From Tableaux to Automata for Description Logics*

Franz Baader

Theoretical Computer Science

TU Dresden

Germany

- Short introduction to description logics.
- Tableau- and automata-based decision procedures for the DL $\mathcal{ALC}$ with general concept inclusions.
- Abstract framework of tableau systems and translation into looping automata.

* Joint work with Jan Hladik, Carsten Lutz, and Frank Wolter

Description Logics

class of knowledge representation formalisms

- Descended from structured inheritance networks [Brachman 78].
- Tried to overcome ambiguities in semantic networks and frames due to their lack of formal semantics.
- Restriction to a small set of "epistemologically adequate" operators for defining concepts (classes).
- Importance of well-defined basic inference procedures: subsumption and instance problem.
- First realization: system KL-ONE [Brachman&Schmolze];
  many successor systems (Classic, Crack, DLP, FaCT, Kris, K-Rep, Loom, Racer, . . .).
- First application: natural language processing;
  now also other domains (configuration, medical terminology, databases, ontologies for the semantic web, . . .).

Description logic system

structure

description language

- constructors for building complex concepts out of atomic concepts and roles
- formal, logic-based semantics

knowledge base

TBox

defines the terminology of the application domain

ABox

states facts about a specific “world”

reasoning component

- derive implicitly respresented knowledge (e.g., subsumption)
- “practical” algorithms

Description language

Constructors of the DL $\mathcal{ALC}$:

$C \sqcap D, C \sqcup D, \neg C, \forall r.C, \exists r.C$

A man	$Human \sqcap \neg Female \sqcap$
that has a rich or beautiful wife	$\exists married\_to.(Rich \sqcup Beautiful) \sqcap$
and only happy children	$\forall child.Happy$

TBox

definition of concepts

$Happy\_man \equiv Human \sqcap \ldots$

more complex constraints

$\exists married\_to.Doctor \sqsubseteq Doctor$

ABox

properties of individuals

$Happy\_man(Franz)$

$married\_to(Franz, Inge)$

$child(Franz, Luisa)$

Formal semantics

An interpretation $\mathcal{I}$ consist of a domain $\Delta^\mathcal{I}$ and it associates

- concepts C with sets $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$,
- roles r with binary relations $r^\mathcal{I}$ on $\Delta^\mathcal{I}$, and
- individuals a with elements $a^\mathcal{I} \in \Delta^\mathcal{I}$.

The semantics of the constructors is defined through identities:

- $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$, $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$
- $(\exists r.C)^\mathcal{I} = \{d \mid \exists e.(d,e) \in r^\mathcal{I} \wedge e \in C^\mathcal{I}\}$,
- $(\forall r.C)^\mathcal{I} = \{d \mid \forall e.(d,e) \in r^\mathcal{I} \rightarrow e \in C^\mathcal{I}\}$.

The interpretation $\mathcal{I}$ is a model of the concept definition/inclusion axiom/assertion

$$\begin{array}{rcl} A \equiv C & \text{iff} & A^\mathcal{I} = C^\mathcal{I}, \\ C \sqsubseteq D & \text{iff} & C^\mathcal{I} \subseteq D^\mathcal{I}, \\ C(a) & \text{iff} & a^\mathcal{I} \in C^\mathcal{I}, \\ r(a,b) & \text{iff} & (a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}. \end{array}$$

Dresden

Reasoning

makes implicitly represented knowledge explicit, provided as service by the DL system, e.g.:

Subsumption: Is C a subconcept of D?

$C \sqsubseteq_{\mathcal{T}} D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of the TBox $\mathcal{T}$.

Satisfiability: Is the concept C non-contradictory?

C is satisfiable w.r.t. $\mathcal{T}$ iff $C^{\mathcal{I}} \neq \emptyset$ for some model $\mathcal{I}$ of $\mathcal{T}$.

Consistency: Is the ABox $\mathcal{A}$ non-contradictory?

$\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff it has a model that is also a model of $\mathcal{T}$.

Instantiation: Is e an instance of C?

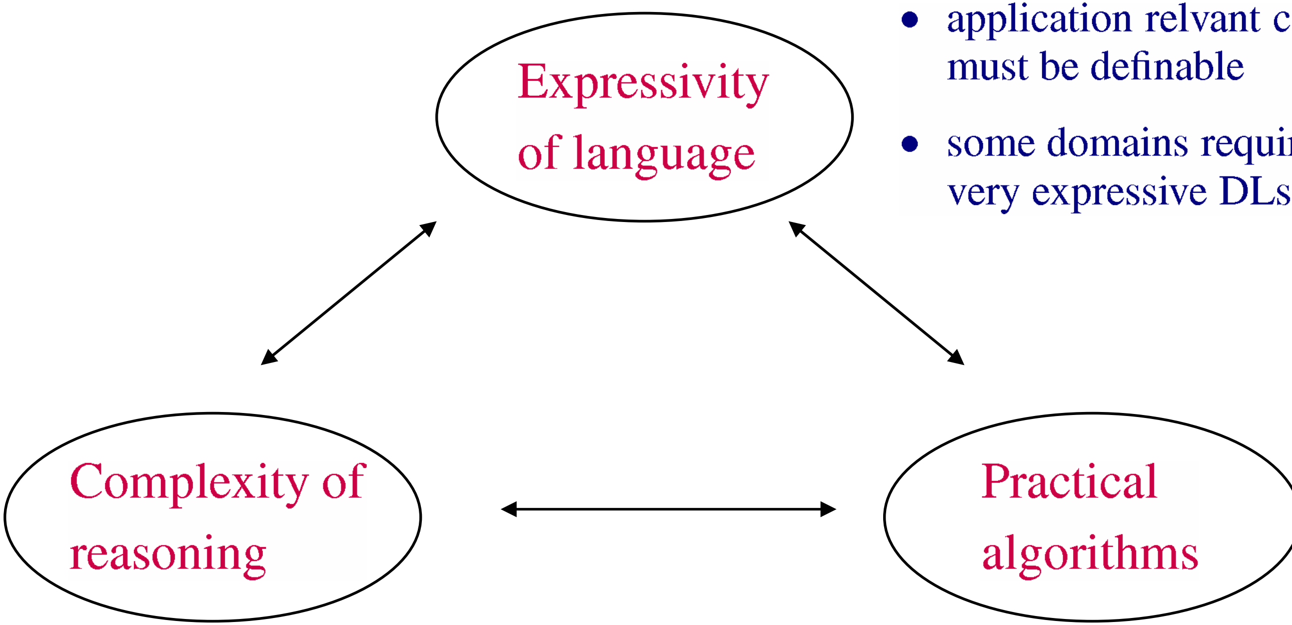
$\mathcal{A} \models_{\mathcal{T}} C(e)$ iff $e^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$.

polynomial reductions

in presence of negation

Focus of DL research

- application relvant concepts must be definable
- some domains require very expressive DLs



- reasoning must be decidable/of low complexity
- requires restricted description languages

- easy to implement and optimize
- behave well in practice

DL research

historic overview

Phase 1:

- implementation of incomplete systems (Back, Classic, Loom)
- based on structural subsumption algorithms

Phase 2:

- development of tableau-based algorithms and complexity results
- first implementation of tableau-based systems (Kris, Crack)
- first formal investigation of optimization methods

Phase 3:

- tableau-based algorithms for very expressive DLs
- highly optimized tableau-based systems (FaCT, Racer)
- relationship to modal logic and decidable fragments of FOL

worst-case optimal algorithms ⟷ *versus* ⟷ practical algorithms

PSpace-complete DLs such as $\mathcal{ALC}$ without general concept inclusions (GCIs) and the DLs implemented in Crack and Kris:

- Tableau-based algorithms are easy to implement and optimize.
- Can be realized within PSpace.

ExpTime-complete DLs such as $\mathcal{ALC}$ with general concept inclusions (GCIs) and the DLs implemented in FaCT and Racer:

- Tableau-based algorithms are still easy to implement and optimize.
- Usually yield NExpTime algorithms.
- Complexity upper-bound ExpTime shown using automata-based approach.
- No practical DL reasoner uses automata-based approach.

Goal

of this work

Avoid having to design two algorithms, one worst-case optimal and one practical, for each ExpTime-complete DL.

Achieved using the following approach:

- Define the abstract notion of tableau systems.
- Characterize the class of ExpTime-admissible tableau systems, which can be translated into looping automata on infinite trees.
- Exponential size of looping automata together with their polynomial time decidable emptiness problem yields ExpTime-upper bound.
- Recursive tableau systems yield tableau-based decision procedures.

Dresden

Tableau approach

for $\mathcal{ALC}$ without GCIs

- Tries to generate a finite, tree-shaped interpretation satisfying C_0 (where C_0 is a concept description in negation normal form).

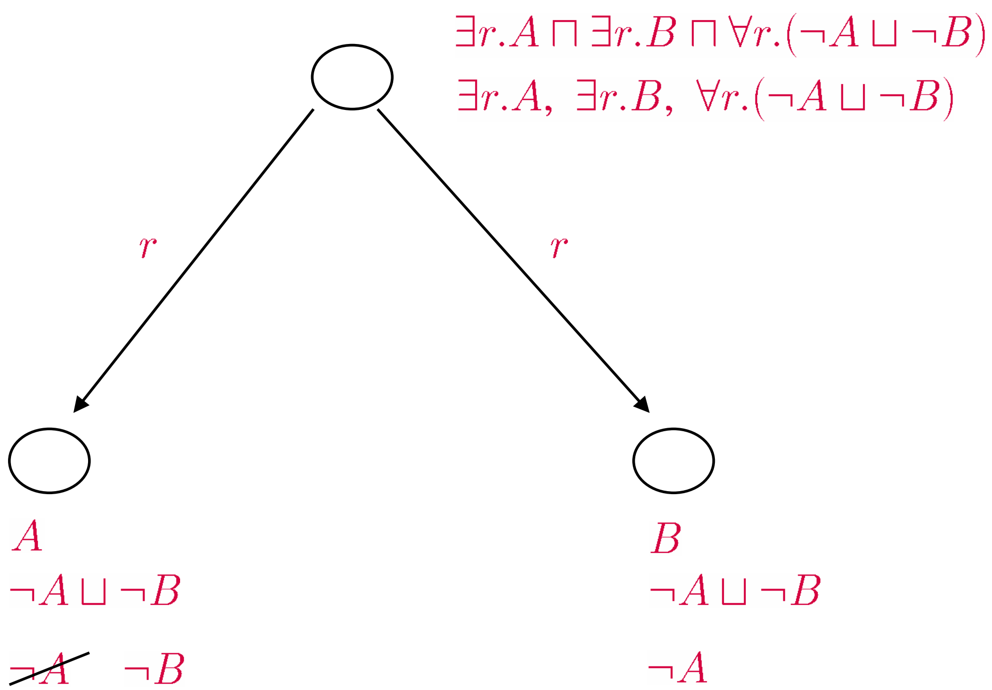
- Generates a root with label $\{C_0\}$, and

- then applies tableau rules:
  - propositional rules expand the label of the given node; rule for disjunction is non-deterministic.
  - existential rule generates new successor nodes;
  - universal rule extends the label of successor nodes.

- Clah trigger detects obvious contradictions in labels (both A and $\neg A$).

Tableau approach

Example:
satisfiability of $\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$



saturated, clash-free completion tree for the input

$$\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$$

Tableau approach

soundness, completeness, termination

Soundness

If there is a run of the algorithm that generates a saturated and clash-free completion tree, then the input concept is satisfiable.

Completeness

If the input concept is satisfiable, then there is a run of the algorithm that generates a saturated and clash-free completion tree.

Termination

Every run of the algorithm terminates with a saturated completion tree.

Dresden

Tableau approach

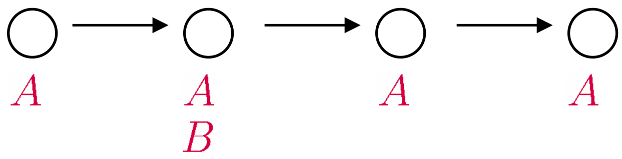
for $\mathcal{ALC}$ with GCIs

- For every GCI $C \sqsubseteq D$, the concept $nnf(\neg C \sqcup D)$ is added to every node of the completion tree.

- Blocking required to ensure termination:

$C_0 = A \sqcap \forall r.B$

$\mathcal{T} = \{A \sqsubseteq \exists r.A\}$



- Length of paths: may become exponential before blocking occurs.

- Non-determinism: treatment of disjunction.

NExpTime complexity

Automata approach

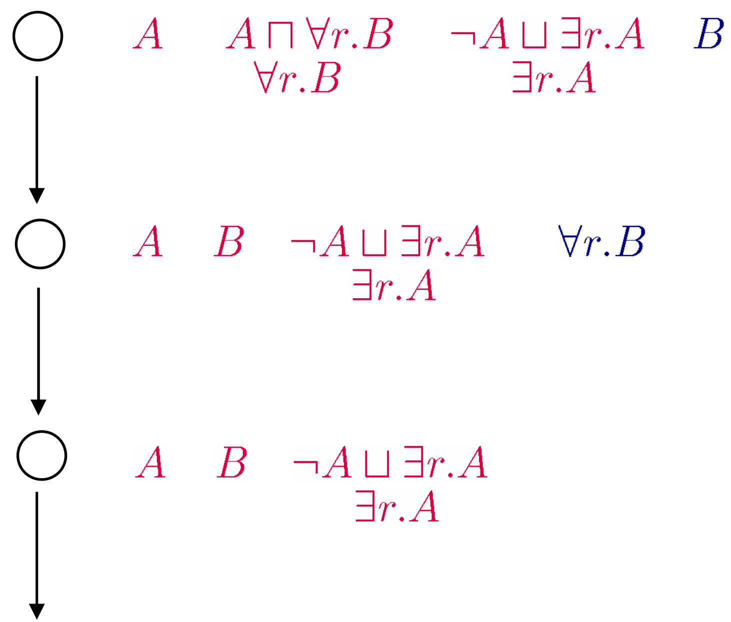
for $\mathcal{ALC}$ with GCIs

- Tests for the existence of a (possibly infinite) tree-shaped interpretation satisfying C_0 w.r.t. $\mathcal{T}$.
- States of the automaton: sets of "subformulae" of $\{C_0\}$ and $\mathcal{T}$ that
  - are propositionally expanded;
  - clash-free;
  - contain $nnf(\neg C \sqcup D)$ for all $C \sqsubseteq D$ in $\mathcal{T}$.
- Initial states: states containing C_0.
- Transitions look for the existence of "appropriate" successor nodes (existential and universal restrictions satisfied).
- Looping tree automaton: accepts if there is an infinite run.
- Non-deterministic automaton.

Automata approach

$C_0 = A \sqcap \forall r.B$

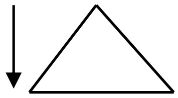
$\mathcal{T} = \{A \sqsubseteq \exists r.A\}$



Dresden

Automata approach

emptiness test: naive top-down approach



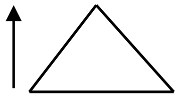
- Tries to construct a (possibly infinite) tree and a run on this tree.
- Starts with an initial state at the root, and then generates successor nodes according to the transition function.
- Looks for state repetition on paths to ensure termination.
- Very similar to tableau-approach with blocking.
- Complexity: NP in size of automaton if the automaton is nondeterministic.

Since the constructed automaton is exponential in the size of the input, this leaves us with a NExpTime procedure.

Dresden

Automata approach

emptiness test:
improved bottom-up approach



- Computes inactive states, i.e., states that cannot occur on an infinite run of the automaton:
  - Starts with obviously inactive states, i.e., states that do not have successor states w.r.t. the transition function.
  - Propagates inactiveness along the transition function.
- Tree language empty iff all initial states are inactive.
- Naive implementation already polynomial.
- Using appropriate data structures, the set of inactive states can be computed in linear time.

Since the constructed automaton is exponential in the size of the input, this provides us with an ExpTime procedure.

Dresden

Comparison

automata versus tableau approach

tableau approach	automata approach
• Constructs tree-shaped interpretation.	• Tests for existence of tree-shaped interpretation.
• Top-down	• Bottom-up
• NExpTime	• ExpTime
• Constructs sets of subformulae on-the-fly.	• First constructs (exponentially large) automaton, then applies emptiness test.

Dresden

Goal

of this work

Avoid having to design two algorithms, one worst-case optimal and one practical, for each ExpTime-complete DL.

Achieved using the following approach:

- Define the abstract notion of tableau systems.
- Characterize the class of ExpTime-admissible tableau systems, which can be translated into looping automata on infinite trees.
- Exponential size of looping automata together with their polynomial time decidable emptiness problem yields ExpTime-upper bound.
- Recursive tableau systems yield tableau-based decision procedures.

Dresden

Tableau systems

abstract notion, generalizes concrete tableau-based algorithms

Tableau system for set of inputs $\mathfrak{I}$: $(C_0, \mathcal{T})$

$$S = (\mathsf{NLE}, \mathsf{EL}, \cdot^S, \mathcal{R}, \mathcal{C}) \qquad S_{\mathcal{ALC}}$$

- NLE: node label elements. Node labels in completion trees are sets of node label elements. — all $\mathcal{ALC}$-concept descriptions
- EL: edge labels. Edges in completion trees are labeled with edge labels. — all $\mathcal{ALC}$-role names
- $\cdot^S$: input Γ mapped to $\Gamma^S = (\mathsf{nle}, \mathsf{el}, \mathsf{ini})$.
  - nle $\subseteq$ NLE and el $\subseteq$ EL finite. — "subconcepts" of input; roles occurring in input
  - ini $\subseteq 2^{\mathsf{nle}}$ (set of initial node labels). — sets containing C_0

Dresden

Tableau systems

(continued) $S = (\mathsf{NLE}, \mathsf{EL}, \cdot^S, \mathcal{R}, \mathcal{C})$

- $\mathcal{R}$: collection of tableau rules.

$$P \rightarrow_{\mathcal{R}} \{P_1, \ldots, P_k\}$$

Patterns, i.e., trees of depth ≤ 1 with node labels from 2^{NLE} and edge labels from EL.

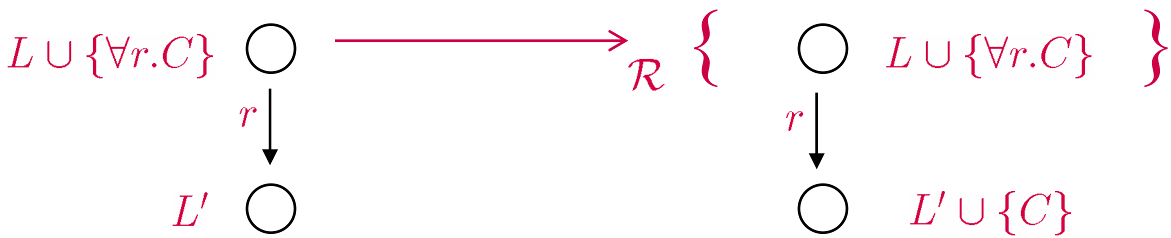
- Some rules of $S_{\mathcal{ALC}}$:

$$\underset{L \cup \{C \sqcup D\}}{\bigcirc} \longrightarrow_{\mathcal{R}} \left\{ \underset{\substack{L \cup \{C \sqcup D\} \\ \cup \{C\}}}{\bigcirc} \quad , \quad \underset{\substack{L \cup \{C \sqcup D\} \\ \cup \{D\}}}{\bigcirc} \right\}$$

Tableau systems

(continued) $S = (\mathsf{NLE}, \mathsf{EL}, \cdot^S, \mathcal{R}, \mathcal{C})$

- Some rules of $S_{\mathcal{ALC}}$ (continued):



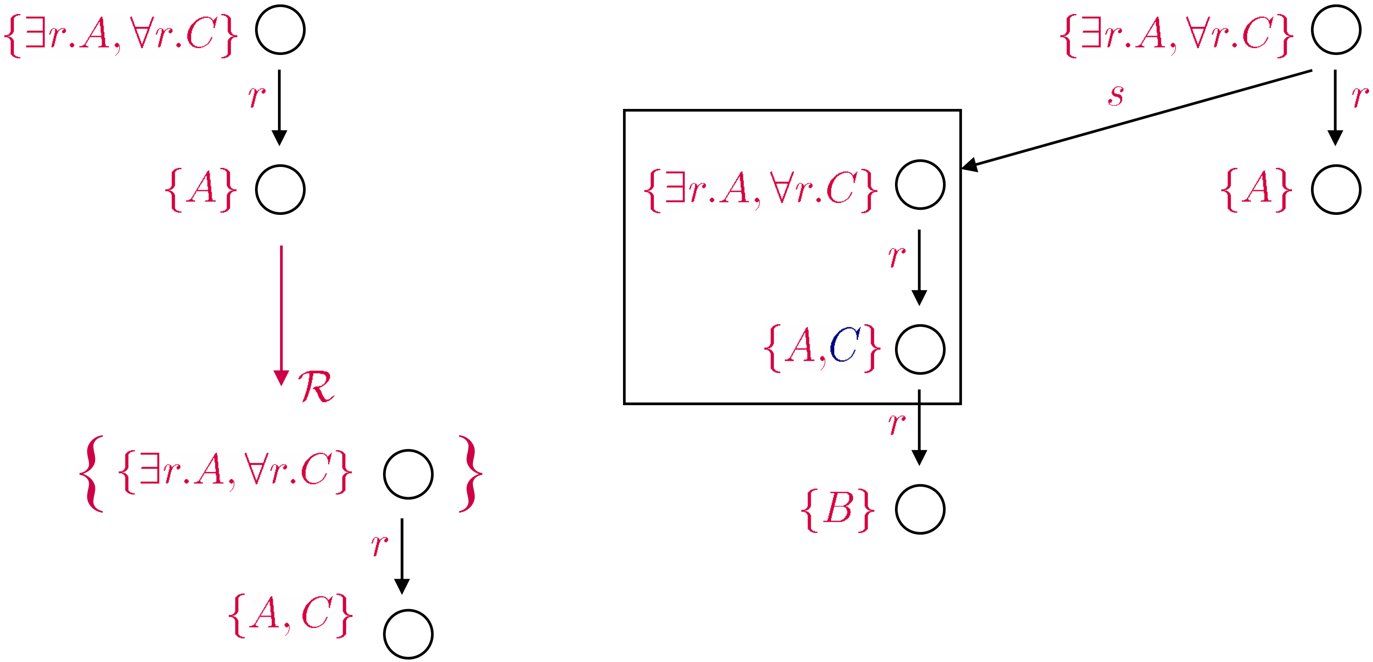
- $\mathcal{C}$: collection of clash triggers, i.e., set of patterns.

- Some clash triggers of $S_{\mathcal{ALC}}$:

Rule application

to S-tree, i.e., tree with the right labels.

- The rule $P \rightarrow_{\mathcal{ALC}} \{P_1, \ldots, P_k\}$ is applicable to a tree T iff pattern P matches a subtree of T.
- Rule application replaces P in T by one of the P_i (non-deterministic).



S-tree for input

$\Gamma \in \mathfrak{I}$ with $\Gamma^S = (\mathsf{nle}, \mathsf{el}, \mathsf{ini})$

Smallest set of trees such that

- All initial S-trees belong to this set: ○ $L \in \mathsf{ini}$
- Application of a rule to an element of this set yields an element of this set.
- Limit of an infinite chain of rule applications starting with an initial S-tree belongs to this set.

- Saturated S-tree: no rule applicable.
- Clash-free S-tree: no clash trigger applicable.

Dresden

Tableau system

Let S be a tableau system for the set of inputs $\mathfrak{I}$, and let $\mathcal{P}$ be a property of inputs, i.e., $\mathcal{P} \subseteq \mathfrak{I}$.

Soundness of S for $\mathcal{P}$

If there is a saturated and clash-free S-tree for Γ, then the input Γ satisfies $\mathcal{P}$.

Completeness of S for $\mathcal{P}$

If the input Γ satisfies $\mathcal{P}$, then there is a saturated and clash-free S-tree for Γ.

Translation to looping automata

Goal

Given a tableau system S that is sound and complete for property $\mathcal{P}$, construct for each input Γ a looping automaton $\mathcal{A}_\Gamma$ such that

$$L(\mathcal{A}_\Gamma) \neq \emptyset \text{ iff } \Gamma \in \mathcal{P}.$$

Two problems

1. S-trees for Γ are generated by rule application from initial S-trees. This is hard to check with automata.

2. Automata work on trees of a fixed arity.

Dresden

Solution to Problem 2

fixed arity

Modify definition of completeness

Let p be a polynomial.

The tableau system S is p-complete

iff

$\Gamma \in \mathcal{P}$ implies that there is a saturated and clash-free S-tree for Γ of outdegree bounded by $p(|\Gamma|)$.

In the following, we assume that S is sound and p-complete for $\mathcal{P}$.

Solution to Problem 1

requires additional conditions

Admissible tableau system:

- Rule application strictly extends the tree.
- If a rule is applicable to an S-tree T "contained" in a saturated S-tree $\widehat{T}$, then it can be applied such that the resulting tree is again contained in $\widehat{T}$.

$\{A \sqcup B\}$ contained in $\{A \sqcup B, A\}$

- Rule application to an S-tree "for an input" yields an S-tree for this input.
- Clash triggers are monotonic.

In the following, we assume that S is admissible.

Solution to Problem 1

main technical lemma

S-tree compatible with input Γ:

- Node labels and edge labels sanctioned by Γ^S.
- Root label contains an initial label for Γ.
- Outdegree bounded by $p(|\Gamma|)$.

Lemma

There is a saturated and clash-free S-tree for Γ

iff

there is a saturated and clash-free S-tree compatible with Γ.

Translation to looping automata

Automaton that accepts the saturated and clash-free S-trees compatible with Γ:

- Definition of states and of initial states ensures that the tree is compatible with Γ.
- Definition of transition function ensures that the tree is saturated and clash-free.

If the tableau system satisfies some additional restrictions (ExpTime-admissible), then the automaton can be constructed in exponential time.

Dresden

Main theorem

Let $\mathfrak{I}$ be a set of inputs, $\mathcal{P} \subseteq \mathfrak{I}$ a property, and p a polynomial. If there exists an ExpTime-admissible tableau system S for $\mathfrak{I}$ that is sound and p-complete for $\mathcal{P}$, then $\mathcal{P}$ is decidable in ExpTime.

Tableau-based decision procedures

from tableau systems

Let $\mathfrak{I}$ be a set of inputs, $\mathcal{P} \subseteq \mathfrak{I}$ a property, and f a recursive function. If there exists a recursive tableau system S for $\mathfrak{I}$ that is sound and f-complete for $\mathcal{P}$, then $\mathcal{P}$ is decidable with a tableau-based procedure.

Two problems must be solved in the proof:

1. Termination ensured by blocking.
2. Selection of applicable rule is don't care non-deterministic.

Related and future work

- From automata to tableaux:
  - The inverse tableau method [Voronkov, 2001] yields an on-the-fly realization of the automata-based decision procedure for $\mathcal{ALC}$ (with or w/o GCIs) [Baader&Tobies, IJCAR'01].
  - Translation of alternating two-way looping automata into a DL that has a (practical) tableau-based decision procedure [Hladik&Sattler, CADE'03].

- Extension of the abstract notion of tableau systems:
  - Larger patterns would facilitate treatment of DLs with number restrictions and inverse roles.
  - Global book keeping component would facilitate treatment of DLs with nominals.