Semantic Web

2. Description logics (2/2)

Gerd Gröner, Matthias Thimm

{groener,thimm}@uni-koblenz.de

Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau

July 9, 2013

WeST People and Knowledge Networks

Recap

- Ontologies: definition and applications
- Structure of an ontology
- The description logic $\mathcal{ALC}$:
  - Syntax: signature, concepts, relations, axioms
  - Semantic: interpretations, models
  - Inference: entailment
  - $\mathcal{ALC}$ and first-order logics

Recap

- Ontologies: definition and applications
- Structure of an ontology
- The description logic $\mathcal{ALC}$:
  - Syntax: signature, concepts, relations, axioms
  - Semantic: interpretations, models
  - Inference: entailment
  - $\mathcal{ALC}$ and first-order logics

Home assignment:
Consider $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{A}_2)$ given via

$$\begin{aligned} \mathcal{T}_2 = \{ \quad & B \sqsubseteq D \quad \} \\ \mathcal{A}_2 = \{ \quad & d : B, (c, d) : S \quad \} \end{aligned}$$

Is the entailment $\mathcal{K}_2 \models c : \forall S.D$ valid? (Home assignment)

Overview

- Formal syntax and semantics provide the basis for understanding description logics

Overview

- Formal syntax and semantics provide the basis for understanding description logics
- Implementing the semantics of e. g. $\mathcal{ALC}$ is intractable for obtaining a proof procedure
  - Generate all interpretations
  - Check whether an interpretation satisfies a potential conclusions

WeST People and Knowledge Networks

Overview

- Formal syntax and semantics provide the basis for understanding description logics
- Implementing the semantics of e. g. $\mathcal{ALC}$ is intractable for obtaining a proof procedure
  - Generate all interpretations
  - Check whether an interpretation satisfies a potential conclusions
- Today we have a look at a very simple proof procedure for deciding consistency: the *Tableau Algorithm*

Overview

- Formal syntax and semantics provide the basis for understanding description logics
- Implementing the semantics of e. g. $\mathcal{ALC}$ is intractable for obtaining a proof procedure
  - Generate all interpretations
  - Check whether an interpretation satisfies a potential conclusions
- Today we have a look at a very simple proof procedure for deciding consistency: the *Tableau Algorithm*
- We also take another look at ontology languages in general and applications

Outline

WeST People and Knowledge Networks

Outline

Inference tasks 1/2

In description logics one usually distinguishes the following *inference tasks*:

- Subsumption problem: Given concepts C_1, C_2 does $\mathcal{K}$ entail $C_1 \sqsubseteq C_2$?
- Instance checking problem: Given concept C and individual t does $\mathcal{K}$ entail $t : C$?
- Relation checking problem: Given relation R and individuals t, t' does $\mathcal{K}$ entail $(t, t') : R$?
- Consistency problem: Is there $\mathcal{I}$ with $\mathcal{I} \models \mathcal{K}$?

WeST People and Knowledge Networks

Inference tasks 2/2

The subsumption and the consistency problem are closely related:

- C_2 subsumes C_1 if C_1 and $\neg C_2$ are inconsistent:

$$\mathcal{K} \models C_1 \sqsubseteq C_2 \quad \text{if and only if} \quad \neg\exists\mathcal{I} : \mathcal{I} \models \mathcal{K} \cup \{t : (C_1 \sqcap \neg C_2)\}$$

Inference tasks 2/2

The subsumption and the consistency problem are closely related:

- C_2 subsumes C_1 if C_1 and $\neg C_2$ are inconsistent:

$$\mathcal{K} \models C_1 \sqsubseteq C_2 \quad \text{if and only if} \quad \neg\exists\mathcal{I} : \mathcal{I} \models \mathcal{K} \cup \{t : (C_1 \sqcap \neg C_2)\}$$

$\rightarrow$ it suffices to investigate algorithms for checking consistency.

The Tableau Algorithm

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and consider the question

$$\exists \mathcal{I} : \mathcal{I} \models \mathcal{K} \ ?$$

The Tableau Algorithm

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and consider the question

$$\exists \mathcal{I} : \mathcal{I} \models \mathcal{K} \text{ ?}$$

The *tableau algorithm* tries to construct a model $\mathcal{I}$ of $\mathcal{K}$:

- If this is successful, $\mathcal{K}$ is consistent
- otherwise it is inconsistent

The Tableau Algorithm

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and consider the question

$$\exists \mathcal{I} : \mathcal{I} \models \mathcal{K} \; ?$$

The *tableau algorithm* tries to construct a model $\mathcal{I}$ of $\mathcal{K}$:

- If this is successful, $\mathcal{K}$ is consistent
- otherwise it is inconsistent

It works on a set $\mathcal{S}$ of ABoxes and iteratively expands on it:

- $\mathcal{S}$ is initialized with the singleton $\mathcal{S} = \{\mathcal{A}\}$

The Tableau Algorithm

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and consider the question

$$\exists \mathcal{I} : \mathcal{I} \models \mathcal{K} \text{ ?}$$

The *tableau algorithm* tries to construct a model $\mathcal{I}$ of $\mathcal{K}$:

- If this is successful, $\mathcal{K}$ is consistent
- otherwise it is inconsistent

It works on a set $\mathcal{S}$ of ABoxes and iteratively expands on it:

- $\mathcal{S}$ is initialized with the singleton $\mathcal{S} = \{\mathcal{A}\}$
- Apply different rules on the elements of $\mathcal{S}$ depending on the axioms in $\mathcal{T}$

The Tableau Algorithm

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and consider the question

$$\exists \mathcal{I} : \mathcal{I} \models \mathcal{K} \text{ ?}$$

The *tableau algorithm* tries to construct a model $\mathcal{I}$ of $\mathcal{K}$:

- If this is successful, $\mathcal{K}$ is consistent
- otherwise it is inconsistent

It works on a set $\mathcal{S}$ of ABoxes and iteratively expands on it:

- $\mathcal{S}$ is initialized with the singleton $\mathcal{S} = \{\mathcal{A}\}$
- Apply different rules on the elements of $\mathcal{S}$ depending on the axioms in $\mathcal{T}$
- If no more rules are applicable, $\mathcal{K}$ is consistent if there is an ABox $\mathcal{A}'$ in $\mathcal{S}$ that is consistent (contains no axioms $t : C$, $t : \neg C$)

The Tableau Algorithm - Negation Normal Form

In order to apply the tableau algorithm we have to assume that $\mathcal{K}$ is in *negation normal form*:

- $\neg(C_1 \sqcup C_2) \quad \longrightarrow \quad \neg C_1 \sqcap \neg C_2$
- $\neg(C_1 \sqcap C_2) \quad \longrightarrow \quad \neg C_1 \sqcup \neg C_2$
- $\neg\exists R.C \quad \longrightarrow \quad \forall R.(\neg C)$
- $\neg\forall R.C \quad \longrightarrow \quad \exists R.(\neg C)$

The Tableau Algorithm - Negation Normal Form

In order to apply the tableau algorithm we have to assume that $\mathcal{K}$ is in *negation normal form*:

- $\neg(C_1 \sqcup C_2) \quad \longrightarrow \quad \neg C_1 \sqcap \neg C_2$
- $\neg(C_1 \sqcap C_2) \quad \longrightarrow \quad \neg C_1 \sqcup \neg C_2$
- $\neg\exists R.C \quad \longrightarrow \quad \forall R.(\neg C)$
- $\neg\forall R.C \quad \longrightarrow \quad \exists R.(\neg C)$

From now on, we assume that every concept appearing in an axiom in $\mathcal{K}$ is in negation normal form.

The Tableau Algorithm - Negation Normal Form

In order to apply the tableau algorithm we have to assume that $\mathcal{K}$ is in *negation normal form*:

- $\neg(C_1 \sqcup C_2) \quad \longrightarrow \quad \neg C_1 \sqcap \neg C_2$
- $\neg(C_1 \sqcap C_2) \quad \longrightarrow \quad \neg C_1 \sqcup \neg C_2$
- $\neg\exists R.C \quad \longrightarrow \quad \forall R.(\neg C)$
- $\neg\forall R.C \quad \longrightarrow \quad \exists R.(\neg C)$

From now on, we assume that every concept appearing in an axiom in $\mathcal{K}$ is in negation normal form. For example

$$\neg(C_1 \sqcap C_2) \sqsubseteq \neg\exists R.\neg(C_3 \sqcup C_4) \quad \longrightarrow \quad \neg C_1 \sqcup \neg C_2 \sqsubseteq \forall R.(C_3 \sqcup C_4)$$

The Tableau Algorithm - Rules 1/2

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{A}' \in \mathcal{S}$.

The Tableau Algorithm - Rules 1/2

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{A}' \in \mathcal{S}$.

- $\sqcap$-rule: if $t : C_1 \sqcap C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \not\subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : C_1, t : C_2\}$ to $\mathcal{S}$.

The Tableau Algorithm - Rules 1/2

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{A}' \in \mathcal{S}$.

- $\sqcap$-rule: if $t : C_1 \sqcap C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \not\subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : C_1, t : C_2\}$ to $\mathcal{S}$.
- $\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.

The Tableau Algorithm - Rules 1/2

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{A}' \in \mathcal{S}$.

- $\sqcap$-rule: if $t : C_1 \sqcap C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \not\subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : C_1, t : C_2\}$ to $\mathcal{S}$.
- $\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.
- $\exists$-rule: if $t : \exists R.C \in \mathcal{A}'$ and there is no t' with $\{(t, t') : R, t' : C\} \subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$, *create* a new individual t'', and add $\mathcal{A}' \cup \{(t, t'') : R, t'' : C\}$ to $\mathcal{S}$.

WeST People and Knowledge Networks

The Tableau Algorithm - Rules 2/2

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{A}' \in \mathcal{S}$.

The Tableau Algorithm - Rules 2/2

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{A}' \in \mathcal{S}$.

- $\forall$-rule: if $\{t : \forall R.C, (t, t') : R\} \subseteq \mathcal{A}'$ and $\{t' : C\} \notin \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t' : C\}$ to $\mathcal{S}$.

Let $\mathcal{S}$ be a set of ABoxes (initialized with $\mathcal{S} = \{\mathcal{A}\}$).

Let $\mathcal{A}' \in \mathcal{S}$.

- $\forall$-rule: if $\{t : \forall R.C, (t, t') : R\} \subseteq \mathcal{A}'$ and $\{t' : C\} \notin \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t' : C\}$ to $\mathcal{S}$.
- $\sqsubseteq$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$ and $t : (\neg C_1 \sqcup C_2) \notin \mathcal{A}'$ for t appearing in $\mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : (\neg C_1 \sqcup C_2)\}$ to $\mathcal{S}$.

WeST People and Knowledge Networks

The Tableau Algorithm - Approach

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $\mathcal{A}$ be consistent.

The Tableau Algorithm - Approach

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $\mathcal{A}$ be consistent.

The tableau algorithm:

1. Set $\mathcal{S} = \{\mathcal{A}\}$
2. Is some rule applicable?
   - yes: goto 3
   - no: $\mathcal{K}$ is consistent; exit
3. Apply the rule to $\mathcal{S}$
4. Remove all $\mathcal{A}' \in \mathcal{S}$ with $t : C, t : \neg C \in \mathcal{A}'$ (for some t, C)
5. $\mathcal{S} = \emptyset$?
   - yes: $\mathcal{K}$ is inconsistent; exit
   - no: goto 2

The Tableau Algorithm - Example

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be given via

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$

WeST People and Knowledge Networks

The Tableau Algorithm - Example

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be given via

$$\begin{aligned} \mathcal{T} &= \{ \quad A \sqsubseteq \exists R.C \quad \} \\ \mathcal{A} &= \{ \quad a : A, b : D, (a, b) : R \quad \} \end{aligned}$$

Observe:

- $\mathcal{K}$ is in negation normal form
- $\mathcal{A}$ is consistent

WeST People and Knowledge Networks

The Tableau Algorithm - Example

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be given via

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$

Observe:

- $\mathcal{K}$ is in negation normal form
- $\mathcal{A}$ is consistent

Initialize $\mathcal{S} = \{\{a : A, b : D, (a, b) : R\}\}$.

People and Knowledge Networks

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$

$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$

$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R\}\}$$

$\sqsubseteq$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$ and $t : (\neg C_1 \sqcup C_2) \notin \mathcal{A}'$ for t appearing in $\mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : (\neg C_1 \sqcup C_2)\}$ to $\mathcal{S}$.

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R\}\}$$

$\sqsubseteq$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$ and $t : (\neg C_1 \sqcup C_2) \notin \mathcal{A}'$ for t appearing in $\mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : (\neg C_1 \sqcup C_2)\}$ to $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C\}\}$$

WeST People and Knowledge Networks

The Tableau Algorithm - Example cont'd

$$
\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq \exists R.C \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : D, (a, b) : R \quad \} \\
\mathcal{S} &= \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C\}\}
\end{aligned}
$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C\}\}$$

$\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a,b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C\}\}$$

$\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \neg A\},$$
$$\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C\}\}$$

$\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \neg A\}, \leftarrow$$
$$\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C\}\}$$

$\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \neg A\},$$
$$\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C\}\}$$

WeST People and Knowledge Networks

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a,b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C\}\}$$

$\exists$-rule: if $t : \exists R.C \in \mathcal{A}'$ and there is no t' with $\{(t,t') : R, t' : C\} \subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$, *create* a new individual t'', and add $\mathcal{A}' \cup \{(t,t'') : R, t'' : C\}$ to $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, \\ (a,t'') : R, t'' : C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$

$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$

$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C\}\}$$

$\sqsubseteq$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$ and $t : (\neg C_1 \sqcup C_2) \notin \mathcal{A}'$ for t appearing in $\mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : (\neg C_1 \sqcup C_2)\}$ to $\mathcal{S}$.

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C\}\}$$

$\sqsubseteq$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$ and $t : (\neg C_1 \sqcup C_2) \notin \mathcal{A}'$ for t appearing in $\mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : (\neg C_1 \sqcup C_2)\}$ to $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, b : \neg A \sqcup \exists R.C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$

$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$

$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\ b : \neg A \sqcup \exists R.C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, b : \neg A \sqcup \exists R.C\}\}$$

$\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a,b) : R \quad \}$$
$$\mathcal{S} = \{\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a,t'') : R, t'' : C, \\ b : \neg A \sqcup \exists R.C\}\}$$

$\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$ and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a,t'') : R, t'' : C, \\ b : \neg A \sqcup \exists R.C, b : \neg A\}, \\ \{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a,t'') : R, t'' : C, \\ b : \neg A \sqcup \exists R.C, b : \exists R.C\}\}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a, b) : R \quad \}$$
$$\begin{aligned}\mathcal{S} = \{&\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C,\\ &\quad b : \neg A \sqcup \exists R.C, b : \neg A\},\\ &\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C,\\ &\quad b : \neg A \sqcup \exists R.C, b : \exists R.C\}\}\end{aligned}$$

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : D, (a,b) : R \quad \}$$
$$\begin{aligned}\mathcal{S} = \{ & \{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\ & \quad b : \neg A \sqcup \exists R.C, b : \neg A\}, \\ & \{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\ & \quad b : \neg A \sqcup \exists R.C, b : \exists R.C\}\}\end{aligned}$$

$\exists$-rule: if $t : \exists R.C \in \mathcal{A}'$ and there is no t' with $\{(t, t') : R, t' : C\} \subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$, *create* a new individual t'', and add $\mathcal{A}' \cup \{(t, t'') : R, t'' : C\}$ to $\mathcal{S}$.

The Tableau Algorithm - Example cont'd

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.C \quad \}$$

$$\mathcal{A} = \{ \quad a : A, b : D, (a,b) : R \quad \}$$

$$\begin{aligned}\mathcal{S} = \{\{&a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a,t'') : R, t'' : C,\\ &b : \neg A \sqcup \exists R.C, b : \neg A\},\\ \{&a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a,t'') : R, t'' : C,\\ &b : \neg A \sqcup \exists R.C, b : \exists R.C\}\}\end{aligned}$$

$\exists$-rule: if $t : \exists R.C \in \mathcal{A}'$ and there is no t' with $\{(t,t') : R, t' : C\} \subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$, *create* a new individual t'', and add $\mathcal{A}' \cup \{(t,t'') : R, t'' : C\}$ to $\mathcal{S}$.

$$\begin{aligned}\mathcal{S} = \{\{&a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a,t'') : R, t'' : C,\\ &b : \neg A \sqcup \exists R.C, b : \neg A\},\\ \{&a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a,t'') : R, t'' : C,\\ &b : \neg A \sqcup \exists R.C, b : \exists R.C, (b,t''') : R, t''' : C\}\}\end{aligned}$$

The Tableau Algorithm - Example cont'd

$$
\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq \exists R.C \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : D, (a, b) : R \quad \} \\
\mathcal{S} &= \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\quad b : \neg A \sqcup \exists R.C, b : \neg A\}, \\
&\quad\ \{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\quad b : \neg A \sqcup \exists R.C, b : \exists R.C, (b, t''') : R, t''' : C\}\}
\end{aligned}
$$

The Tableau Algorithm - Example cont'd

$$
\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq \exists R.C \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : D, (a, b) : R \quad \} \\
\mathcal{S} &= \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\qquad b : \neg A \sqcup \exists R.C, b : \neg A\}, \\
&\qquad \{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\qquad b : \neg A \sqcup \exists R.C, b : \exists R.C, (b, t''') : R, t''' : C\}\}
\end{aligned}
$$

Observations:

- No more rules applicable

The Tableau Algorithm - Example cont'd

$$
\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq \exists R.C \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : D, (a,b) : R \quad \} \\
\mathcal{S} &= \{\{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\qquad b : \neg A \sqcup \exists R.C, b : \neg A\}, \\
&\qquad \{a : A, b : D, (a,b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\qquad b : \neg A \sqcup \exists R.C, b : \exists R.C, (b, t''') : R, t''' : C\}\}
\end{aligned}
$$

Observations:

- No more rules applicable
- All ABoxes in $\mathcal{S}$ are consistent

The Tableau Algorithm - Example cont'd

$$
\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq \exists R.C \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : D, (a, b) : R \quad \} \\
\mathcal{S} &= \{\{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\qquad b : \neg A \sqcup \exists R.C, b : \neg A\}, \\
&\qquad \{a : A, b : D, (a, b) : R, a : \neg A \sqcup \exists R.C, a : \exists R.C, (a, t'') : R, t'' : C, \\
&\qquad\qquad b : \neg A \sqcup \exists R.C, b : \exists R.C, (b, t''') : R, t''' : C\}\}
\end{aligned}
$$

Observations:

- No more rules applicable
- All ABoxes in $\mathcal{S}$ are consistent

$\rightarrow$ The knowledge base $\mathcal{K}$ is consistent.

The Tableau Algorithm - Observations

- If a rule has been applied to an ABox in $\mathcal{S}$ it will never be applied again (with the same parameters)

The Tableau Algorithm - Observations

- If a rule has been applied to an ABox in $\mathcal{S}$ it will never be applied again (with the same parameters)
- Only the $\sqcup$-rule adds a new ABox to $\mathcal{S}$

The Tableau Algorithm - Observations

- If a rule has been applied to an ABox in $\mathcal{S}$ it will never be applied again (with the same parameters)
- Only the $\sqcup$-rule adds a new ABox to $\mathcal{S}$
- If $\mathcal{A}'$ replaces $\mathcal{A}$ then $\mathcal{A} \subseteq \mathcal{A}'$

The Tableau Algorithm - Observations

- If a rule has been applied to an ABox in $\mathcal{S}$ it will never be applied again (with the same parameters)
- Only the $\sqcup$-rule adds a new ABox to $\mathcal{S}$
- If $\mathcal{A}'$ replaces $\mathcal{A}$ then $\mathcal{A} \subseteq \mathcal{A}'$

Theorem

- *If the tableau algorithm terminates with $\mathcal{S} = \emptyset$ then $\mathcal{K}$ is inconsistent*
- *If the tableau algorithm terminates with $\mathcal{S} \neq \emptyset$ then $\mathcal{K}$ is consistent*

WeST People and Knowledge Networks

The Tableau Algorithm - Observations

- If a rule has been applied to an ABox in $\mathcal{S}$ it will never be applied again (with the same parameters)
- Only the $\sqcup$-rule adds a new ABox to $\mathcal{S}$
- If $\mathcal{A}'$ replaces $\mathcal{A}$ then $\mathcal{A} \subseteq \mathcal{A}'$

Theorem

- *If the tableau algorithm terminates with $\mathcal{S} = \emptyset$ then $\mathcal{K}$ is inconsistent*
- *If the tableau algorithm terminates with $\mathcal{S} \neq \emptyset$ then $\mathcal{K}$ is consistent*

What about termination?

The Tableau Algorithm - Blocking Rules

What happens when applying the tableau algorithm to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

The Tableau Algorithm - Blocking Rules

What happens when applying the tableau algorithm to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

$\rightarrow$ Infinite application of the $\sqsubseteq$- and $\sqcup$- and $\exists$-rules.

The Tableau Algorithm - Blocking Rules

What happens when applying the tableau algorithm to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

$\rightarrow$ Infinite application of the $\sqsubseteq$- and $\sqcup$- and $\exists$-rules.

To ensure termination we introduce the notion of *block*. If

- t is an individual created by application of a rule and
- there is an individual t' with
  1. $\{C \mid t : C \in \mathcal{A}\} \subseteq \{C \mid t' : C \in \mathcal{A}\}$ and
  2. t' has been created before t

then t is blocked (by t').

WeST People and Knowledge Networks

The Tableau Algorithm - Blocking Rules cont'd

- $\sqcap$-rule: if $t : C_1 \sqcap C_2 \in \mathcal{A}'$, t **is not blocked**, and $\{t : C_1, t : C_2\} \not\subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : C_1, t : C_2\}$ to $\mathcal{S}$.
- $\sqcup$-rule: if $t : C_1 \sqcup C_2 \in \mathcal{A}'$, t **is not blocked**, and $\{t : C_1, t : C_2\} \cap \mathcal{A}' = \emptyset$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add both $\mathcal{A}' \cup \{t : C_1\}$ and $\mathcal{A}' \cup \{t : C_2\}$ to $\mathcal{S}$.
- $\exists$-rule: if $t : \exists R.C \in \mathcal{A}'$, t **is not blocked**, and there is no t' with $\{(t, t') : R, t' : C\} \subseteq \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$, *create* a new individual t'', and add $\mathcal{A}' \cup \{(t, t'') : R, t'' : C\}$ to $\mathcal{S}$.
- $\forall$-rule: if $\{t : \forall R.C, (t, t') : R\} \subseteq \mathcal{A}'$, t **is not blocked**, and $\{t' : C\} \notin \mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t' : C\}$ to $\mathcal{S}$.
- $\sqsubseteq$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$, t **is not blocked**, and $t : (\neg C_1 \sqcup C_2) \notin \mathcal{A}'$ for t appearing in $\mathcal{A}'$ then remove $\mathcal{A}'$ from $\mathcal{S}$ and add $\mathcal{A}' \cup \{t : (\neg C_1 \sqcup C_2)\}$ to $\mathcal{S}$.

The Tableau Algorithm - Blocking Rules cont'd

What happens now to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

The Tableau Algorithm - Blocking Rules cont'd

What happens now to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

Theorem

- *When the tableau algorithm with blocking terminates with $\mathcal{S} = \emptyset$ then $\mathcal{K}$ is inconsistent*

The Tableau Algorithm - Blocking Rules cont'd

What happens now to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

Theorem

- *When the tableau algorithm with blocking terminates with $\mathcal{S} = \emptyset$ then $\mathcal{K}$ is inconsistent*
- *When the tableau algorithm with blocking terminates with $\mathcal{S} \neq \emptyset$ then $\mathcal{K}$ is consistent*

The Tableau Algorithm - Blocking Rules cont'd

What happens now to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

Theorem

- *When the tableau algorithm with blocking terminates with* $\mathcal{S} = \emptyset$ *then* $\mathcal{K}$ *is inconsistent*
- *When the tableau algorithm with blocking terminates with* $\mathcal{S} \neq \emptyset$ *then* $\mathcal{K}$ *is consistent*
- *The tableau algorithm with blocking always terminates on* $\mathcal{ALC}$.

The Tableau Algorithm - Blocking Rules cont'd

What happens now to

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.A \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \} \qquad ?$$

Theorem

- *When the tableau algorithm with blocking terminates with $\mathcal{S} = \emptyset$ then $\mathcal{K}$ is inconsistent*
- *When the tableau algorithm with blocking terminates with $\mathcal{S} \neq \emptyset$ then $\mathcal{K}$ is consistent*
- *The tableau algorithm with blocking always terminates on $\mathcal{ALC}$.*
- *The tableau algorithm with blocking runs in EXPSPACE (worst case).*

Outline

Ontology languages revisited

- $\mathcal{ALC}$ is just one example of a description logic

Ontology languages revisited

- $\mathcal{ALC}$ is just one example of a description logic
- Over the years a lot of different description logics have been proposed that differ in
  - complexity
  - expressiveness

WeST People and Knowledge Networks

Ontology languages revisited

- $\mathcal{ALC}$ is just one example of a description logic
- Over the years a lot of different description logics have been proposed that differ in
  - complexity
  - expressiveness
- The search for the *right* description logic is ongoing

Ontology languages revisited

- $\mathcal{ALC}$ is just one example of a description logic
- Over the years a lot of different description logics have been proposed that differ in
  - complexity
  - expressiveness
- The search for the *right* description logic is ongoing
- There is always the issue of balancing between complexity and expressiveness

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Further *features* include

- $\mathcal{N}$: unqualified number restrictions: $(\geq_3 \textit{hasChild})$

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Further *features* include

- $\mathcal{N}$: unqualified number restrictions: $(\geq_3 \textit{hasChild})$
- $\mathcal{Q}$ualified number restrictions: $(\geq_2 \textit{hasChild}.\textit{Female})$

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Further *features* include

- $\mathcal{N}$: unqualified number restrictions: $(\geq_3 \, hasChild)$
- $\mathcal{Q}$ualified number restrictions: $(\geq_2 \, hasChild.Female)$
- $\mathcal{O}$ne-of (nominals): $\{t_1, \ldots, t_n\}$

WeST People and Knowledge Networks

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Further *features* include

- $\mathcal{N}$: unqualified number restrictions: $(\geq_3 \mathit{hasChild})$
- $\mathcal{Q}$ualified number restrictions: $(\geq_2 \mathit{hasChild}.\mathit{Female})$
- $\mathcal{O}$ne-of (nominals): $\{t_1, \ldots, t_n\}$
- $\mathcal{F}$unctionality: $(\leq \mathit{hasFather})$

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Further *features* include

- $\mathcal{N}$: unqualified number restrictions: $(\geq_3 \; hasChild)$
- $\mathcal{Q}$ualified number restrictions: $(\geq_2 \; hasChild.Female)$
- $\mathcal{O}$ne-of (nominals): $\{t_1, \ldots, t_n\}$
- $\mathcal{F}$unctionality: $(\leq hasFather)$
- Role operators:
  - $\mathcal{I}$: role inverse: $hasChild^- \equiv hasParent$

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Further *features* include

- $\mathcal{N}$: unqualified number restrictions: $(\geq_3 \mathit{hasChild})$
- $\mathcal{Q}$ualified number restrictions: $(\geq_2 \mathit{hasChild}.\mathit{Female})$
- $\mathcal{O}$ne-of (nominals): $\{t_1, \ldots, t_n\}$
- $\mathcal{F}$unctionality: $(\leq \mathit{hasFather})$
- Role operators:
  - $\mathcal{I}$: role inverse: $\mathit{hasChild}^- \equiv \mathit{hasParent}$
  - $\mathcal{S}$: Transitive roles $tr(R)$ $(tr(\mathit{hasParent}) \equiv \mathit{hasAncestor})$

WeST People and Knowledge Networks

Ontology languages revisited

The *base* description logic is $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements).

Further *features* include

- $\mathcal{N}$: unqualified number restrictions: $(\geq_3 \mathit{hasChild})$
- $\mathcal{Q}$ualified number restrictions: $(\geq_2 \mathit{hasChild}.\mathit{Female})$
- $\mathcal{O}$ne-of (nominals): $\{t_1, \ldots, t_n\}$
- $\mathcal{F}$unctionality: $(\leq \mathit{hasFather})$
- Role operators:
  - $\mathcal{I}$: role inverse: $\mathit{hasChild}^- \equiv \mathit{hasParent}$
  - $\mathcal{S}$: Transitive roles $tr(R)$ $(tr(\mathit{hasParent}) \equiv \mathit{hasAncestor})$
  - $\mathcal{H}$: role hierarchies: $R \circ R' \subseteq R''$ $(\mathit{hasParent} \circ \mathit{hasParent} \equiv \mathit{hasGrandparent})$
  - ...

WeST People and Knowledge Networks

Ontology languages revisited

- Other description logic types can be described by their names:
  - $\mathcal{ALCQIO}$: $\mathcal{ALC}$ with qualified number restrictions, inverse roles, and nominals.

Ontology languages revisited

- Other description logic types can be described by their names:
  - $\mathcal{ALCQIO}$: $\mathcal{ALC}$ with qualified number restrictions, inverse roles, and nominals.
  - $\mathcal{SHOIN}$: $\mathcal{ALC}$ with transitive roles, role hierarchies, role inverse, nominals, unqualified number restrictions (this is the same as OWL-DL)

WeST People and Knowledge Networks

Ontology languages revisited

- Other description logic types can be described by their names:
  - $\mathcal{ALCQIO}$: $\mathcal{ALC}$ with qualified number restrictions, inverse roles, and nominals.
  - $\mathcal{SHOIN}$: $\mathcal{ALC}$ with transitive roles, role hierarchies, role inverse, nominals, unqualified number restrictions (this is the same as OWL-DL)
- Some description languages with further restrictions:
  - $\mathcal{EL}$: Only $C_1 \sqcap C_2$ and $\exists R.\top$ allowed

Ontology languages revisited

- Other description logic types can be described by their names:
  - $\mathcal{ALCQIO}$: $\mathcal{ALC}$ with qualified number restrictions, inverse roles, and nominals.
  - $\mathcal{SHOIN}$: $\mathcal{ALC}$ with transitive roles, role hierarchies, role inverse, nominals, unqualified number restrictions (this is the same as OWL-DL)
- Some description languages with further restrictions:
  - $\mathcal{EL}$: Only $C_1 \sqcap C_2$ and $\exists R.\top$ allowed
  - $\mathcal{EL}++$: $\mathcal{EL}$ with nominals and some additional role operators

WeST People and Knowledge Networks

Outline

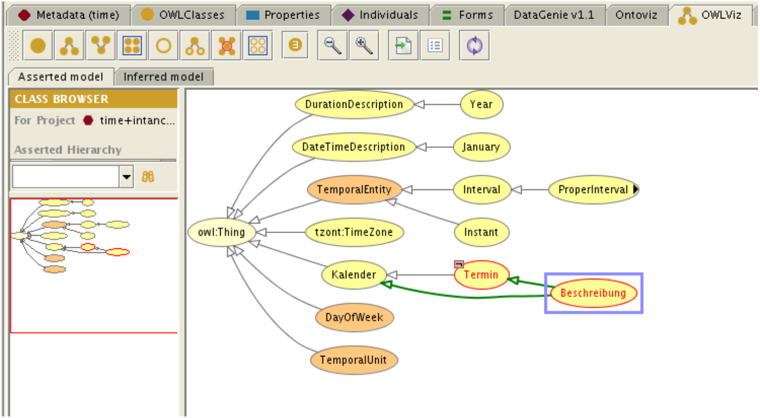
Tools

- In contrast to many other KR languages (propositional logic, default logic, ...) description logics have been developed out of the need to apply them

Tools

- In contrast to many other KR languages (propositional logic, default logic, ...) description logics have been developed out of the need to apply them
- Therefore, a lot of tools are around that enable ontology engineering using description logics

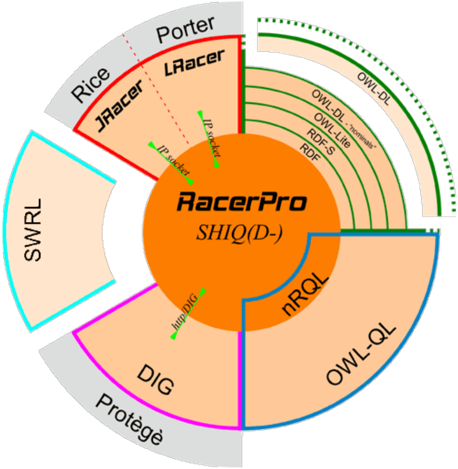
Tools: Protégé



- Free open-source ontology editor
- Webpage: `http://protege.stanford.edu`

WeST People and Knowledge Networks

Tools: RacerPro



- Commercial description logic reasoner
- Webpage: `http://www.franz.com/agraph/racer`

Further Tools

HermiT

- Free open-source description logic reasoner (OWL)
- Webpage: `http://hermit-reasoner.com`

Further Tools

HermiT

- Free open-source description logic reasoner (OWL)
- Webpage: `http://hermit-reasoner.com`

FacT++

- Free open-source description logic reasoner
- Webpage: `http://owl.man.ac.uk/factplusplus/`

Further Tools

HermiT

- Free open-source description logic reasoner (OWL)
- Webpage: `http://hermit-reasoner.com`

FacT++

- Free open-source description logic reasoner
- Webpage: `http://owl.man.ac.uk/factplusplus/`

OWL API

- Official open source JAVA interfaces for programming DL applications
- Webpage: `http://owlapi.sourceforge.net`

WeST People and Knowledge Networks

Outline

WeST
People and Knowledge Networks

Summary

- The tableau algorithm for $\mathcal{ALC}$:
  - checks consistency of a knowledge base
  - sound and complete
  - terminates always when using blocks
- Ontology languages revisited
  - Nomenclature of description logics
  - expressivity vs. complexity
- Tools for working with description logics

WeST People and Knowledge Networks

Pointers to further reading

- The Description Logic Complexity Navigator
  http://www.cs.man.ac.uk/~ezolin/dl/
- Franz Baader. Tableau Algorithms for Description Logics.
  http://lat.inf.tu-dresden.de/~baader/Talks/Tableaux2000.pdf
- Franz Baader, Ulrike Sattler. An Overview of Tableau Algorithms for Description Logics. http://www.eecs.yorku.ca/course_archive/2010-11/F/6390A/DLmaterial/BaaderSattler-tableauxForDL.pdf

Exercises

- Apply the tableau algorithm to check whether the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent:

$$\mathcal{T} = \{ \quad A \sqsubseteq C, B \sqsubseteq \neg C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, a : B \quad \}$$

Exercises

- Apply the tableau algorithm to check whether the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent:

$$\mathcal{T} = \{ \quad A \sqsubseteq C, B \sqsubseteq \neg C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, a : B \quad \}$$

- Download Protégé (`http://protege.stanford.edu`) and play around with it (Home assignment)

Exercises

- Apply the tableau algorithm to check whether the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent:

$$\mathcal{T} = \{ \quad A \sqsubseteq C, B \sqsubseteq \neg C \quad \}$$
$$\mathcal{A} = \{ \quad a : A, a : B \quad \}$$

- Download Protégé (`http://protege.stanford.edu`) and play around with it (Home assignment)
- Apply the tableau algorithm with blocking to check whether the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent:

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists R.B, \quad B \sqsubseteq A \sqcap \forall S.C \quad \}$$
$$\mathcal{A} = \{ \quad a : A \quad \}$$

(Home assignment)