An Introduction to Description Logic IX

Tableau-based algorithm

Marco Cerami

Palacký University in Olomouc
Department of Computer Science
Olomouc, Czech Republic

Olomouc, March 26th 2015

esf european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitiveness
INVESTMENTS IN EDUCATION DEVELOPMENT

Introduction

Introduction

- The first fast structural algorithms are **incomplete** for languages more expressive than basic ones.
- In particular, when the language contains negation and/or disjunction, there is the need to **manage the nondeterminism** arising from these constructors.
- In the 90's a new kind of algorithms is studied: the **tableau-based algorithms**.
- The subject of this presentation appeared in the 1991 paper **Attributive concept descriptions with complements**, by M. Schmidt-Schauß and G. Smolka.
- The target language is $\mathcal{ALC}$, that is a notational variant of multi-modal logic $\mathbf{K}_m$.

The language $\mathcal{ALC}$

Attributive Language with Complement

- The name $\mathcal{AL}$ stands for **Attributive Language** and the letter $\mathcal{C}$ stands for **Complement**.
- Language $\mathcal{ALC}$ began to be studied in the early 90's;
- Below we define the language $\mathcal{ALC}$:

C, D	$\longrightarrow$	A	atomic concept
		$\neg C$	complementation
		$C \sqcap D$	conjunction
		$C \sqcup D$	disjunction
		$\forall R.C$	value restriction
		$\exists R.C$	existential quantification

Examples

Some examples of $\mathcal{ALC}$ concepts:

$$\texttt{Person} \sqcap \forall \texttt{hasChild}.\neg\texttt{Male}$$

"person who has no sons"

$$\texttt{Person} \sqcap \neg\exists \texttt{hasChild}.\texttt{Male}$$

"person who has no sons"

$$\texttt{Person} \sqcap (\forall \texttt{hasChild}.\texttt{Male} \sqcup \forall \texttt{hasChild}.\texttt{Female})$$

"person who has either only sons or only daughters"

Interpretations

An **interpretation** is a pair

$$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$$

where:

- $\Delta^{\mathcal{I}}$ is a nonempty set, called **domain**;
- $\cdot^{\mathcal{I}}$ is an **interpretation function** that assigns:
  - to each individual name $a \in N_I$ an element
  $$a^{\mathcal{I}} \in \Delta^{\mathcal{I}},$$
  - to each atomic concept A a subset of the domain set
  $$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}},$$
  - to each role name R a binary relation on the domain set
  $$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}.$$

Semantics of complex concepts

The interpretation function $\cdot^{\mathcal{I}}$ can be **inductively extended** to complex concepts in the following way:

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\forall R.C)^{\mathcal{I}} = \{v \in \Delta^{\mathcal{I}} \mid \text{for every } w \in \Delta^{\mathcal{I}} \colon R^{\mathcal{I}}(v, w) \Rightarrow C^{\mathcal{I}}(w)\}$$

$$(\exists R.C)^{\mathcal{I}} = \{v \in \Delta^{\mathcal{I}} \mid \text{exists } w \in \Delta^{\mathcal{I}} \text{ s. t. } R^{\mathcal{I}}(v, w) \wedge C^{\mathcal{I}}(w)\}$$

Concept satisfiability

- A concept C is said to be **satisfiable** with respect to the knowledge base $\mathcal{K}$ when there exists an interpretation $\mathcal{I}$ satisfying $\mathcal{K}$, such that $C^{\mathcal{I}} \neq \emptyset$.
- We will consider concept satisfiability with respect to the **empty Knowledge Bases**, that is, there is no background knowledge to be managed.
- Since **every interpretation** $\mathcal{I}$ is a model of the empty KB, then a concept is satisfiable when there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.

Example I

The concept:

$$C_1 = \texttt{Female} \sqcap \forall \texttt{hasChild}.\texttt{Male}$$

is satisfiable, since in interpretation $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$, where:

- $\Delta^{\mathcal{I}_1} = \{v\}$,
- $\texttt{Male}^{\mathcal{I}_1} = \emptyset$,
- $\texttt{Female}^{\mathcal{I}_1} = \{v\}$,
- $\texttt{hasChild}^{\mathcal{I}_1} = \emptyset$.

it holds that

$$\begin{aligned} C_1^{\mathcal{I}_1} &= \texttt{Female}^{\mathcal{I}_1} \cap (\forall \texttt{hasChild}.\texttt{Male})^{\mathcal{I}_1} \\ &= \{v\} \cap \{x \in \Delta^{\mathcal{I}_1} \mid \text{f.e. } y \in \Delta^{\mathcal{I}_1} : \texttt{hasChild}^{\mathcal{I}_1}(x,y) \Rightarrow \texttt{Male}^{\mathcal{I}_1}(y)\} \\ &= \{v\} \cap \{v\} \\ &= \{v\} \\ &\neq \emptyset \end{aligned}$$

Example II

The concept:

$$C_2 = \texttt{Female} \sqcap \exists \texttt{hasChild}.\texttt{Male}$$

is satisfiable, since in interpretation $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2})$, where:

- $\Delta^{\mathcal{I}_2} = \{v, w\}$,
- $\texttt{Male}^{\mathcal{I}_2} = \{w\}$,
- $\texttt{Female}^{\mathcal{I}_2} = \{v\}$,
- $\texttt{hasChild}^{\mathcal{I}_2} = \{\langle v, w \rangle\}$.

it holds that

$$\begin{aligned} C_2^{\mathcal{I}_2} &= \texttt{Female}^{\mathcal{I}_2} \cap (\exists \texttt{hasChild}.\texttt{Male})^{\mathcal{I}_2} \\ &= \{v\} \cap \{x \in \Delta^{\mathcal{I}_2} \mid \text{exist } y \in \Delta^{\mathcal{I}_2}, \text{s.t. } \texttt{hasChild}^{\mathcal{I}_2}(x, y) \wedge \texttt{Male}^{\mathcal{I}_2}(y)\} \\ &= \{v\} \cap \{v\} \\ &= \{v\} \\ &\neq \emptyset \end{aligned}$$

Remarks

- As we have seen from the previous examples, the **quantifiers do not behave in the same way**.
- The **universal quantifier** is satisfied in an element of interpretation $\mathcal{I}_1$ **without successors** in the relation $\texttt{hasChild}^{\mathcal{I}_1}$.
- This is due to the fact that the semantics of universal quantifiers is an **implication** between two conditions, so, if the antecedent does not hold, the implication holds.
- The **existential quantifier** is satisfied in an element of interpretation $\mathcal{I}_1$ **with successors** in the relation $\texttt{hasChild}^{\mathcal{I}_2}$.
- This is due to the fact that the semantics of existential quantifiers is a **conjunction** between two conditions, so, if one of the conjuncts does not hold, the conjunction does not hold.

- Indeed, concept C_2 **would not be satisfied** by interpretation $\mathcal{I}_1$:

$$
\begin{aligned}
C_2^{\mathcal{I}_1} &= \texttt{Female}^{\mathcal{I}_1} \cap (\exists \texttt{hasChild}.\texttt{Male})^{\mathcal{I}_1} \\
&= \{v\} \cap \{x \in \Delta^{\mathcal{I}_1} \mid \text{exist } y \in \Delta^{\mathcal{I}_1}, \text{s.t.} \\
&\quad \texttt{hasChild}^{\mathcal{I}_1}(x, y) \wedge \texttt{Male}^{\mathcal{I}_1}(y)\} \\
&= \{v\} \cap \varnothing \\
&= \varnothing
\end{aligned}
$$

- Clearly, a suitable procedure for deciding concept satisfiability should **take into account this behavior of the quantifiers**.

Tableau algorithm for $\mathcal{ALC}$

General remarks

- The general idea of tableau-based algorithms is to build up a **tree model** that possibly satisfies a given concept C_0, where:
  - **nodes** represent the elements of the interpretation domain,
  - **edges** represent role-relations between domain elements.
- During the construction, concept C_0 is **syntactically broken down** into simpler concepts.
- It **stops** either when a **clash** is found or when the tree constructions **can not go further**.
- The starting point is to obtain a **negation normal form** of concept C_0.

Negation normal form

- $\mathcal{ALC}$ concepts can be linearly translated into **negation normal form**, in the following way:

$$\neg(C \sqcap D) \quad \rightsquigarrow \quad \neg C \sqcup \neg D$$

$$\neg(C \sqcup D) \quad \rightsquigarrow \quad \neg C \sqcap \neg D$$

$$\neg \forall R.C \quad \rightsquigarrow \quad \exists R.\neg C$$

$$\neg \exists R.C \quad \rightsquigarrow \quad \forall R.\neg C$$

- That is, concepts with the negation appearing only in front of atomic concepts.

Tableau

- A **tableau** is a tuple:

$$\mathbf{T} = \langle \mathcal{S}, \mathcal{L}, \mathcal{E} \rangle$$

where:

  - $\mathcal{S}$ is a set of elements,
  - $\mathcal{L}: \mathcal{S} \longrightarrow 2^{N_C}$ is a function that assigns a set of concept names to the elements of $\mathcal{S}$,
  - $\mathcal{E}: \mathcal{S} \times \mathcal{S} \longrightarrow 2^{N_R}$ is a function that assigns a set of role names to pair of elements of $\mathcal{S}$.

- The tableau construction is **initialized** by setting:
  - $\mathcal{S} = \{v_0\}$,
  - $\mathcal{L}(v_0) = C_0$,
  - $\mathcal{E}(\langle v_0, v_0 \rangle) = \varnothing$.

Completion rules

- The construction goes on by **breaking down** a given concept C_0 into **simpler subconcepts**.
- While breaking down C_0, **builds up** a tree-model.
- Each node of the tree is built in such a way that should **satisfy some subconcept** of C_0.
- In what follows we will see in detail the **completion rules** used to break down concept C_0.

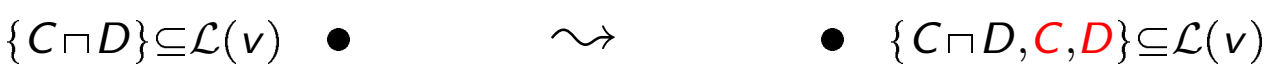
Rule for conjunction $\rightarrow_{\sqcap}$

If, for a given $v \in \mathcal{S}$, we have that

$$C \sqcap D \in \mathcal{L}(v),$$

then **add both** C **and** D to $\mathcal{L}(v)$:

$$\mathcal{L}(v) = \{C \sqcap D, \ldots\} \quad \rightsquigarrow \quad \mathcal{L}(v) = \{C \sqcap D, C, D, \ldots\}$$



Rule for disjunction $\rightarrow_{\sqcup}$

If, for a given $v \in \mathcal{S}$, we have that

$$C \sqcup D \in \mathcal{L}(v),$$

then **add either** C **or** D to $\mathcal{L}(v)$:

$$\mathcal{L}(v) = \{C \sqcup D, \ldots\} \quad \rightsquigarrow \quad \mathcal{L}(v) = \{C \sqcup D, C, \ldots\}$$

$$\{C \sqcup D\} \subseteq \mathcal{L}(v) \quad \bullet \quad \rightsquigarrow \quad \bullet \quad \{C \sqcup D, C\} \subseteq \mathcal{L}(v)$$

Rule for existential quantification $\rightarrow_{\exists}$

If, for a given $v \in \mathcal{S}$, we have that

$$\exists R.C \in \mathcal{L}(v),$$

then

1. **add a new element** w to $\mathcal{S}$:

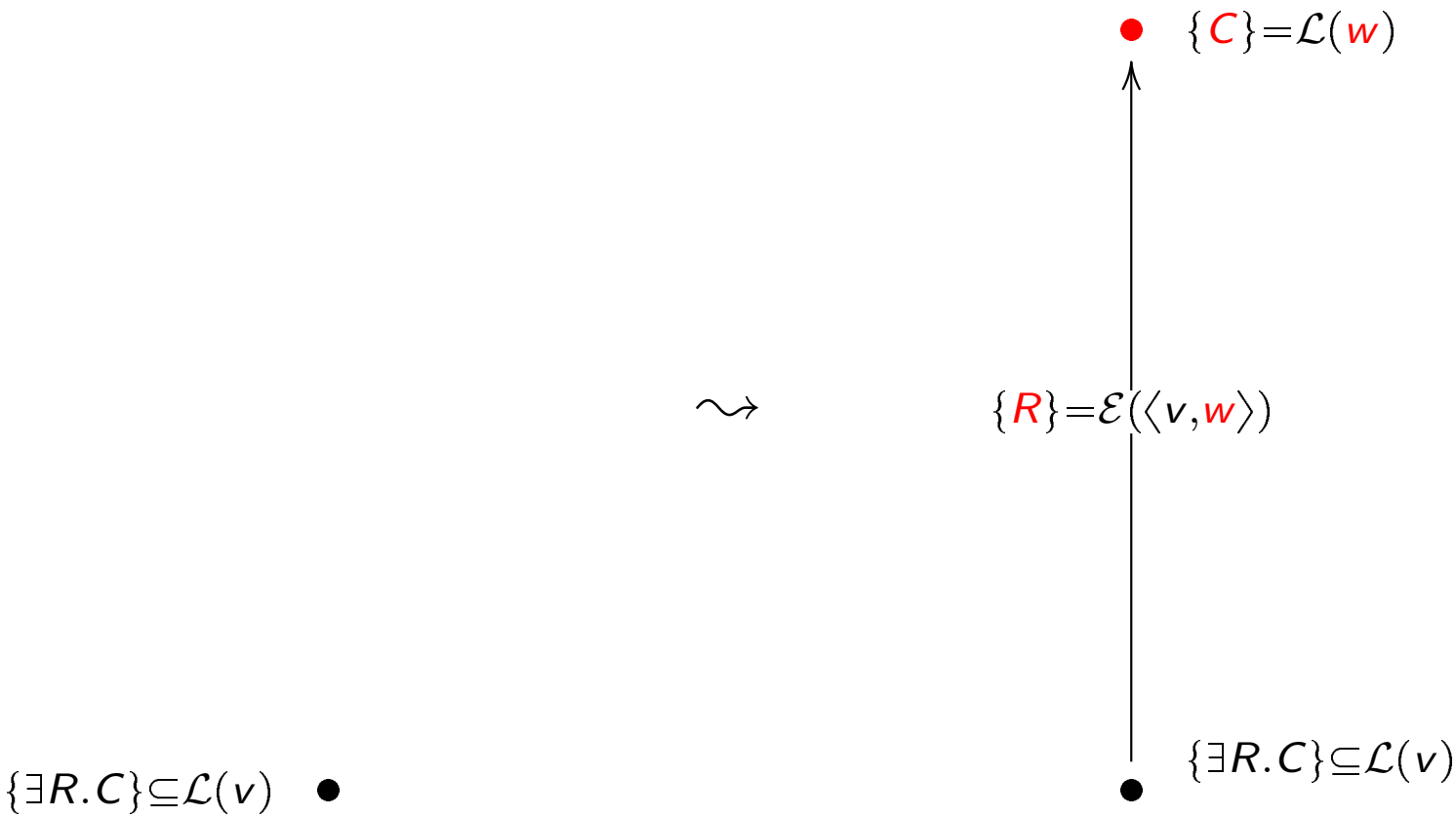
$$\mathcal{S} = \{v, \ldots\} \quad \rightsquigarrow \quad \mathcal{S} = \{v, w \ldots\},$$

2. **add role** R to $\mathcal{E}(\langle v, w \rangle)$:

$$\mathcal{E}(\langle v, w \rangle) = \varnothing \quad \rightsquigarrow \quad \mathcal{E}(\langle v, w \rangle) = \{R\},$$

3. **add concept** C to $\mathcal{L}(w)$:

$$\mathcal{L}(w) = \varnothing \quad \rightsquigarrow \quad \mathcal{L}(w) = \{C\}$$


$\{C\}=\mathcal{L}(w)$
$\{R\}=\mathcal{E}(\langle v,w\rangle)$
$\{\exists R.C\}\subseteq\mathcal{L}(v)$
$\{\exists R.C\}\subseteq\mathcal{L}(v)$

Rule for universal quantification $\rightarrow_\forall$

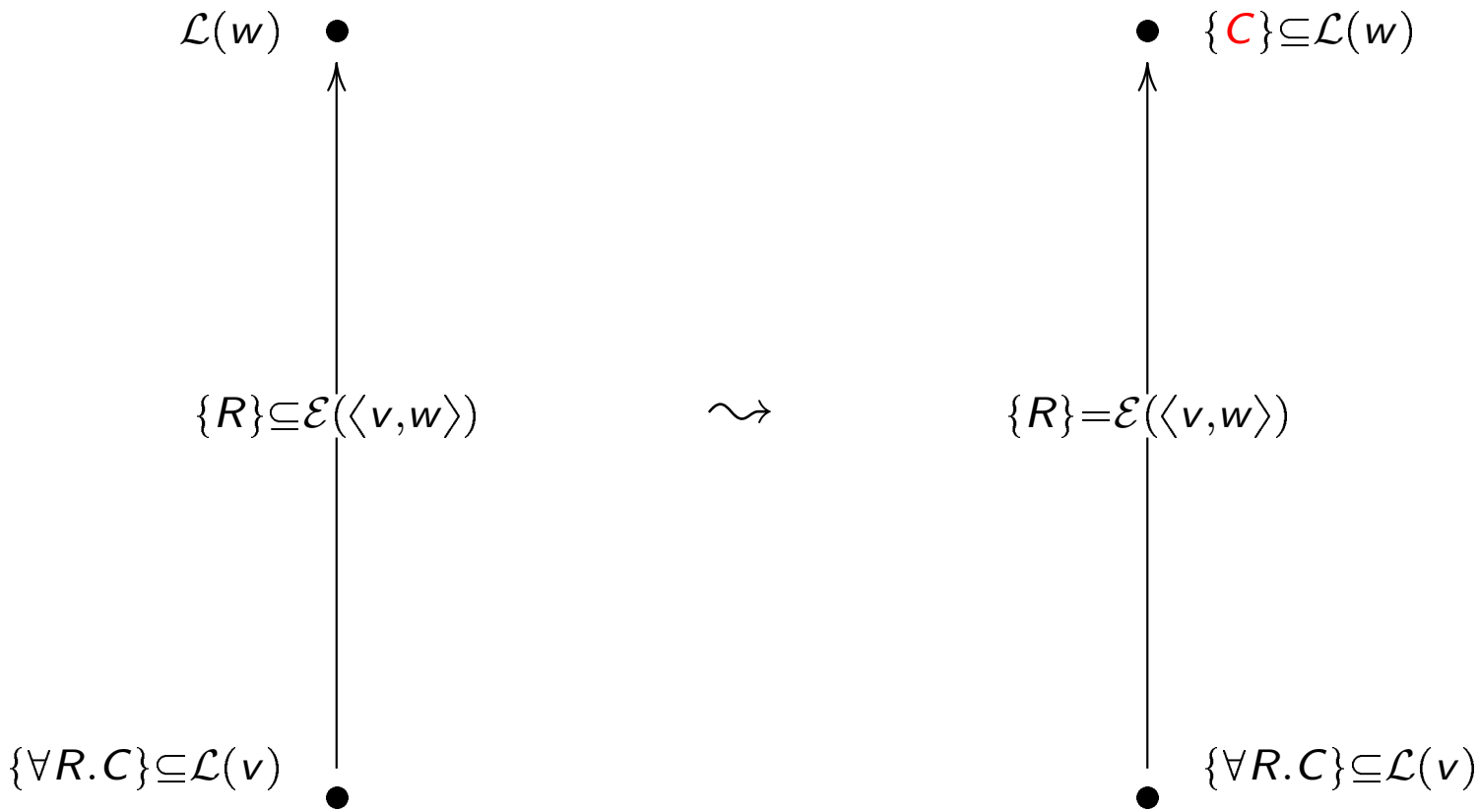
If, for a given $v \in \mathcal{S}$, we have that

1. $\forall R.C \in \mathcal{L}(v)$,
2. **already exists** $w \in \mathcal{S}$ such that

$$R \in \mathcal{E}(\langle v, w \rangle),$$

then **add concept** C to $\mathcal{L}(w)$:

$$\mathcal{L}(w) = \varnothing \qquad \rightsquigarrow \qquad \mathcal{L}(w) = \{C\}$$


$\mathcal{L}(w)$
$\{R\} \sqsubseteq \mathcal{E}(\langle v,w \rangle)$
$\{\forall R.C\} \sqsubseteq \mathcal{L}(v)$
$\rightsquigarrow$
$\{C\} \sqsubseteq \mathcal{L}(w)$
$\{R\} = \mathcal{E}(\langle v,w \rangle)$
$\{\forall R.C\} \sqsubseteq \mathcal{L}(v)$

Termination

- The construction of the tableau **T terminates** after a finite number of steps.
- The construction of the tableau is **stopped** in two cases:
  - either no rule is anymore applicable, in whose case we say that the tableau is **complete**,
  - or there is a node $v \in \mathcal{S}$ and a concept C such that

    $$\{C, \neg C\} \subseteq \mathcal{L}(v)$$

    in whose case we say that there is a **clash**.
- Through the completion trees, a **tree structure T** is built such that:
  - The **depth** of **T** is bounded by the nesting degree of C_0.
  - The **breadth** of **T** is linear in C_0.

Completeness

The proof of **completeness** is achieved in two steps:

the rule application yields a **complete completion tree** on concept C_0 $\iff$ there exists a **tableau T** for concept C_0 $\iff$ there exists an **interpretation** $\mathcal{I}$ which satisfies C_0

PSPACE implementation

An algorithm for constructing a tableau **T** for a concept C_0 **can be implemented** in such a way that it runs in PSPACE:

1. Since
   - the **size of the nodes** as sets of concepts, is **linear** on the size of C_0,
   - the **depth** of each branch is **linear** on the size of C_0,

   then **each branch is polynomial** on the size of C_0.

2. Since the **branches are independent** from each other and the **construction of a node** depends only on its predecessors, then:
   - a **depth first strategy** for constructing the tableau can be used,
   - The space used for constructing a branch can be **re-used**..