_i

T_1, T_2 to describe whether it is the turn of Player 1 or Player 2

$F_1, \ldots, F_n$ to indicate which variable has been flipped to reach the current configuration

The reduction

from IBG to concept satisfiability w.r.t. a general TBox

We assume $\Gamma_1 = \{p_1, \ldots, p_m\}$ and $\Gamma_2 = \{p_{m+1}, \ldots, p_n\}$.

$\mathcal{T}_G$ consists of the following GCIs:

- The initial configuration is as required:

$$I \sqsubseteq T_1 \sqcap \bigsqcap_{1 \leq i \leq n,\ t_0(p_i)=0} \neg P_i \sqcap \bigsqcap_{1 \leq i \leq n,\ t_0(p_i)=1} P_i$$

- If it is the turn of Player 1, then there are $|\Gamma_1|+1$ successors:

$$T_1 \sqsubseteq \exists r.(\neg F_{\sqcap} \cdots \sqcap \neg F_n) \sqcap \bigsqcap_{1 \leq i \leq m} \exists r.F_i$$

- If it is the turn of Player 2, then there is one successor:

$$T_2 \sqsubseteq \exists r.(\neg F_1 \sqcap \cdots \sqcap \neg F_n) \sqcup \bigsqcup_{m < i \leq n} \exists r.F_i$$

- At most one variable is flipped in each move:

$$\top \sqsubseteq \bigsqcap_{1 \leq i < j \leq n} \neg(F_i \sqcap F_j)$$

Dresden

The reduction

from IBG to concept satisfiability w.r.t. a general TBox

We assume $\Gamma_1 = \{p_1, \ldots, p_m\}$ and $\Gamma_2 = \{p_{m+1}, \ldots, p_n\}$.

$\mathcal{T}_G$ consists of the following GCIs:

- Variables that are flipped change their truth value:

  $$\top \sqsubseteq \bigsqcap_{1 \leq i \leq n} \Big((\, P_i \rightarrow \forall r.(F_i \rightarrow \neg P_i)\,) \sqcap (\, \neg P_i \rightarrow \forall r.(F_i \rightarrow P_i)\,) \Big)$$

- Variables that are not flipped keep their truth value:

  $$\top \sqsubseteq \bigsqcap_{1 \leq i \leq n} \Big((\, P_i \rightarrow \forall r.(\neg F_i \rightarrow P_i)\,) \sqcap (\, \neg P_i \rightarrow \forall r.(\neg F_i \rightarrow \neg P_i)\,) \Big)$$

- The players alternate:

  $$T_1 \sqsubseteq \forall r.T_2 \qquad \text{and} \qquad T_2 \sqsubseteq \forall r.T_1$$

- The formula φ is never satisfied: $\top \sqsubseteq \neg\varphi^*$

  where φ^* denote the result of converting φ into an $\mathcal{ALC}$ concept.

Dresden

The reduction

from IBG to concept satisfiability w.r.t. a general TBox

It is easy to see that $\mathcal{T}_G$ can be constructed in time polynomial in the description of G.

Lemma 5.12

Player 2 has a winning strategy in G iff I is satisfiable w.r.t $\mathcal{T}_G$.

Proof: blackboard.

Theorem 5.13

In $\mathcal{ALC}$, concept satisfiability and subsumption w.r.t. general TBoxes is ExpTime-complete.

Dresden

5.2 Concept satisfiability in $\mathcal{ALCOI}$

Inverse roles: if r is a role, then r^- denotes its inverse $\mathcal{I}$

$$(r^-)^\mathcal{I} := \{(e,d) \mid (d,e) \in r^\mathcal{I}\}$$

Nominals: $\{a\}$ for $a \in \mathbf{I}$ with semantics $\mathcal{O}$

$$\{a\}^\mathcal{I} \quad := \quad \{a^\mathcal{I}\}$$

Using type elimination, it is not hard to show that satisfiability w.r.t. general TBoxes in $\mathcal{ALCOI}$ remains in ExpTime.

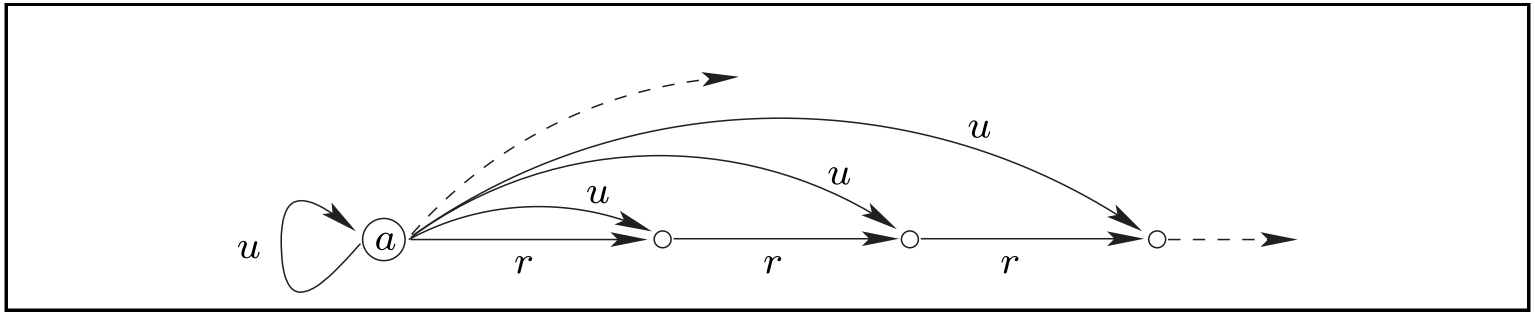
We show that concept satisfiability in $\mathcal{ALCOI}$ is ExpTime-hard already without TBox.

First, note that the interaction between inverse roles and nominals allows us to enforce infinite chains of role successors:

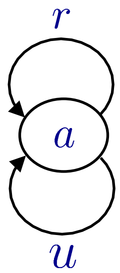
Example: $C = \{a\} \sqcap \exists u.\{a\} \sqcap \forall u.\exists r.\exists u^-.\{a\}.$

Dresden

5.2 Concept satisfiability in $\mathcal{ALCOI}$



possibly cyclic



First, note that the interaction between inverse roles and nominals allows us to enforce infinite chains of role successors:

Example: $C = \{a\} \sqcap \exists u.\{a\} \sqcap \forall u.\exists r.\exists u^{-}.\{a\}$.

Dresden

ExpTime-hardness of concept satisfiability in $\mathcal{ALCOI}$

More generally, the interaction between inverse roles and nominals allows us to enforce that all elements of the domain are reachable via some role u from some nominal a.

> GCIs $C \sqsubseteq D$ can then be propagated to all elements of the domain by adding a value restriction $\forall u.(\neg C \sqcup D)$ to a.

More precisely, we reduce satisfiability of an $\mathcal{ALC}$ concept w.r.t. an $\mathcal{ALC}$ TBox to concept satisfiability in $\mathcal{ALCOI}$:

Given an $\mathcal{ALC}$ concept C_0 and an $\mathcal{ALC}$ TBox $\mathcal{T}$, we define

$$D_0 = C_0 \sqcap \{a\} \sqcap \exists u.\{a\} \sqcap \forall u.(\bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D) \sqcap \forall u.(\bigsqcap_{i<k} \forall r_i.\exists u^-.\{a\}),$$

where $r_0, \ldots, r_{k-1}$ are all role names occurring in C and $\mathcal{T}$ and their inverses, and u is a fresh role name.

Dresden

ExpTime-hardness of concept satisfiability in $\mathcal{ALCOI}$

Given an $\mathcal{ALC}$ concept C_0 and an $\mathcal{ALC}$ TBox $\mathcal{T}$, we define

$$D_0 = C_0 \sqcap \{a\} \sqcap \exists u.\{a\} \sqcap \forall u.(\bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D) \sqcap \forall u.(\bigsqcap_{i<k} \forall r_i.\exists u^{-}.\{a\}),$$

where $r_0, \ldots, r_{k-1}$ are all role names occurring in C and $\mathcal{T}$ and their inverses, and u is a fresh role name.

It remains to show that this reduction is correct, i.e.,

C_0 is satisfiable w.r.t. $\mathcal{T}$ iff D_0 is satisfiable

Proof: blackboard.

Theorem 5.15

In $\mathcal{ALCOI}$, concept satisfiability and subsumption (without TBoxes) are ExpTime-hard.

Dresden

5.3 Undecidable extensions of $\mathcal{ALC}$

Role value maps were available as concept constructors already in the first DL system, KL-ONE.

Syntax: $(r_1 \circ \cdots \circ r_k \sqsubseteq s_1 \circ \cdots \circ s_\ell)$,

where $r_1, \ldots, r_k$ and $s_1, \ldots, s_\ell$ are role names.

Semantics: we define

$$(r_1 \circ \cdots \circ r_k)^\mathcal{I}(d_0) = \{d_k \in \Delta^\mathcal{I} \mid \exists d_1, \ldots, d_{k-1} : (d_i, d_{i+1}) \in r_i^\mathcal{I} \text{ for } 0 \leq i < k\}.$$

and

$$(r_1 \circ \cdots \circ r_k \sqsubseteq s_1 \circ \cdots \circ s_\ell)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid (r_1 \cdots r_k)^\mathcal{I}(d) \subseteq (s_1 \cdots s_\ell)^\mathcal{I}(d)\}.$$

Role value maps

example

Consider the following part of a TBox about universities:

$$\begin{array}{rcl} \mathsf{Course} & \sqsubseteq & \exists \mathsf{held\text{-}at}.\mathsf{University} \\ \mathsf{Lecturer} & \sqsubseteq & \exists \mathsf{teaches}.\mathsf{Course} \sqcap \exists \mathsf{employed\text{-}by}.\mathsf{University} \end{array}$$

To express that someone who teaches a course held at a university must be employed by that specific university, we need role value maps:

$$\top \sqsubseteq (\mathsf{teaches} \circ \mathsf{held\text{-}at} \sqsubseteq \mathsf{employed\text{-}by}).$$

Though very useful, role value maps are not available in modern DL systems since they cause undecidability:

- We first show undecidability in the presence of GCIs,
- and then strengthen this result by showing that undecidability already holds without GCIs.

Role value maps

cause undecidability

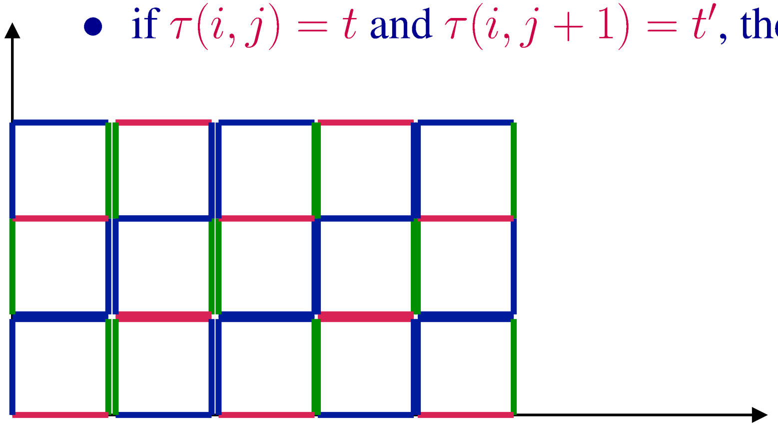
To show undecidability we reduce a known undecidable problem to satisfiability in the extension of $\mathcal{ALC}$ with role value maps.

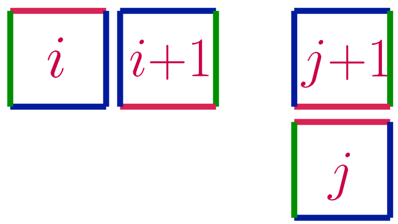
Definition 5.19 (tiling problem)

A tiling problem is a triple $P = (T, H, V)$, where T is a finite set of tile types and $H, V \subseteq T \times T$ represent the horizontal and vertical matching conditions.

A mapping $\tau : \mathbb{N} \times \mathbb{N} \rightarrow T$ is a solution for P if for all $i, j \geq 0$, the following holds:

- if $\tau(i, j) = t$ and $\tau(i + 1, j) = t'$, then $(t, t') \in H$;
- if $\tau(i, j) = t$ and $\tau(i, j + 1) = t'$, then $(t, t') \in V$.





Role value maps

cause undecidability

To show undecidability we reduce a known undecidable problem to satisfiability in the extension of $\mathcal{ALC}$ with role value maps.

Definition 5.19 (tiling problem)

A tiling problem is a triple $P = (T, H, V)$, where T is a finite set of tile types and $H, V \subseteq T \times T$ represent the horizontal and vertical matching conditions.

A mapping $\tau : \mathbb{N} \times \mathbb{N} \to T$ is a solution for P if for all $i, j \geq 0$, the following holds:

- if $\tau(i, j) = t$ and $\tau(i+1, j) = t'$, then $(t, t') \in H$;
- if $\tau(i, j) = t$ and $\tau(i, j+1) = t'$, then $(t, t') \in V$.

Decision problem

Given a tiling problem P

Question does P have a solution?

is known to be undecidable.

Dresden

The reduction

Given a tiling problem $P = (T, H, V)$, we construct a general TBox $\mathcal{T}_P$ with role value maps such that models of $\mathcal{T}_P$ represent solutions to P.

Concept names: for each tile $t \in T$ $(1 \leq i \leq n)$ a concept name A_t

Role names: r_x and r_y for the horizontal and vertical successor relations

The TBox $\mathcal{T}_P$ consists of the following GCIs:

(i) Every position has a horizontal and a vertical successor:

$$\top \sqsubseteq \exists r_x.\top \sqcap \exists r_y.\top$$

(ii) Every position is labelled with exactly one tile type:

$$\top \sqsubseteq \bigsqcup_{t \in T} A_t \sqcap \bigsqcap_{t,t' \in T, t \neq t'} \neg(A_t \sqcap A_{t'})$$

(iii) Adjacent tiles satisfy the matching conditions:

$$\top \sqsubseteq \bigsqcup_{(t,t') \in H} (A_t \sqcap \forall r_x.A_{t'}) \sqcap \bigsqcup_{(t,t') \in V} (A_t \sqcap \forall r_y.A_{t'})$$

Dresden

The reduction continued

The TBox $\mathcal{T}_P$ consists of the following GCIs:

(i) Every position has a horizontal and a vertical successor:

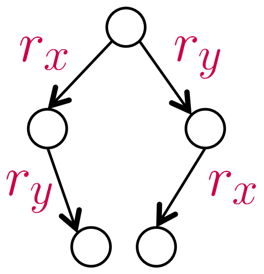
$$\top \sqsubseteq \exists r_x.\top \sqcap \exists r_y.\top$$

(ii) Every position is labelled with exactly one tile type:

$$\top \sqsubseteq \bigsqcup_{t \in T} A_t \sqcap \bigsqcap_{t,t' \in T, t \neq t'} \neg(A_t \sqcap A_{t'})$$

(iii) Adjacent tiles satisfy the matching conditions:

$$\top \sqsubseteq \bigsqcup_{(t,t') \in H} (A_t \sqcap \forall r_x.A_{t'}) \sqcap \bigsqcup_{(t,t') \in V} (A_t \sqcap \forall r_y.A_{t'})$$



no connection

Dresden

The reduction

continued

The TBox $\mathcal{T}_P$ consists of the following GCIs:

(i) Every position has a horizontal and a vertical successor:

$$\top \sqsubseteq \exists r_x.\top \sqcap \exists r_y.\top$$

(ii) Every position is labelled with exactly one tile type:

$$\top \sqsubseteq \bigsqcup_{t \in T} A_t \sqcap \bigsqcap_{t,t' \in T, t \neq t'} \neg(A_t \sqcap A_{t'})$$

(iii) Adjacent tiles satisfy the matching conditions:

$$\top \sqsubseteq \bigsqcup_{(t,t') \in H} (A_t \sqcap \forall r_x.A_{t'}) \sqcap \bigsqcup_{(t,t') \in V} (A_t \sqcap \forall r_y.A_{t'})$$

(iv) Every $r_x r_y$-successor is also a $r_y r_x$-successor and vice versa:

$$\top \sqsubseteq (r_x \circ r_y \sqsubseteq r_y \circ r_x)$$
$$\top \sqsubseteq (r_y \circ r_x \sqsubseteq r_x \circ r_y)$$

Dresden

Role value maps

cause undecidability

Lemma 5.20 (correctness of the reduction)

$\top$ is satisfiable w.r.t. $\mathcal{T}_P$ iff P has a solution.

Proof: blackboard.

Theorem 5.21

In the extension of $\mathcal{ALC}$ with role value maps, concept satisfiability and subsumption w.r.t. general TBoxes are undecidable.

Theorem 5.22

In the extension of $\mathcal{ALC}$ with role value maps, concept satisfiability and subsumption (without TBoxes) are undecidable.

Proof: blackboard.