Description Logics

Franz Baader
Theoretical Computer Science
TU Dresden
Germany

1. Motivation and introduction to Description Logics
2. Tableau-based reasoning procedures
3. Automata-based reasoning procedures
4. Complexity of reasoning in Description Logics
5. Reasoning in inexpressive Description Logics

Dresden

Reasoning procedures

requirements

1. The procedure should be a decision procedure for the problem.
2. The procedure should be as efficient as possible:
   preferably optimal w.r.t. the (worst-case) complexity of the problem
3. The procedure should be practical:
   easy to implement and optimize, and behave well in applications

Given a DL (like $\mathcal{ALC}$) and an inference problem (like satisfiability) one must answer the following questions:

- Is the inference problem decidable for this DL?
- If yes, how complex is the problem?

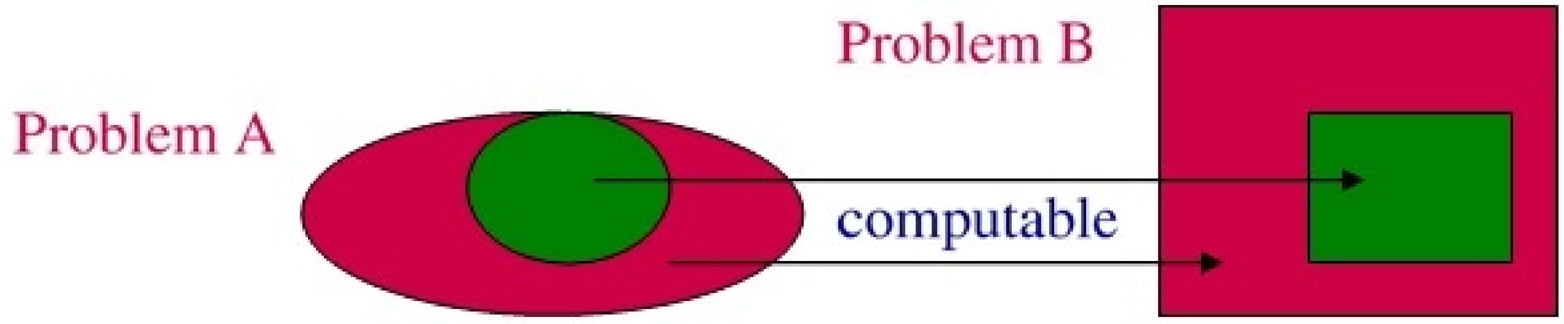
Dresden

(Un)decidability of a problem

- To show that a problem is decidable, it is enough to describe a decision procedure,

  and prove that it is one (sound, complete, terminating).

- To show that a problem is undecidable, one must show that there cannot be a decision procedure:
  - Diagonalization: leads assumption that there is such a procedure to a contradiction (e.g.: Halting problem for TMs).
  - Reduction: show that a problem known to be undecidable can be reduced to our problem.



A decidable ⟵ B decidable

A undecidable ⟶ B undecidable

Complexity

of a problem

Complexity class: collects problems that can be solved within a certain resource bound

- P: problems solvable in polynomial time by a deterministic machine
- NP: problems solvable in polynomial time by a nondeterministic machine
- PSpace: problems solvable with polynomial space by a deterministic machine
- NPSpace: problems solvable with polynomial space by a nondeterministic machine
- ExpTime: problems solvable in exponential time by a deterministic machine
- NExpTime: problems solvable in exponential time by a nondeterministic machine

$$\text{P} \subseteq \text{NP} \subseteq \text{PSpace} = \text{NPSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime}$$

Savitch's theorem

Strictness of the inclusions: $\text{P} \subset \text{ExpTime}$

open problem for the others!

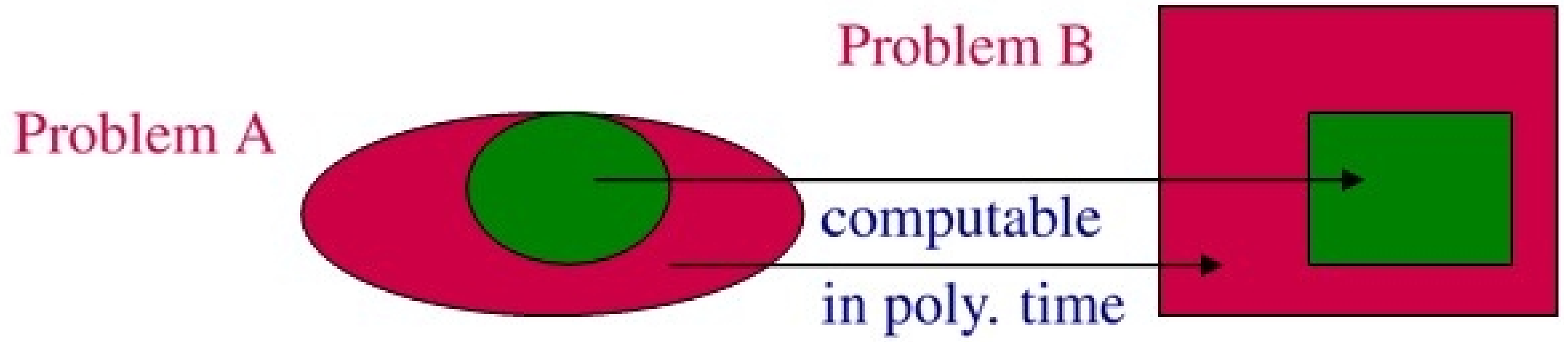
Dresden

Complexity

of a problem

A problem is complete for a complexity class K if it is in K and hard for K:

- in K: show that there is a decision procedure that works within the resource bound defining K
- hard for K: all problems in K can be reduced in polynomial time to this problem
  - direct proof: show that any TM that runs within the resource bound can be polynomially simulated by a problem instance
  - proof by reduction:



A in K ⟵ B in K

A K-hard ⟶ B K-hard

Complexity

of reasoning in $\mathcal{ALC}$

- The satisfiability, subsumption, instance and consistency problem in $\mathcal{ALC}$ (without TBox) are PSpace-complete.

  The same is true w.r.t. acyclic TBoxes.

  In PSpace: modification of the tableau-based algorithm 1.

  PSpace-hard: reduction of QBF (quantified Boolean formulae)

- W.r.t. general TBoxes, all these problems are ExpTime-complete.

  In ExpTime: automata-based algorithm

  ExpTime-hard: simulation of polynomial space alternating TMs

- There are "simple" extensions of $\mathcal{ALC}$ for which satisfiability (and thus all other problems) are undecidable.

  $\mathcal{ALC}$ with general TBoxes and feature agreements

  reduction of the domino problem 2.

Satisfiability problem

in $\mathcal{ALC}$ without TBoxes

Tableau-based decision procedure

- start with ABox of form $\mathcal{A}_0 = \{C_0(a_0)\}$;
- because of Savitch's theorem, we can ignore non-determinism, i.e., the $\sqcup$-rule chooses one successor ABox;
- thus only one complete ABox is generated;
- unfortunately, this complete ABox may be exponential in the size of C_0:

$$
\begin{array}{rcl}
C_1 & := & \exists r.A \sqcap \exists r.B \\
C_{i+1} & := & \exists r.A \sqcap \exists r.B \sqcap \forall r.C_i
\end{array}
$$

size of C_n is linear in n

The tableau-based decision procedure generates a tree-shaped model with 2^n leafs.

PSpace algorithm

for satisfiability in $\mathcal{ALC}$ without TBoxes

The complete ABox (tree-shaped model) generated by the tableau-based decision procedure may be exponential, but

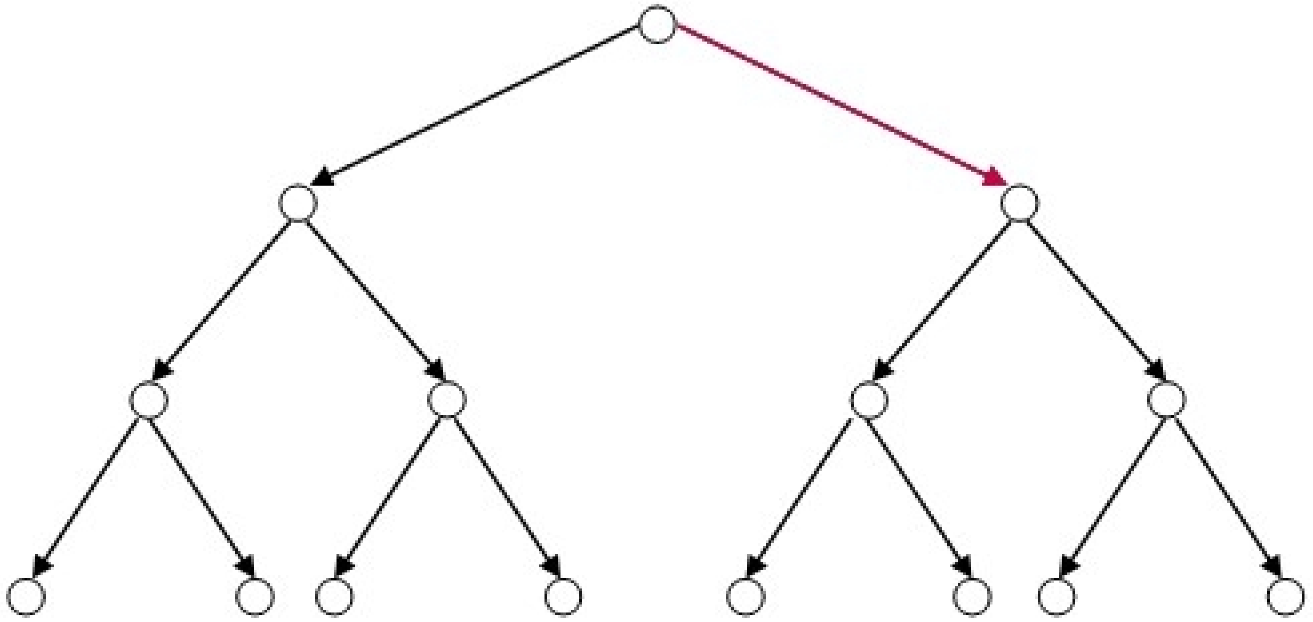
- the branching factor is linear in the size of C_0;
- the length of each path in the tree is linear in the size of C_0;
- the size of each node label (concept assertions for this node) is polynomial in the size of C_0;
- rule application is local: concerns a node and one direct successor;
- obvious contradictions are local: concern the label of one node.

Idea:

generate/explore the tree in a depth-first manner
while keeping only one path in memory

PSpace algorithm

for satisfiability in $\mathcal{ALC}$ without TBoxes



Idea:

generate/explore the tree in a depth-first manner

while keeping only one path in memory

PSpace algorithm

formulation as a recursive procedure

The procedure sat takes as input a finite set $\mathcal{C}$ of $\mathcal{ALC}$-concept descriptions, and returns true iff their conjunction is satisfiable.

sat($\mathcal{C}$) = if $\{A, \neg A\} \subseteq \mathcal{C}$ for some $A \in N_C$
then return false

else if $C \sqcap D \in \mathcal{C}$
then sat($(\mathcal{C} \setminus \{C \sqcap D\}) \cup \{C, D\}$)

else if $C \sqcup D \in \mathcal{C}$
then sat($(\mathcal{C} \setminus \{C \sqcup D\}) \cup \{C\}$) or sat($(\mathcal{C} \setminus \{C \sqcup D\}) \cup \{D\}$)

else if for all $\exists r.C \in \mathcal{C}$
sat($\mathcal{C} \cup \{D \mid \forall r.D \in \mathcal{C}\}$)
then return true
else return false

PSpace algorithm

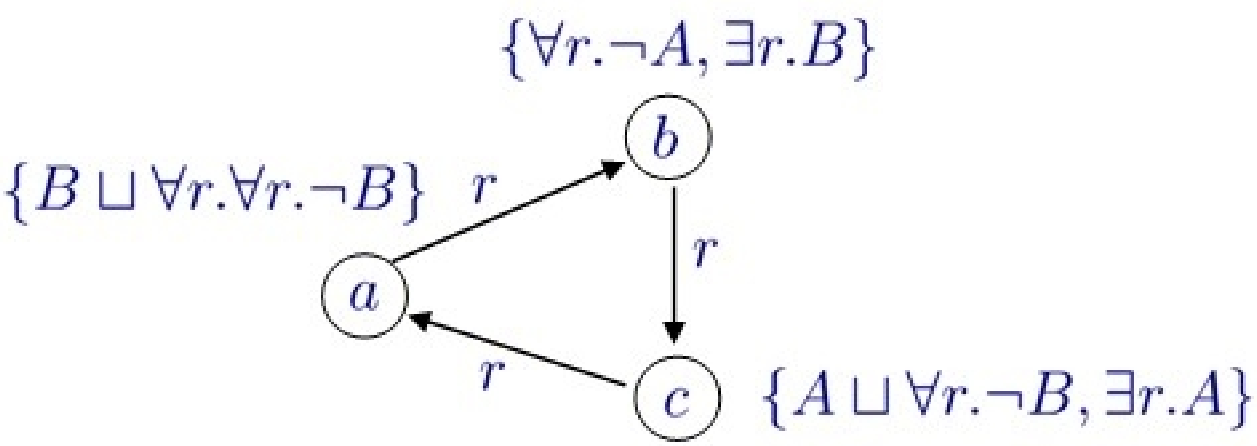
extension to ABox consistency

Precompletion: apply non-generating tableau-rules ($\sqcap$-rule, $\sqcup$-rule,$\forall$-rule) to the ABox until no such rule applies

- non-deterministic rule ($\sqcup$-rule) again harmless due to Savitch's theorem;
- size of each precompletion polynomial in the size of the input ABox;
- a precompleted ABox $\mathcal{A}$ is consistent iff the concepts

$$C_a := \bigsqcap_{C(a) \in \mathcal{A}} C$$

are (separately) satisfiable for all individual names a in $\mathcal{A}$.



PSpace algorithm

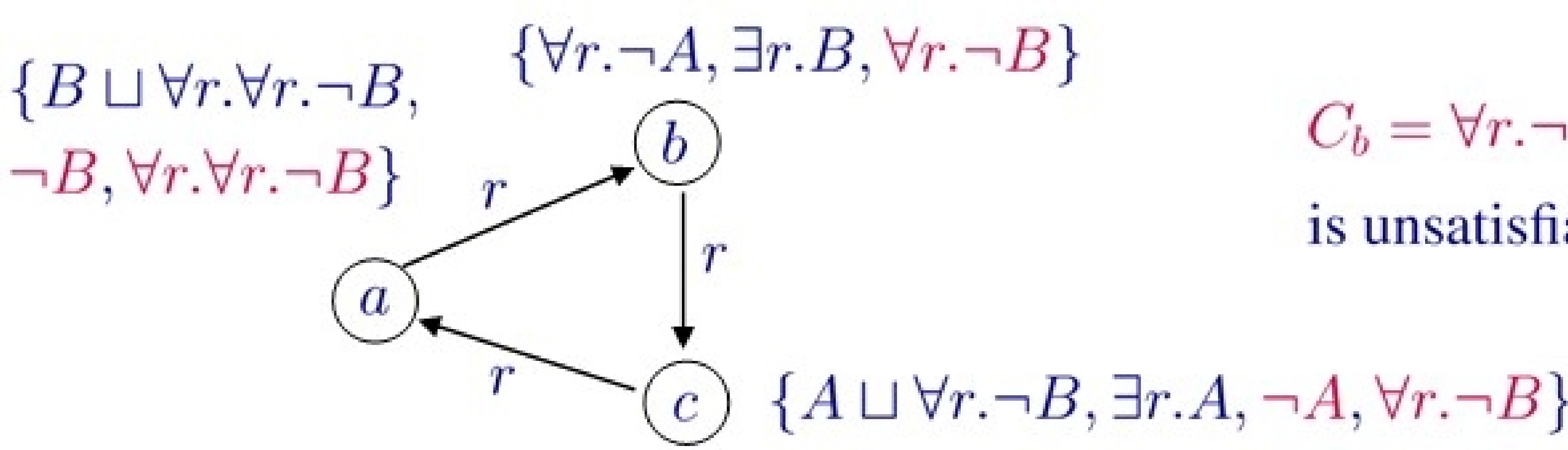
extension to ABox consistency

Precompletion: apply non-generating tableau-rules ($\sqcap$-rule, $\sqcup$-rule, $\forall$-rule) to the ABox until no such rule applies

- non-deterministic rule ($\sqcup$-rule) again harmless due to Savitch's theorem;
- size of each precompletion polynomial in the size of the input ABox;
- a precompleted ABox $\mathcal{A}$ is consistent iff the concepts

$$C_a := \bigsqcap_{C(a)\in\mathcal{A}} C$$

are (separately) satisfiable for all individual names a in $\mathcal{A}$.



$C_b = \forall r.\neg A \sqcap \exists r.B \sqcap \forall r.\neg B$

is unsatisfiable

Dresden

PSpace algorithm

extension to acyclic TBoxes

Problem: expansion of TBox may result in an exponential blow-up

Idea: expansion only "on demand"

The expansion-rule

Condition: $\mathcal{A}$ contains $A(a)$ for a definition $A \equiv C \in \mathcal{T}$, but not $C(a)$

Action: $\mathcal{A}' := \mathcal{A} \cup \{C(a)\}$

The approach for obtaining a PSpace algorithm described before also works in the presence of this rule.

PSpace hardness

by reduction to QBF

A quantified Boolean formula (QBF) is of the form

$$\psi = Q_1 p_1. \cdots Q_n p_n.\varphi$$

where $Q_i \in \{\exists, \forall\}$ and φ is a propositional formula over the variables $p_1, \ldots, p_n$.

Validity of ψ: well-known PSpace-complete problem

- if $n = 0$ then φ contains no variables:
  ψ valid iff φ evaluates to 1.

- if $n > 0$, then consider
  $$\begin{aligned} \psi_0 &:= Q_2 p_2. \cdots Q_n p_n.\varphi[p_1 \leftarrow 0] \quad \text{and} \\ \psi_1 &:= Q_2 p_2. \cdots Q_n p_n.\varphi[p_1 \leftarrow 1] \end{aligned}$$

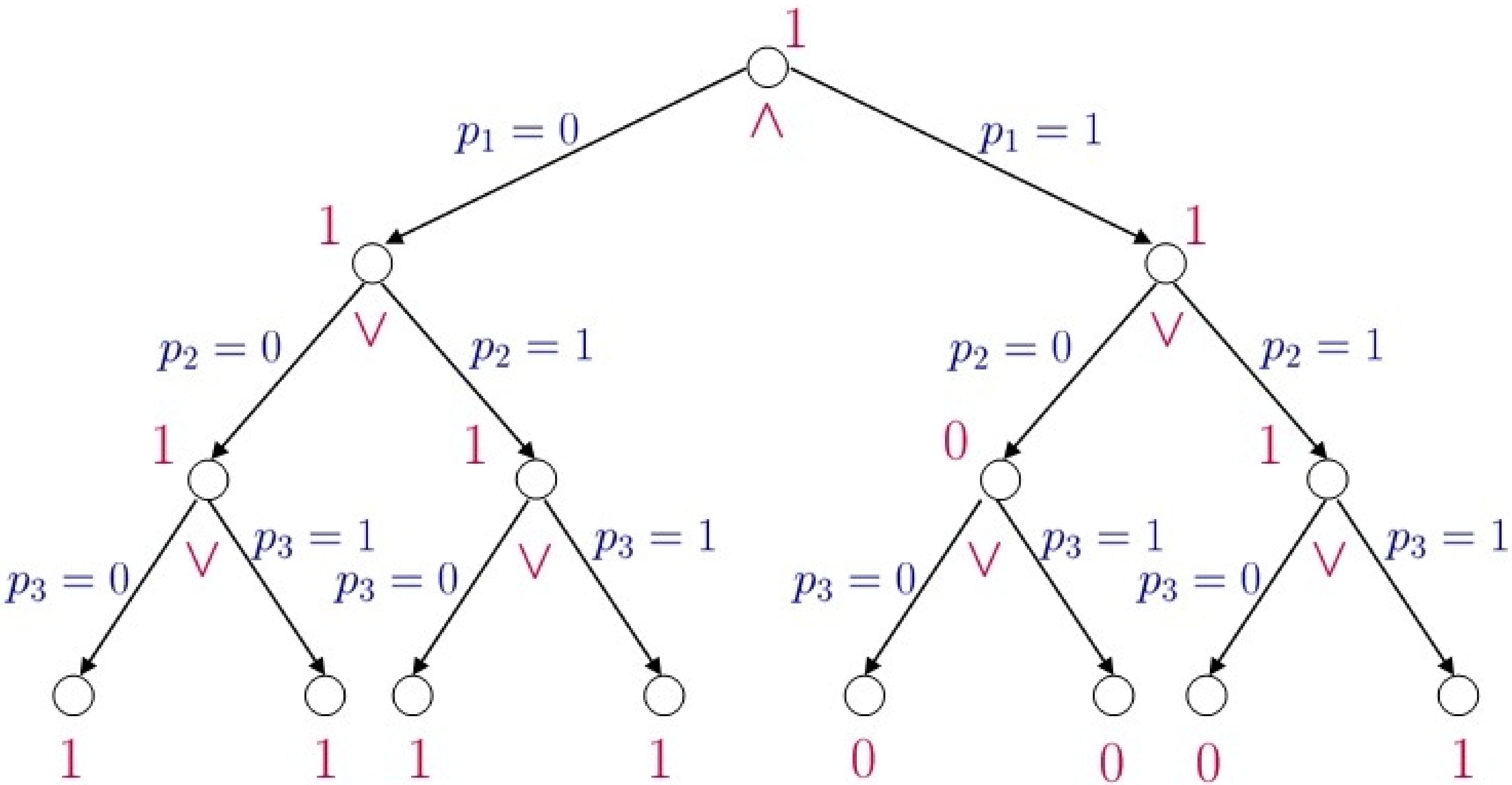
  if $Q_1 = \forall$ then ψ valid iff ψ_0 and ψ_1 valid

  if $Q_1 = \exists$ then ψ valid iff ψ_0 or ψ_1 valid

Validity of QBF

example

$\forall p_1.\exists p_2.\exists p_3.(p_1 \rightarrow (p_2 \wedge p_3))$ is valid



Dresden

Reduction of validity of QBF to satisfiability in $\mathcal{ALC}$

Idea: describe such an evaluation tree with an $\mathcal{ALC}$-concept

Role names:

r	yields the edges of the tree

Concept names:

$P_1, \ldots, P_n$	one for each propositional variable p_i
$T_0, \ldots, T_{n+1}$	T_i contains nodes at depth $\geq i$

Auxiliary concept descriptions:

$$\text{Depth} := \prod_{i=1}^{n+1} T_i \Rightarrow T_{i-1}$$

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

Dresden

Reduction

of validity of QBF to satisfiability in $\mathcal{ALC}$

Auxiliary concept descriptions:

Determined: from depth i on, the value of p_i is fixed

$$\bigsqcap_{i=1}^{n} (T_i \Rightarrow ((P_i \Rightarrow \forall r.P_i) \sqcap (\neg P_i \Rightarrow \forall r.\neg P_i)))$$

Branching: encodes the quantifier prefix $Q_1 p_1. \cdots Q_n p_n$

$$\bigsqcap_{\substack{0 \leq i \leq n-1 \\ Q_{i+1} = \forall}} (T_i \sqcap \neg T_{i+1}) \Rightarrow (\exists r.(T_{i+1} \sqcap \neg T_{i+2} \sqcap P_{i+1}) \sqcap \exists r.(T_{i+1} \sqcap \neg T_{i+2} \sqcap \neg P_{i+1}))$$

$$\sqcap$$

$$\bigsqcap_{\substack{0 \leq i \leq n-1 \\ Q_{i+1} = \exists}} (T_i \sqcap \neg T_{i+1}) \Rightarrow (\exists r.(T_{i+1} \sqcap \neg T_{i+2} \sqcap P_{i+1}) \sqcup \exists r.(T_{i+1} \sqcap \neg T_{i+2} \sqcap \neg P_{i+1}))$$

Reduction

of validity of QBF to satisfiability in $\mathcal{ALC}$

Auxiliary concept descriptions:

Encoding of φ: to obtain C_φ we

replace p_i by P_i, and the Boolean operations $\wedge, \vee, \neg$ by $\sqcap, \sqcup, \neg$

The reduction concept C_ψ:

$$T_0 \sqcap \neg T_1 \sqcap \bigsqcap_{i=0}^{n} \underbrace{\forall r. \cdots \forall r.}_{i \text{ times}} \quad (\text{Depth} \sqcap$$

Determined $\sqcap$

Branching $\sqcap$

$(T_n \Rightarrow C_\varphi))$

ψ valid iff C_ψ satisfiable

→ satisfiability PSpace-hard

subsumption, consistency, instance problem as well

Dresden

$\mathcal{ALCF}$

extension of $\mathcal{ALC}$ by feature agreements

Feature: symbol for a partial function (functional role)

$f^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{I}}$ partial function

Feature chain: composition of partial functions

$(f_1 \ldots f_n)^{\mathcal{I}} = f_1^{\mathcal{I}} \circ \cdots \circ f_n^{\mathcal{I}}$: $\quad (f_1 \ldots f_n)^{\mathcal{I}}(d) = f_n(\cdots f_2(f_1(d)) \cdots)$

Features can be used like roles in value and existential restrictions:

$\forall f.C$ and $\exists f.C$

Feature agreement: $u \doteq v$ where u, v are feature chains

with semantics:

$(u \doteq v)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid u^{\mathcal{I}}(d) = v^{\mathcal{I}}(d)$ and both are defined$\}$

$\mathcal{ALCF}$

extension of $\mathcal{ALC}$ by feature agreements

Example:

individuals having the same eyecolor as their mother:

$$\text{mother eyecolor} \doteq \text{eyecolor}$$

GCI:

if the eyecolor of father and mother agree, then the child also has this eyecolor

$$\text{father eyecolor} \doteq \text{mother eyecolor} \sqsubseteq \text{father eyecolor} \doteq \text{eyecolor}$$

Feature agreement: $u \doteq v$ where u, v are feature chains

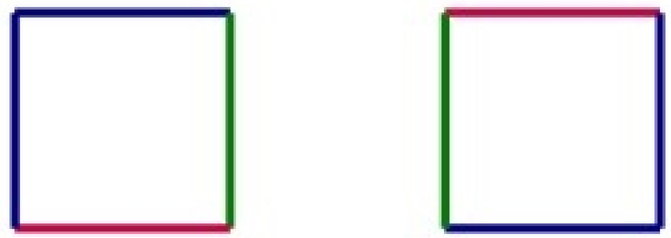
with semantics:

$$(u \doteq v)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid u^{\mathcal{I}}(d) = v^{\mathcal{I}}(d) \text{ and both are defined}\}$$

Domino problem

well-known undecidable problem

Domino types: squares with colored edges

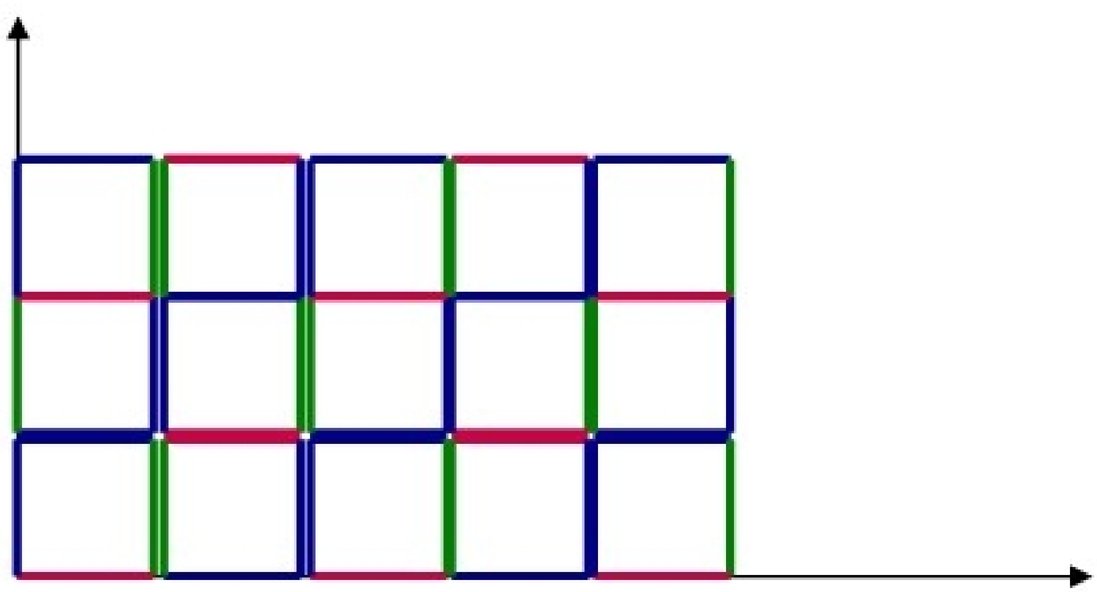


arbitrarily many for every type

may not be turned

Domino problem:

can we tile the quarter plane such that touching edges match



Domino problem

more formal definition

Domino system $\mathcal{D} = (D, H, V)$

D:	finite set of domino types
H:	horizontal compatibility relation $H \subseteq D \times D$
V:	vertical compatibility relation $V \subseteq D \times D$

Solution of $\mathcal{D}$

mapping $t : \mathbb{N} \times \mathbb{N} \rightarrow D$ such that

- $(t(x, y), t(x + 1, y)) \in H$
- $(t(x, y), t(x, y + 1)) \in V$

Domino problem

Given a domino system $\mathcal{D}$

Question does it have a solution

undecidable

Reduction

of the domino problem to satisfiability in $\mathcal{ALCF}$ with GCIs

Concept names:

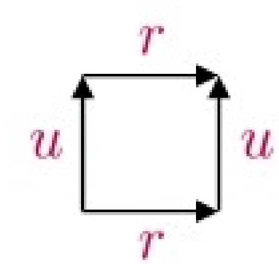
A_d for every $d \in D$ — $a \in A_d$ means: domino d is placed at this position

Role names:

r and u — r for "right" $\longrightarrow$ — u for "up" $\uparrow$

GCIs:

$\top \sqsubseteq r\,u \doteq u\,r$ — enforces the grid



$\top \sqsubseteq \bigsqcap_{d \neq d'} \neg(A_d \sqcap A_{d'})$ — every position has at most one domino

Dresden

Reduction

of the domino problem to satisfiability in $\mathcal{ALCF}$ with GCIs

GCIs:

$$\top \sqsubseteq \bigsqcap_{d \in D} (A_d \Rightarrow \bigsqcup_{(d,d') \in H} \exists r.A_{d'})$$ horizontal compatibility

$$\top \sqsubseteq \bigsqcap_{d \in D} (A_d \Rightarrow \bigsqcup_{(d,d') \in V} \exists u.A_{d'})$$ vertical compatibility

Concept description: $C_0 := \bigsqcup_{d \in D} A_d$

> C_0 is satisfiable w.r.t. the above GCIs
>
> iff
>
> the domino problem has a solution

satisfiability w.r.t. GCIs in $\mathcal{ALCF}$ undecidable

Dresden

$\mathcal{ALC}$ versus $\mathcal{ALCF}$

complexity of satisfiability and subsumption

	$\mathcal{ALC}$	$\mathcal{ALCF}$
no TBox	PSpace-complete	PSpace-complete
acyclic TBox	PSpace-complete	NExpTime-complete
general TBox	ExpTime-complete	undecidable