Semantic Web

2. Description logics

PD Dr. Matthias Thimm

thimm@uni-koblenz.de

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

Overview

- Description logics are the formal basis for most knowledge representation formalisms on the semantic web
- Description logic knowledge bases are a way to formalize *ontologies*
- We will discuss general concepts of ontology formalization and the description logic $\mathcal{ALC}$
  - Syntax
  - Semantics
  - Inference

Outline

Outline

Ontologies 1/3

Ontology is the philosophical study of the nature of being, becoming, existence, or reality, as well as the basic categories of being and their relations.

In computer science and information science, an ontology formally represents knowledge as a set of concepts within a domain, and the relationships between pairs of concepts. It can be used to model a domain and support reasoning about concepts.

[Wikipedia]

Ontologies 2/3

- In computer science an ontology is an engineering artifact
  - with a specific vocabulary
  - with a set of assumptions on the meaning of the terms in the vocabulary
- shared understanding of a specific domain
- formal
- from the perspective of artificial intelligence
  - ontology = knowledge base (logical theory)
- from the perspective of data bases
  - ontology = conceptual data model

Ontologies 3/3

Examples

- Biology: Inheritance hierarchies
- Life sciences:
  - disease classifications,
  - anatomy
  - genetics
  - proteins
- Linguistics: word relations, synonyms, metonyms, etc. (WordNet)
- Economics: process modeling

Structure of an ontology

Ontologies are comprised of different components:

- Individuals, entities, instances, objects: *Tweety*, *Hammer07*
- Classes, concepts: *Bird*, *Tool*
- Attributes, properties: *hasColor*
- Relations: *friendOf*, *knows*
- Functional relations: *fatherOf*
- Axioms: *Bird isA Animal*

Ontology languages

- Graphical languages
  - Inheritance networks
  - Semantic networks
- Logic-based languages
  - Description logics ($\mathcal{ALC}$, $\mathcal{SHOIN}$, OIL, OWL)
  - Rule languages (Answer set programming)
  - First-order logic
  - Non-classical logics
- Object-oriented languages
  - UML
  - Z
- Web schema languages
  - XML/XMLS (not an ontology language per se)
  - RDF/RDFS

Ontology engineering

- An ontology (or any other form of knowledge base)
  - can be used for reasoning, answering queries
  - should be reusable
- Process of building an ontology is called *ontology engineering*
  1. Define language
     - terms
     - concepts,
     - relations, etc.
  2. Define knowledge
     - What are equivalent concepts?
     - Are there subset relations?
     - Which constraints have to be imposed? etc.
  3. Use ontology for reasoning

$\rightarrow$ more on that later

Outline

Description logic $\mathcal{ALC}$ - Overview 1/2

Ontologies can be particularly well described using *description logics*.

We use $\mathcal{ALC}$ (*Attributive Language with Complements*) as an example (Schmidt-Schauß, Smolka; 1991).

Description logic $\mathcal{ALC}$ - Overview 2/2

Languages building on the logic $\mathcal{ALC}$ use a strict grammar to define their syntax.

The basic ingredients for the definition of a logical language are

1. Definition of the *signature*:
   - What are the atomic elements?
2. Definition of the *syntax*:
   - How are complex formulas built from atomic elements?
3. Definition of the *semantics*:
   - What is the meaning of formulas?
   - What can be inferred from formulas?

Description logic $\mathcal{ALC}$ - Signature

A *signature* $\mathcal{S} = (N_C, N_R, N_O)$ for an $\mathcal{ALC}$ knowledge base contains

- a set N_C of atomic concept names
  - Example: *Human*, *Animal*, *FlyingObject*
  - We always assume $\top, \bot \in N_C$
  - $\top$: the concept of everything
  - $\bot$: the concept of nothing
- a set N_R of atomic relation names
  - Example: *friendOf*, *hasColor*, *fatherOf*
  - *Remark*: properties, relations, and functional expressions are indistinguishable in most DLs
- a set N_O of individual names
  - Example: *Carl*, *Dave*, *Hammer07*

Description logic $\mathcal{ALC}$ - Syntax

The syntax of a logic describes how complex formulas are build from atomic elements.

$\mathcal{ALC}$ uses the following *connectives* to combine atomic elements:

$\sqcap$	intersection or conjunction of concepts
$\sqcup$	union or disjunction of concepts
$\neg$	complement of a concept
$\forall$	universal restriction
$\exists$	existential restriction
$\sqsubseteq$	concept inclusion
$:$	concept/role assertion

Description logic $\mathcal{ALC}$ - Concepts 1/2

Let $\mathcal{S} = (N_C, N_R, N_O)$ be an $\mathcal{ALC}$ signature.

The set $\mathcal{C}_\mathcal{S}$ of valid *concepts* of $\mathcal{S}$ is the minimal set $\mathcal{C}$ satisfying

1. $N_C \subseteq \mathcal{C}$
   - every atomic concept is a concept
2. for every $C_1, C_2 \in \mathcal{C}$ we have $C_1 \sqcap C_2 \in \mathcal{C}$
   - the intersection of two concepts is a concept as well
   - Example: Female $\sqcap$ Architect (the concept of all female architects)
3. for every $C_1, C_2 \in \mathcal{C}$ we have $C_1 \sqcup C_2 \in \mathcal{C}$
   - the union of two concepts is a concept as well
   - Example: Student $\sqcup$ Professor (the concept of all people who are students or professors)

Description logic $\mathcal{ALC}$ - Concepts 2/2

4. for every $C \in \mathcal{C}$ we have $\neg C \in \mathcal{C}$
   - the complement of a concept is a concept as well
   - Example: $\neg$Student (the concept of all people who are not a student)
5. for every $C \in \mathcal{C}, R \in N_R$ we have $\forall R.C \in \mathcal{C}$
   - The set of individuals whose R-successors are in C is a concept
   - Example: $\forall$hasChild.Female
     (the concept of all people who *only* have daughters)
6. for every $C \in \mathcal{C}, R \in N_R$ we have $\exists R.C \in \mathcal{C}$
   - The set of individuals where one R-successor is in C is a concept
   - Example: $\exists$visited.GermanCity
     (the concept of all people who visited *some* German city)

Description logic $\mathcal{ALC}$ - Axioms 1/2

$\mathcal{ALC}$ distinguishes between two types of axioms (formulas, sentences):

1. Terminological axioms: for general statements about concepts and roles
2. Assertional axioms: for specific statements about individuals

The set of *terminological axioms* $\mathfrak{T}_\mathcal{S}$ of $\mathcal{S}$ is the minimal set $\mathfrak{T}$ satisfying

- for every $C_1, C_2 \in \mathcal{C}_\mathcal{S}$ we have $C_1 \sqsubseteq C_2 \in \mathfrak{T}$
  - C_1 is a subconcept of C_2 ("C_2 subsumes C_1")
  - Example: Woman $\sqsubseteq$ Human (every woman is a human)

Description logic $\mathcal{ALC}$ - Axioms 2/2

The set of *assertional axioms* $\mathfrak{A}_{\mathcal{S}}$ of $\mathcal{S}$ is the minimal set $\mathfrak{A}$ satisfying

- for every $C \in \mathcal{C}_{\mathcal{S}}$ and $t \in N_O$ we have $t : C \in \mathfrak{A}$
  - t satisfies concept C ("t is an element of concept C")
  - Example: $\mathit{Carl} : \mathit{Human}$ ("Carl is a human")
- for every $R \in N_R$ and $t, t' \in N_O$ we have $(t, t') : R \in \mathfrak{A}$
  - t and t' are in relation R
  - Example: $(\mathit{Carl}, \mathit{Dave}) : \mathit{fatherOf}$ ("Carl is the father of Dave")

Description logic $\mathcal{ALC}$ - Knowledge base

A *knowledge base* contains the axioms explicitly accepted to be true.

In $\mathcal{ALC}$ a knowledge base is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with

- $\mathcal{T} \subseteq \mathfrak{T}_{\mathcal{S}}$ is a (finite) set of terminological axioms ($\mathcal{T}$ is also called TBox)
- $\mathcal{A} \subseteq \mathfrak{A}_{\mathcal{S}}$ is a (finite) set of assertional axioms ($\mathcal{A}$ is also called ABox)

Example

Let $\mathcal{S}_1 = (N_C, N_R, N_O)$ be the signature defined via

$$\begin{aligned} N_C &= \{\text{Human}, \text{Man}, \text{Woman}, \text{Architect}, \text{Father}\} \\ N_R &= \{\text{hasChild}, \text{friendOf}\} \\ N_O &= \{\text{Carl}, \text{Dave}, \text{Anna}\} \end{aligned}$$

Consider $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A}_1)$ given by

$$\begin{aligned} \mathcal{T}_1 = \{ \quad & \text{Architect} \sqsubseteq \text{Human} \\ & \text{Man} \sqcup \text{Woman} \sqsubseteq \text{Human} \\ & \text{Human} \sqsubseteq \forall \text{hasChild}.\text{Human} \\ & \text{Father} \sqsubseteq \text{Man} \sqcap \exists \text{hasChild}.\top \quad \} \\ \mathcal{A}_1 = \{ \quad & \text{Dave} : \text{Architect}, \text{Anna} : \text{Woman}, \\ & \text{Carl} : \text{Father}, (\text{Dave}, \text{Anna}) : \text{hasChild} \quad \} \end{aligned}$$

- What is the meaning of the concepts above?
- What can be “inferred” from the knowledge base?

Description logic $\mathcal{ALC}$ - Semantics 1/4

Semantics specify the *meaning* of concepts and axioms with respect to the "real world".

Let $\mathcal{S} = (N_C, N_R, N_O)$ be a signature. An *interpretation* $\mathcal{I}$ for $\mathcal{S}$ is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ with

1. $\Delta^\mathcal{I}$ is a non-empty set called the *domain* or the *universe*
2. $\cdot^\mathcal{I}$ is the *interpretation function* which maps
    - every $t \in N_O$ to an element $t^\mathcal{I} \in \Delta^\mathcal{I}$
    - every $C \in N_C$ to a set $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$
    - every $R \in N_R$ to a set $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ describes a *possible world*, i. e. a complete description of the things that are true in some assumed situation.

Let $\mathbb{I}_\mathcal{S}$ denote the set of all interpretations for $\mathcal{S}$.

Description logic $\mathcal{ALC}$ - Semantics 2/4

Let $\mathcal{S} = (N_C, N_R, N_O)$ be a signature and $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ an interpretation.

We abbreviate

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(C_1 \sqcup C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{t \in \Delta^{\mathcal{I}} \mid \text{for all } t' \text{ with } (t,t') \in R^{\mathcal{I}} \text{ it is } t' \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{t \in \Delta^{\mathcal{I}} \mid \text{there is } t' \text{ with } (t,t') \in R^{\mathcal{I}} \text{ and } t' \in C^{\mathcal{I}}\}
\end{aligned}$$

Description logic $\mathcal{ALC}$ - Semantics 3/4

Let $\mathcal{S} = (N_C, N_R, N_O)$ be a signature and $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ an interpretation.

$\mathcal{I}$ satisfies an axiom A, written $\mathcal{I} \models A$, if

- $\mathcal{I} \models C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
- $\mathcal{I} \models t : C$ if $t^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I} \models (t_1, t_2) : R$ if $(t_1^{\mathcal{I}}, t_2^{\mathcal{I}}) \in R^{\mathcal{I}}$

Description logic $\mathcal{ALC}$ - Semantics 4/4

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies a set of axioms $\mathcal{F} \subseteq \mathfrak{T}_{\mathcal{S}} \cup \mathfrak{A}_{\mathcal{S}}$, written $\mathcal{I} \models \mathcal{F}$ if

$$\forall A \in \mathcal{F} : \mathcal{I} \models A$$

An interpretation $\mathcal{I}$ satisfies a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, written $\mathcal{I} \models \mathcal{K}$ if

- $\mathcal{I} \models \mathcal{T}$
- $\mathcal{I} \models \mathcal{A}$

If $\mathcal{I} \models \mathcal{K}$ then $\mathcal{I}$ is called a *model* of $\mathcal{K}$.

Example 1/2

Let $\mathcal{S}_1 = (N_C, N_R, N_O)$ be the signature defined via

$$\begin{aligned}
N_C &= \{\text{Human}, \text{Man}, \text{Woman}, \text{Architect}, \text{Father}\}\\
N_R &= \{\text{hasChild}, \text{friendOf}\}\\
N_O &= \{\text{Carl}, \text{Dave}, \text{Anna}\}
\end{aligned}$$

Consider the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ defined via

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \{ && \text{CarlCarlson}, \text{DavidDavidson}, \text{AnnaAndrews} \}\\
\text{Carl}^{\mathcal{I}} &= && \text{CarlCarlson} \qquad \ldots\\
\text{Human}^{\mathcal{I}} &= \{ && \text{CarlCarlson}, \text{DavidDavidson}, \text{AnnaAndrews} \}\\
\text{Man}^{\mathcal{I}} &= \{ && \text{CarlCarlson}, \text{DavidDavidson} \}\\
\text{Architect}^{\mathcal{I}} &= \{ && \text{DavidDavidson} \}\\
\text{Father}^{\mathcal{I}} &= \{ && \text{CarlCarlson}, \text{DavidDavidson} \}\\
\text{Woman}^{\mathcal{I}} &= \{ && \text{AnnaAndrews} \}\\
\text{hasChild}^{\mathcal{I}} &= \{ && (\text{DavidDavidson}, \text{AnnaAndrews}), (\text{CarlCarlson}, \text{DavidDavidson})\}\\
\text{friendOf}^{\mathcal{I}} &= \{ && (\text{AnnaAndrews}, \text{DavidDavidson}) \}
\end{aligned}$$

Example 2/2

Some observations:

$$
\begin{aligned}
(\mathsf{Man} \sqcap \mathsf{Woman})^{\mathcal{I}} &= \mathsf{Man}^{\mathcal{I}} \cap \mathsf{Woman}^{\mathcal{I}} = \emptyset \\
(\mathsf{Man} \sqcup \mathsf{Woman})^{\mathcal{I}} &= \mathsf{Man}^{\mathcal{I}} \cup \mathsf{Woman}^{\mathcal{I}} \\
&= \{\mathsf{CarlCarlson}, \mathsf{DavidDavidson}, \mathsf{AnnaAndrews}\}
\end{aligned}
$$

$$
\begin{aligned}
&(\mathsf{Man} \sqcap \exists \mathsf{hasChild}.\top)^{\mathcal{I}} = \mathsf{Man}^{\mathcal{I}} \cap (\exists \mathsf{hasChild}.\top)^{\mathcal{I}} \\
&\quad = \mathsf{Man}^{\mathcal{I}} \cap \{t \in \Delta^{\mathcal{I}} \mid \text{there is } t' \text{ with } (t, t') \in \mathsf{hasChild}^{\mathcal{I}} \text{ and } t' \in \top^{\mathcal{I}}\} \\
&\quad = \{\mathsf{CarlCarlson}, \mathsf{DavidDavidson}\}
\end{aligned}
$$

It follows

- $\mathcal{I} \models \mathsf{Father} \sqsubseteq \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\top$
- $\mathcal{I} \models \mathsf{Dave} : \mathsf{Architect}$
- $\mathcal{I} \models \mathsf{Anna} : \mathsf{Woman}$

Description logic $\mathcal{ALC}$ - Inferences 1/2

Inference describes the process from drawing new conclusions from a knowledge base.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base. We say that $\mathcal{K}$ *entails* an axiom $A \in \mathfrak{T}_{\mathcal{S}} \cup \mathfrak{A}_{\mathcal{S}}$, written $\mathcal{K} \models A$, if

> *for every interpretation $\mathcal{I} \in \mathbb{I}_{\mathcal{S}}$ we have $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models A$*

Note that for every $C_1 \sqsubseteq C_2 \in \mathcal{T}$ we automatically have $\mathcal{K} \models C_1 \sqsubseteq C_2$ (the same is true for the ABox $\mathcal{A}$).

Description logic $\mathcal{ALC}$ - Inferences 2/2

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

$\rightarrow$ more on that later

Example

Consider again $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A}_1)$ given by

$$\begin{aligned}
\mathcal{T}_1 = \{ \quad & \mathsf{Architect} \sqsubseteq \mathsf{Human} \\
& \mathsf{Man} \sqcup \mathsf{Woman} \sqsubseteq \mathsf{Human} \\
& \mathsf{Human} \sqsubseteq \forall \mathsf{hasChild}.\mathsf{Human} \\
& \mathsf{Father} \sqsubseteq \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\top \quad \} \\
\mathcal{A}_1 = \{ \quad & \mathsf{Dave} : \mathsf{Architect}, \mathsf{Anna} : \mathsf{Woman}, \\
& \mathsf{Carl} : \mathsf{Father}, (\mathsf{Dave}, \mathsf{Anna}) : \mathsf{hasChild} \quad \}
\end{aligned}$$

Here we have

- $\mathcal{K} \models \mathsf{Dave} : \mathsf{Human}$
- $\mathcal{K} \models \mathsf{Carl} : \exists \mathsf{hasChild}.\mathsf{Human}$
- $\mathcal{K} \not\models \mathsf{Man} \sqcap \mathsf{Woman} \sqsubseteq \bot \qquad (!)$

$\mathcal{ALC}$ and first-order logic 1/2

- Description logics have been invented as a modeling language for taxonomic/ontological knowledge
- Formally, most of them are (decidable) fragments of first-order logic

Comparison of notation:

$\mathcal{ALC}$ term	FOL term
Concept	(unary) Predicate
Relation	(binary) Predicate
Individual	Individual

$\mathcal{ALC}$ and first-order logic 2/2

Comparison of formulas

$\mathcal{ALC}$ axiom	FOL formula
$t : C$	$C(t)$
$(t, t') : R$	$R(t, t')$
$C_1 \sqsubseteq C_2$	$\forall X : C_1(X) \Rightarrow C_2(X)$
$C_1 \sqsubseteq C_2 \sqcap C_3$	$\forall X : C_1(X) \Rightarrow (C_2(X) \wedge C_3(X))$
$C_1 \sqsubseteq \forall R.C_2$	$\forall X : C_1(X) \Rightarrow (\forall Y : R(X, Y) \Rightarrow C_2(Y))$
$C_1 \sqsubseteq \exists R.C_2$	$\forall X : C_1(X) \Rightarrow (\exists Y : R(X, Y) \wedge C_2(Y))$

Outline

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 conclusion
- We now have a look at a very simple proof procedure for deciding consistency: the *Tableau Algorithm*

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

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* 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*:

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

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

- $\sqcap$-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 $\mathcal{S}$.

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 $\mathcal{S}$.
- $\sqsubseteq$-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 - 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 1

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

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

Observe:

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

Initialize $\mathcal{S} = \{\{a : A, b : B\}\}$.

The Tableau Algorithm - Example 1 cont'd

Let $\mathcal{K} = (\mathcal{T}_1, \mathcal{A}_1)$ be given via

$$\begin{aligned}\mathcal{T} &= \{ \quad A \sqsubseteq B \quad \} \\ \mathcal{A} &= \{ \quad a : A, b : B \quad \} \\ \mathcal{S} &= \{\{a : A, b : B\}\}\end{aligned}$$

$\sqsubseteq$-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 $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : B, a : \neg A \sqcup B\}\}$$

The Tableau Algorithm - Example 1 cont'd

Let $\mathcal{K} = (\mathcal{T}_1, \mathcal{A}_1)$ be given via

$$\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq B \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : B \quad \} \\
\mathcal{S} &= \{\{a : A, b : B, a : \neg A \sqcup B\}\}
\end{aligned}$$

$\sqcup$-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 $\mathcal{S}$.

$$\begin{aligned}
\mathcal{S} = \{&\{a : A, b : B, a : \neg A \sqcup B, {\color{red} a : \neg A}\}, \\
&\{a : A, b : B, a : \neg A \sqcup B, {\color{red} a : B}\}\}
\end{aligned}$$

The Tableau Algorithm - Example 1 cont'd

Let $\mathcal{K} = (\mathcal{T}_1, \mathcal{A}_1)$ be given via

$$\mathcal{T} = \{ \quad A \sqsubseteq B \quad \}$$
$$\mathcal{A} = \{ \quad a : A, b : B \quad \}$$
$$\mathcal{S} = \{\{a : A, b : B, a : \neg A \sqcup B\}\}$$

$\sqcup$-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 $\mathcal{S}$.

$$\mathcal{S} = \{\sout{\{a : A, b : B, a : \neg A \sqcup B, \textcolor{red}{a : \neg A}\}}, \\ \{a : A, b : B, a : \neg A \sqcup B, \textcolor{red}{a : B}\}\}$$

The Tableau Algorithm - Example 1 cont'd

Let $\mathcal{K} = (\mathcal{T}_1, \mathcal{A}_1)$ be given via

$$\begin{aligned} \mathcal{T} &= \{ \quad A \sqsubseteq B \quad \} \\ \mathcal{A} &= \{ \quad a : A, b : B \quad \} \\ \mathcal{S} &= \{\{a : A, b : B, a : \neg A \sqcup B, a : B\}\} \end{aligned}$$

$\sqsubseteq$-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 $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, b : B, a : \neg A \sqcup B, a : B, b : \neg A \sqcup B\}\}$$

The Tableau Algorithm - Example 1 cont'd

Let $\mathcal{K} = (\mathcal{T}_1, \mathcal{A}_1)$ be given via

$$\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq B \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : B \quad \} \\
\mathcal{S} &= \{\{a : A, b : B, a : \neg A \sqcup B, a : B, b : \neg A \sqcup B\}\}
\end{aligned}$$

$\sqcup$-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 $\mathcal{S}$.

$$\begin{aligned}
\mathcal{S} = \{&\{a : A, b : B, a : \neg A \sqcup B, a : B, b : \neg A \sqcup B, b : \neg A\}, \\
&\{a : A, b : B, a : \neg A \sqcup B, a : B, b : \neg A \sqcup B, b : B\}\}
\end{aligned}$$

The Tableau Algorithm - Example 1 cont'd

Let $\mathcal{K} = (\mathcal{T}_1, \mathcal{A}_1)$ be given via

$$\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq B \quad \} \\
\mathcal{A} &= \{ \quad a : A, b : B \quad \} \\
\mathcal{S} &= \{\{a : A, b : B, a : \neg A \sqcup B, a : B, b : \neg A \sqcup B, b : \neg A\}, \\
&\qquad \{a : A, b : B, a : \neg A \sqcup B, a : B, b : \neg A \sqcup B\}\}
\end{aligned}$$

- No more rule applicable
- $\mathcal{K}$ is consistent

The Tableau Algorithm - Example 2

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

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

Observe:

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

Initialize $\mathcal{S} = \{\{a : A, a : \neg D\}\}$.

The Tableau Algorithm - Example 2 cont'd

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

$$\begin{aligned}\mathcal{T} &= \{ \quad A \sqsubseteq B \sqcap D \quad \} \\ \mathcal{A} &= \{ \quad a : A, a : \neg D \quad \} \\ \mathcal{S} &= \{\{a : A, a : \neg D\}\}\end{aligned}$$

$\sqsubseteq$-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 $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D)\}\}$$

The Tableau Algorithm - Example 2 cont'd

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

$$\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq B \sqcap D \quad \} \\
\mathcal{A} &= \{ \quad a : A, a : \neg D \quad \} \\
\mathcal{S} &= \{\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D)\}\}
\end{aligned}$$

$\sqcup$-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 $\mathcal{S}$.

$$\begin{aligned}
\mathcal{S} = \{&\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), a : \neg A\}, \\
&\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), a : (B \sqcap D)\}\}
\end{aligned}$$

The Tableau Algorithm - Example 2 cont'd

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

$$\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq B \sqcap D \quad \} \\
\mathcal{A} &= \{ \quad a : A, a : \neg D \quad \} \\
\mathcal{S} &= \{\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D)\}\}
\end{aligned}$$

$\sqcup$-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 $\mathcal{S}$.

$$\begin{aligned}
\mathcal{S} = \{&\sout{\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), \color{red}{a : \neg A}\}}, \\
&\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), \color{red}{a : (B \sqcap D)}\}\}
\end{aligned}$$

The Tableau Algorithm - Example 2 cont'd

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

$$\begin{aligned}
\mathcal{T} &= \{ \quad A \sqsubseteq B \sqcap D \quad \} \\
\mathcal{A} &= \{ \quad a : A, a : \neg D \quad \} \\
\mathcal{S} &= \{\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), a : (B \sqcap D)\}\}
\end{aligned}$$

$\sqcap$-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 $\mathcal{S}$.

$$\mathcal{S} = \{\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), a : (B \sqcap D), a : B, a : D\}\}$$

The Tableau Algorithm - Example 2 cont'd

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

$$\begin{aligned} \mathcal{T} &= \{ \quad A \sqsubseteq B \sqcap D \quad \} \\ \mathcal{A} &= \{ \quad a : A, a : \neg D \quad \} \\ \mathcal{S} &= \{\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), a : (B \sqcap D)\}\} \end{aligned}$$

$\sqcap$-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 $\mathcal{S}$.

$\mathcal{S} = \{$~~$\{a : A, a : \neg D, a : \neg A \sqcup (B \sqcap D), a : (B \sqcap D), a : B, a : D$~~$\}\}$

The Tableau Algorithm - Example 2 cont'd

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

$$\begin{aligned}\mathcal{T} &= \{ \quad A \sqsubseteq B \sqcap D \quad \} \\ \mathcal{A} &= \{ \quad a : A, a : \neg D \quad \} \\ \mathcal{S} &= \emptyset\end{aligned}$$

- $\mathcal{K}$ is inconsistent

The Tableau Algorithm - Example 3

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

The Tableau Algorithm - Example 3 cont'd

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

$\sqsubseteq$-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 $\mathcal{S}$.

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

The Tableau Algorithm - Example 3 cont'd

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

$\sqcup$-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 $\mathcal{S}$.

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

The Tableau Algorithm - Example 3 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 $\mathcal{S}$.

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

The Tableau Algorithm - Example 3 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 3 cont'd

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

$\sqsubseteq$-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 $\mathcal{S}$.

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

The Tableau Algorithm - Example 3 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 3 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 3 cont'd

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

Observation: $\sqsubseteq$-rule is now applicable to t'' and t''', algorithm will not terminate

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

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

$\rightarrow$ 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').

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

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

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
- 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).

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: $R \circ R' \subseteq R''$
    $(\mathit{hasParent} \circ \mathit{hasParent} \equiv \mathit{hasGrandparent})$
  - ...

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

Outline

Summary

- 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
- 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

Pointers to further reading

- John F. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, 2000.
- Steffen Staab, Rudi Studer (Editors). Handbook on Ontologies, Springer Verlag, 2009.
- 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`