Vorlesung Künstliche Intelligenz Wintersemester 2006/07

Teil III:
Wissensrepräsentation und Inferenz

Kap. 11: Beschreibungslogiken

Mit Material von
• Carsten Lutz, Uli Sattler: http://www.computationallogic.org/content/events/iccl-ss-2005/lectures/lutz/index.php?id=24
• Ian Horrocks: http://www.cs.man.ac.uk/~horrocks/Teaching/cs646/

Beschreibungslogiken (Description Logics)

A family of logic based Knowledge Representation formalisms

- Descendants of semantic networks and KL-ONE
- Describe domain in terms of concepts (classes), roles (relationships) and individuals

Distinguished by:
- Formal semantics (typically model theoretic)
  - Decidable fragments of FOL
  - Closely related to Propositional Modal & Dynamic Logics
- Provision of inference services
  - Sound and complete decision procedures for key problems
  - Implemented systems (highly optimised)

- Einfache Sprache zum Start: \( ALC \) (Attributive Language with Complement)
- Im Semantic Web wird \( SHOIN(D_0) \) eingesetzt. Hierauf basiert die Semantik von OWL DL.

Geschichte

- Ihre Entwicklung wurde inspiriert durch semantische Netze und Frames.
- Frühere Namen:
  - KL-ONE like languages
  - terminological logics
- Ziel war eine Wissensrepräsentation mit formaler Semantik.

Literatur

Ontology/KR languages aim to model (part of) world

Terms in language correspond to entities in world

Meaning given by, e.g.:
- Mapping to another formalism, such as FOL, with own well defined semantics
- or a Model Theory (MT)

MT defines relationship between syntax and interpretations
- There can be many interpretations (models) of one piece of syntax
- Models supposed to be analogue of (part of) world
  - E.g., elements of model correspond to objects in world
- Formal relationship between syntax and models
  - Structure of models reflect relationships specified in syntax
- Inference (e.g., subsumption) defined in terms of MT
  - E.g., \( T \vdash A \subseteq B \) iff in every model of \( T \), \( \text{ext}(A) \subseteq \text{ext}(B) \)

Many logics (including standard First Order Logic) use a model theory based on (Zermelo-Frankel) set theory

The domain of discourse (i.e., the part of the world being modelled) is represented as a set (often refered as \( \Delta \))

Objects in the world are interpreted as elements of \( \Delta \)
- Classes/concepts (unary predicates) are subsets of \( \Delta \)
- Properties/roles (binary predicates) are subsets of \( \Delta \times \Delta \) (i.e., \( \Delta^2 \))
- Ternary predicates are subsets of \( \Delta^3 \) etc.

The sub-class relationship between classes can be interpreted as set inclusion.

Formally, the vocabulary is the set of names we use in our model of (part of) the world
- \{Daisy, Cow, Animal, Mary, Person, Z123ABC, Car, drives, ...\}

An interpretation \( \mathcal{I} \) is a tuple \( \langle \Delta, \mathcal{I} \rangle \)
- \( \Delta \) is the domain (a set)
- \( \mathcal{I} \) is a mapping that maps
  - Names of objects to elements of \( \Delta \)
  - Names of unary predicates (classes/concepts) to subsets of \( \Delta \)
  - Names of binary predicates (properties/roles) to subsets of \( \Delta \times \Delta \)
  - And so on for higher arity predicates (if any)
DL Knowledge Base

A DL Knowledge Base (KB) normally separated into 2 parts:

- **TBox** is a set of axioms describing structure of domain (i.e., a conceptual schema), e.g.:
  - HappyFather = Man $\land \exists$ has-child Female $\land$ ...
  - Elephant = Animal $\land$ Large $\land$ Grey
  - transitive(ancestor)

- **ABox** is a set of axioms describing a concrete situation (data), e.g.:
  - John:HappyFather
  - <John, Mary>: has-child

Separation has no logical significance
- But may be conceptually and implementationally convenient

DL Semantics

Interpretation function $I$ extends to concept expressions in the obvious way, i.e.:

$$(C \cap D)^I = C^I \cap D^I$$
$$(C \cup D)^I = C^I \cup D^I$$
$$(\neg C)^I = \Delta^I \setminus C^I$$
$$(x)^I = \{x^I\}$$
$$(\exists R.C)^I = \{x \mid \exists y. \langle x, y \rangle \in R^I \land y \in C^I\}$$
$$(\forall R.C)^I = \{x \mid \forall y. \langle x, y \rangle \in R^I \Rightarrow y \in C^I\}$$
$$(\leq n R)^I = \{x \mid \# \{y \mid \langle x, y \rangle \in R^I\} \leq n\}$$
$$(\geq n R)^I = \{x \mid \# \{y \mid \langle x, y \rangle \in R^I\} \geq n\}$$

Two sorts of Tbox axioms often distinguished
- "Definitions"
  - $C \subseteq D$ or $C \equiv D$ where $C$ is a concept name
- General Concept Inclusion axioms (GCIs)
  - $C \subseteq D$ where $C$ is an arbitrary concept
Knowledge Base Semantics

An interpretation $I$ satisfies (models) an axiom $A$ ($I \models A$):
- $I \models C \subseteq D$ iff $C^I \subseteq D^I$
- $I \models C \equiv D$ iff $C^I = D^I$
- $I \models R \subseteq S$ iff $R^I \subseteq S^I$
- $I \models R \equiv S$ iff $R^I = S^I$
- $I \models \exists R.C$ iff $(x,R)^I \in R^I$
- $I \models \forall R.C$ iff $(x,R)^I \in R^I$

$I$ satisfies a Tbox $T$ ($I \models T$) iff $I$ satisfies every axiom $A$ in $T$

$I$ satisfies an Abox $A$ ($I \models A$) iff $I$ satisfies every axiom $A$ in $A$

$I$ satisfies an KB $K$ ($I \models K$) iff $I$ satisfies both $T$ and $A$

Inference Tasks

Knowledge is correct (captures intuitions)
- $C$ subsumes $D$ w.r.t. $K$ iff for every model $I$ of $K$, $C^I \subseteq D^I$

Knowledge is minimally redundant (no unintended synonyms)
- $C$ is equivalent to $D$ w.r.t. $K$ iff for every model $I$ of $K$, $C^I = D^I$

Knowledge is meaningful (classes can have instances)
- $C$ is satisfiable w.r.t. $K$ iff there exists some model $I$ of $K$ s.t. $C^I \neq \emptyset$

Querying knowledge
- $x$ is an instance of $C$ w.r.t. $K$ iff for every model $I$ of $K$, $x^I \in C^I$
- $(x,y)$ is an instance of $R$ w.r.t. $K$ iff for every model $I$ of $K$, $(x,R,y)^I \in R^I$

Knowledge base consistency
- A KB $K$ is consistent iff there exists some model $I$ of $K$.

Syntax für DLs (ohne concrete domains)

<table>
<thead>
<tr>
<th>Concepts</th>
<th>ALC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>$A, B$</td>
</tr>
<tr>
<td>Not</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>And</td>
<td>$C \land D$</td>
</tr>
<tr>
<td>Or</td>
<td>$C \lor D$</td>
</tr>
<tr>
<td>Exists</td>
<td>$\exists R.C$</td>
</tr>
<tr>
<td>For all</td>
<td>$\forall R.C$</td>
</tr>
<tr>
<td>At least</td>
<td>$\geq 2n R.C$</td>
</tr>
<tr>
<td>At most</td>
<td>$\leq 5n R.C$</td>
</tr>
<tr>
<td>Nominal</td>
<td>${1, \ldots, n}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roles</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>$R$</td>
</tr>
<tr>
<td>Inverse</td>
<td>$R^-$</td>
</tr>
</tbody>
</table>

S = ALC + Transitivity

OWL DL = SHOIN(D) (D: concrete domain)

The Description Logic $\mathcal{ALC}$: Syntax

Atomic types: concept names $A, B, \ldots$ (unary predicates)
role names $R, S, \ldots$ (binary predicates)

Constructors:
- $\neg C$ (negation)
- $C \land D$ (conjunction)
- $C \lor D$ (disjunction)
- $\exists R.C$ (existential restriction)
- $\forall R.C$ (value restriction)

Implications:
- $C \rightarrow D = \neg C \lor D$ (implication)
- $C \leftrightarrow D = C \rightarrow D \land D \rightarrow C$ (bi-implication)

Abbreviations:
- $\top = A \lor \neg A$ (top concept)
- $\bot = A \land \neg A$ (bottom concept)
Examples

- Person ∩ Female
- Person ∩ ∃attends.Course
- Person ∩ ∀attends.(Course → ¬Easy)
- Person ∩ ∃teaches.(Course ∩ ∀attended-by.(Bored ∪ Sleeping))

Interpretations

Semantics based on interpretations $(Δ^I, ⦃⋅⦄^I)$, where
- $Δ^I$ is a non-empty set (the domain)
- ⦃⋅⦄^I is the interpretation function mapping
  each concept name $A$ to a subset $A^I$ of $Δ^I$ and
  each role name $R$ to a binary relation $R^I$ over $Δ^I$.

Intuition: interpretation is complete description of the world

Technically: interpretation is first-order structure
  with only unary and binary predicates

Semantics of Complex Concepts

$(¬C)^I = Δ^I \setminus C^I$  
$(C ∩ D)^I = C^I \cap D^I$  
$(C ∪ D)^I = C^I \cup D^I$

$(∃R.C)^I = \{d |$ there is an $e ∈ Δ^I$ with $(d, e) ∈ R^I$ and $e ∈ C^I\}$

$(∀R.C)^I = \{d |$ for all $e ∈ Δ^I$, $(d, e) ∈ R^I$ implies $e ∈ C^I\}$
**TBoxes**

Capture an application’s terminology means defining concepts

**TBoxes** are used to store concept definitions:

**Syntax:**
- finite set of concept equations $A \triangleq C$
- with $A$ concept name and $C$ concept
- left-hand sides must be unique!

**Semantics:**
- interpretation $\mathcal{I}$ satisfies $A \triangleq C$ if $A^\mathcal{I} = C^\mathcal{I}$
- $\mathcal{I}$ is model of $\mathcal{T}$ if it satisfies all definitions in $\mathcal{T}$

E.g.: Lecturer $\triangleq$ Person $\sqcap \exists$teaches.Course

Yields two kinds of concept names: defined and primitive

**TBox: Example**

**TBoxes** are used as ontologies:

Woman $\triangleq$ Person $\sqcap$ Female

Man $\triangleq$ Person $\sqcap \neg$Woman

Lecturer $\triangleq$ Person $\sqcap \exists$teaches.Course

Student $\triangleq$ Person $\sqcap \exists$attends.Course

BadLecturer $\triangleq$ Person $\sqcap \forall$teaches.(Course $\rightarrow$ Boring)

**TBox: Example II**

A TBox restricts the set of admissible interpretations.

Lecturer $\triangleq$ Person $\sqcap \exists$teaches.Course

Student $\triangleq$ Person $\sqcap \exists$attends.Course

Reasoning Tasks — Subsumption

$C$ subsumed by $D$ w.r.t. $\mathcal{T}$ (written $C \sqsubseteq^\mathcal{T} D$)

iff

$C^\mathcal{I} \subseteq D^\mathcal{I}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$

**Intuition:** If $C \sqsubseteq^\mathcal{T} D$, then $D$ is more general than $C$

**Example:**

Lecturer $\triangleq$ Person $\sqcap \exists$teaches.Course

Student $\triangleq$ Person $\sqcap \exists$attends.Course

Then

Lecturer $\sqcap \exists$attends.Course $\sqsubseteq^\mathcal{T}$ Student
**Reasoning Tasks — Classification**

**Classification**: arrange all defined concepts from a TBox in a hierarchy w.r.t. generality.

\[
\begin{align*}
\text{Woman} & \equiv \text{Person} \sqcap \neg \text{Female} \\
\text{Man} & \equiv \text{Person} \sqcap \neg \text{Woman} \\
\text{MaleLecturer} & \equiv \text{Man} \sqcap \exists \text{teachesCourse}
\end{align*}
\]

Can be computed using multiple subsumption tests.

Provides a principled view on ontology for browsing, maintaining, etc.

---

**Reasoning Tasks — Satisfiability**

**Intuition**: If unsatisfiable, the concept contains a contradiction.

**Example**: \[ \text{Woman} \equiv \text{Person} \sqcap \neg \text{Female} \]

\[ \text{Man} \equiv \text{Person} \sqcap \neg \text{Woman} \]

Then \[ \exists \text{Sibling.}\text{Man} \sqcap \neg \text{Sibling.}\text{Woman} \] is unsatisfiable w.r.t. \( \mathcal{T} \).

Subsumption can be reduced to (un)satisfiability and vice versa:

- \( C \sqsubseteq_T D \) iff \( C \sqcap \neg D \) is not satisfiable w.r.t. \( \mathcal{T} \)

- \( C \) is satisfiable w.r.t. \( \mathcal{T} \) if not \( C \sqsubseteq_T \bot \).

Many reasoners decide satisfiability rather than subsumption.

---

**A Concept Hierarchy**

Excerpt from a process engineering ontology.

---

**Reasoning Tasks — Classification**

**Classification**: arrange all defined concepts from a TBox in a hierarchy w.r.t. generality.

\[
\begin{align*}
\text{Woman} & \equiv \text{Person} \sqcap \neg \text{Female} \\
\text{Man} & \equiv \text{Person} \sqcap \neg \text{Woman} \\
\text{MaleLecturer} & \equiv \text{Man} \sqcap \exists \text{teachesCourse}
\end{align*}
\]

Can be computed using multiple subsumption tests.

Provides a principled view on ontology for browsing, maintaining, etc.

---

**Reasoning Tasks — Satisfiability**

**Intuition**: If unsatisfiable, the concept contains a contradiction.

**Example**: \[ \text{Woman} \equiv \text{Person} \sqcap \neg \text{Female} \]

\[ \text{Man} \equiv \text{Person} \sqcap \neg \text{Woman} \]

Then \[ \exists \text{Sibling.}\text{Man} \sqcap \neg \text{Sibling.}\text{Woman} \] is unsatisfiable w.r.t. \( \mathcal{T} \).

Subsumption can be reduced to (un)satisfiability and vice versa:

- \( C \sqsubseteq_T D \) iff \( C \sqcap \neg D \) is not satisfiable w.r.t. \( \mathcal{T} \)

- \( C \) is satisfiable w.r.t. \( \mathcal{T} \) if not \( C \sqsubseteq_T \bot \).

Many reasoners decide satisfiability rather than subsumption.
**Definitorial TBoxes**

A **primitive interpretation** for TBox $\mathcal{T}$ interpretes
- the **primitive** concept names in $\mathcal{T}$
- all role names

A TBox is called **definitorial** if every primitive interpretation for $\mathcal{T}$ can be **uniquely** extended to a model of $\mathcal{T}$.

i.e.: primitive concepts (and roles) uniquely determine defined concepts

Not all TBoxes are definitorial:

$$\text{Person} \triangleq \exists \text{parent}. \text{Person}$$

Non-definitorial TBoxes describe **constraints**, e.g. from **background knowledge**

---

**Acyclic TBoxes**

TBox $\mathcal{T}$ is **acyclic** if there are no definitorial cycles:

$$\text{Lecturer} \triangleq \text{Person} \sqcap \exists \text{teaches}. \text{Course}$$

$$\text{Course} \triangleq \exists \text{has-title}. \text{Title} \sqcap \exists \text{taught-by}. \text{Lecturer}$$

**Expansion of acyclic TBox $\mathcal{T}$:**

exhaustively replace defined concept names with their definition
(terminates due to acyclicity)

Acyclic TBoxes are always definitorial:

first expand, then set $A^\mathcal{T} \triangleq C^\mathcal{T}$ for all $A \triangleq C \in \mathcal{T}$

---

**Acyclic TBoxes II**

For reasoning, acyclic TBox can be eliminated
- to decide $C \sqsubseteq_D D$ with $\mathcal{T}$ acyclic,
  - expand $\mathcal{T}$
  - replace defined concept names in $C, D$ with their definition
  - decide $C \sqsubseteq D$
- analogously for satisfiability

May yield an **exponential blow-up**:

$$A_0 \triangleq \forall r. A_1 \sqcap \forall s. A_1$$

$$A_1 \triangleq \forall r. A_2 \sqcap \forall s. A_2$$

$$\ldots$$

$$A_{n-1} \triangleq \forall r. A_n \sqcap \forall s. A_n$$

---

**General Concept Inclusions**

View of TBox as **set of constraints**

**General TBox:** finite set of general concept implications (GCIs)

$C \sqsubseteq D$

with both $C$ and $D$ allowed to be complex

e.g. Course $\sqcap \forall \text{attended-by}. \text{Sleeping} \sqsubseteq \text{Boring}$

Interpretation $\mathcal{I}$ is **model** of general TBox $\mathcal{T}$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$ for all $C \sqsubseteq D \in \mathcal{T}$.

$C \triangleq D$ is abbreviation for $C \sqsubseteq D, D \sqsubseteq C$

e.g. Student $\sqcap \exists \text{has-favourite}. \text{SoccerTeam} \triangleq \text{Student} \sqcap \exists \text{has-favourite}. \text{Beer}$

Note: $C \sqsubseteq D$ equivalent to $\top \triangleq C \rightarrow D$
ABoxes

ABoxes describe a snapshot of the world

An ABox is a finite set of assertions

\[ \alpha : C \quad [\text{\(\alpha\) individual name, \(C\) concept}] \]
\[ (\alpha, \beta) : R \quad [\text{\(\alpha, \beta\) individual names, \(R\) role name}] \]

E.g. \{peter : Student, (dl-course, uli) : taught-by\}

Interpretations \(I\) map each individual name \(\alpha\) to an element of \(\Delta^I\).

\(I\) satisfies an assertion

\[ \alpha : C \quad \text{iff} \quad \alpha^I \in C^I \]
\[ (\alpha, \beta) : R \quad \text{iff} \quad (\alpha^I, \beta^I) \in R^I \]

\(I\) is a model for an ABox \(\mathcal{A}\) if \(I\) satisfies all assertions in \(\mathcal{A}\).

ABox consistency

Given an ABox \(\mathcal{A}\) and a TBox \(\mathcal{T}\), do they have a common model?

Instance checking

Given an ABox \(\mathcal{A}\), a TBox \(\mathcal{T}\), an individual name \(\alpha\), and a concept \(C\)

\[ \alpha^I \in C^I \quad \text{hold in all models of} \quad \mathcal{A} \quad \text{and} \quad \mathcal{T} \quad \text{?} \quad \text{(written} \mathcal{A}, \mathcal{T} \models \alpha : C) \]

The two tasks are interreducible:

\[ \mathcal{A} \text{ consistent w.r.t.} \mathcal{T} \text{ iff } \mathcal{A}, \mathcal{T} \not\models \alpha : \bot \]
\[ \mathcal{A}, \mathcal{T} \models \alpha : C \text{ iff } \mathcal{A} \cup \{ \alpha : \neg C \} \text{ is not consistent} \]

Example for ABox Reasoning

ABox

\[ \text{dumbo} : \text{Mammal} \quad \text{t14} : \text{Trunk} \]
\[ \text{g23} : \text{Darkgrey} \quad (\text{dumbo}, \text{t14}) : \text{bodypart} \]
\[ (\text{dumbo}, \text{g23}) : \text{color} \]
\[ \text{dumbo} : \forall \text{color.Lightgrey} \]

TBox

\[ \text{Elephant} \models \text{Mammal} \land \exists \text{bodypart}.\text{Trunk} \land \forall \text{color.Grey} \]
\[ \text{Grey} \models \text{Lightgrey} \lor \text{Darkgrey} \]
\[ \bot \models \text{Lightgrey} \land \text{Darkgrey} \]

1. ABox is inconsistent w.r.t. TBox.
2. dumbo is an instance of Elephant

Note:

- interpretations describe the state if the world in a complete way
- ABoxes describe the state if the world in an incomplete way

(uli, dl-course) : taught-by  uli : Female

does not imply

dl-course : $\forall$tought-by.Female

An ABox has many models!

An ABox constraints the set of admissible models similar to a TBox
2. Tableau algorithms for $\mathcal{ALC}$ and extensions

We see a tableau algorithm for $\mathcal{ALC}$ and extend it with

1. general TBoxes and
2. inverse roles

Goal: Design sound and complete decision procedures for satisfiability (and subsumption) of DLs which are well-suited for implementation purposes

---

A tableau algorithm for the satisfiability of $\mathcal{ALC}$ concepts

Goal: design an algorithm which takes an $\mathcal{ALC}$ concept $C_0$ and

1. returns “satisfiable” iff $C_0$ is satisfiable and
2. terminates, on every input,

i.e., which decides satisfiability of $\mathcal{ALC}$ concepts.

Recall: such an algorithm cannot exist for FOL since
satisfiability of FOL is undecidable.

Idea: our algorithm

• is tableau-based and
• tries to construct a model of $C_0$
• by breaking $C_0$ down syntactically, thus
• inferring new constraints on such a model.

---

Preliminaries: Negation Normal Form

To make our life easier, we transform each concept $C_0$ into an equivalent $C_1$ in NNF

Equivalent: $C_0 \sqsubseteq C_1$ and $C_1 \sqsubseteq C_0$

NNF: negation occurs only in front of concept names

How? By pushing negation inwards (de Morgan et. al):

\[ \neg(C \sqcup D) \Rightarrow \neg C \sqcup \neg D \]
\[ \neg(C \sqcap D) \Rightarrow \neg C \sqcap \neg D \]
\[ \neg \neg C \Rightarrow C \]
\[ \neg \forall R.C \Rightarrow \exists R.\neg C \]
\[ \neg \exists R.C \Rightarrow \forall R.\neg C \]

From now on: concepts are in NNF and

$\text{sub}(C)$ denotes the set of all sub-concepts of $C$

---

More intuition

Find out whether $A \sqcap \exists R.B \sqcap \forall R.\neg B$ is satisfiable...

$A \sqcap \exists R.B \sqcap \forall R.\neg B \sqcup \exists S.E$

Our tableau algorithm works on a completion tree which

• represents a model $I$: nodes represent elements of $\Delta^I$

  $\Rightarrow$ each node $x$ is labelled with concepts $L(x) \subseteq \text{sub}(C_0)$

  $C \in L(x)$ is read as “$x$ should be an instance of $C$”

  edges represent role successorship

  $\Rightarrow$ each edge $(x, y)$ is labelled with a role-name from $C_0$

  $R \in L((x, y))$ is read as “$(x, y)$ should be in $R^I$”

• is initialised with a single root node $x_0$ with $L(x_0) = \{C_0\}$

• is expanded using completion rules
Completion rules for $\mathcal{ALC}$

- $\Box$-rule: if $C_1 \cap C_2 \in L(x)$ and $\{C_1, C_2\} \not\subseteq L(x)$
  then set $L(x) = L(x) \cup \{C_1, C_2\}$

- $\top$-rule: if $C_1 \cup C_2 \in L(x)$ and $\{C_1, C_2\} \cap L(x) = \emptyset$
  then set $L(x) = L(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

$\exists$-rule: if $\exists S.C \in L(x)$ and $x$ has no $S$-successor $y$ with $C \in L(y)$,
then create a new node $y$ with $L((x, y)) = \{S\}$ and $L(y) = \{C\}$

$\forall$-rule: if $\forall S.C \in L(x)$ and there is an $S$-successor $y$ of $x$ with $C \not\in L(y)$
then set $L(y) = L(y) \cup \{C\}$

Properties of the completion rules for $\mathcal{ALC}$

We only apply rules if their application does "something new"

- $\Box$-rule: if $C_1 \cap C_2 \in L(x)$ and $\{C_1, C_2\} \not\subseteq L(x)$
  then set $L(x) = L(x) \cup \{C_1, C_2\}$

- $\top$-rule: if $C_1 \cup C_2 \in L(x)$ and $\{C_1, C_2\} \cap L(x) = \emptyset$
  then set $L(x) = L(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

$\exists$-rule: if $\exists S.C \in L(x)$ and $x$ has no $S$-successor $y$ with $C \in L(y)$,
then create a new node $y$ with $L((x, y)) = \{S\}$ and $L(y) = \{C\}$

$\forall$-rule: if $\forall S.C \in L(x)$ and there is an $S$-successor $y$ of $x$ with $C \not\in L(y)$
then set $L(y) = L(y) \cup \{C\}$

Properties of the completion rules for $\mathcal{ALC}$

The $\top$-rule is non-deterministic:

- $\Box$-rule: if $C_1 \cap C_2 \in L(x)$ and $\{C_1, C_2\} \not\subseteq L(x)$
  then set $L(x) = L(x) \cup \{C_1, C_2\}$

- $\top$-rule: if $C_1 \cup C_2 \in L(x)$ and $\{C_1, C_2\} \cap L(x) = \emptyset$
  then set $L(x) = L(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

$\exists$-rule: if $\exists S.C \in L(x)$ and $x$ has no $S$-successor $y$ with $C \in L(y)$,
then create a new node $y$ with $L((x, y)) = \{S\}$ and $L(y) = \{C\}$

$\forall$-rule: if $\forall S.C \in L(x)$ and there is an $S$-successor $y$ of $x$ with $C \not\in L(y)$
then set $L(y) = L(y) \cup \{C\}$

Last details on tableau algorithm for $\mathcal{ALC}$

Clash: a c-tree contains a clash if it has a node $x$ with $\bot \in L(x)$ or
$\{A, \neg A\} \subseteq L(x)$ — otherwise, it is clash-free

Complete: a c-tree is complete if none of the completion rules can be applied to it

Answer behaviour: when started for $C_0$ (in NNF!), the tableau algorithm

- is initialised with a single root node $x_0$ with $L(x_0) = \{C_0\}$
- repeatedly applies the completion rules (in whatever order it likes)
- answer "$C_0$ is satisfiable" iff the completion rules can be applied in such a way that it results in a complete and clash-free c-tree (careful: this is non-deterministic)

...go back to examples
Properties of our tableau algorithm

**Lemma:** Let $C_0$ an $\mathcal{ALC}$-concept in NNF. Then
1. the algorithm terminates when applied to $C_0$ and
2. the rules can be applied such that they generate a clash-free and complete completion tree iff $C_0$ is satisfiable.

**Corollary:**
1. Our tableau algorithm decides satisfiability and subsumption of $\mathcal{ALC}$.
2. Satisfiability (and subsumption) in $\mathcal{ALC}$ is decidable in PSpace.
3. $\mathcal{ALC}$ has the finite model property
   i.e., every satisfiable concept has a finite model.
4. $\mathcal{ALC}$ has the tree model property
   i.e., every satisfiable concept has a tree model.
5. $\mathcal{ALC}$ has the finite tree model property
   i.e., every satisfiable concept has a finite tree model.

Extend tableau algorithm to $\mathcal{ALC}$ with general TBoxes: Preliminaries

We extend our tableau algorithm by adding a new completion rule:
- remember that nodes represent elements of $\Delta^T$ and
- if $C \sqsubseteq D \in T$, then for each element $x$ in a model $\mathcal{I}$ of $T$
  if $x \in C^\mathcal{I}$, then $x \in D^\mathcal{I}$
  hence $x \in (\neg C)^\mathcal{I}$ or $x \in D^\mathcal{I}$
  $x \in (\neg C \sqcup D)^\mathcal{I}$
  $x \in (\text{NNF}(\neg C \sqcup D))^\mathcal{I}$
  for $\text{NNF}(E)$ the negation normal form of $E$

Extend tableau algorithm to $\mathcal{ALC}$ with general TBoxes

**Recall:**
- Concept inclusion: of the form $C \sqsubseteq D$ for $C, D$ (complex) concepts
- (General) TBox: a finite set of concept inclusions
- $\mathcal{I}$ satisfies $C \sqsubseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$
- $\mathcal{I}$ is a model of TBox $T$ iff $\mathcal{I}$ satisfies each concept equation in $T$
- $C_0$ is satisfiable w.r.t. $T$ iff there is a model $\mathcal{I}$ of $T$ with $C_0^\mathcal{I} \neq \emptyset$

**Goal – Lemma:** Let $C_0$ an $\mathcal{ALC}$-concept and $T$ be a an $\mathcal{ALC}$-TBox. Then
1. the algorithm terminates when applied to $T$ and $C_0$ and
2. the rules can be applied such that they generate a clash-free and complete completion tree if $C_0$ is satisfiable w.r.t. $T$.

Completion rules for $\mathcal{ALC}$ with TBoxes

\begin{align*}
\sqcap\text{-rule: } & \quad C_1 \cap C_2 \in \mathcal{L}(x) \text{ and } \{C_1, C_2\} \not\subseteq \mathcal{L}(x) \quad \text{then set } \mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\} \\
\sqcup\text{-rule: } & \quad C_1 \sqcup C_2 \in \mathcal{L}(x) \text{ and } \{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset \quad \text{then set } \mathcal{L}(x) = \mathcal{L}(x) \cup \{C\} \text{ for some } C \in \{C_1, C_2\} \\
\exists\text{-rule: } & \quad \exists S.C \in \mathcal{L}(x) \text{ and } x \text{ has no } S\text{-successor } y \text{ with } C \in \mathcal{L}(y), \quad \text{then create a new node } y \text{ with } \mathcal{L}((x, y)) = \{S\} \text{ and } \mathcal{L}(y) = \{C\} \\
\forall\text{-rule: } & \quad \forall S.C \in \mathcal{L}(x) \text{ and there is an } S\text{-successor } y \text{ of } x \text{ with } C \notin \mathcal{L}(y), \quad \text{then set } \mathcal{L}(y) = \mathcal{L}(y) \cup \{C\} \\
T\text{-rule: } & \quad C_1 \sqsubseteq C_2 \in T \text{ and } \text{NNF}(\neg C_1 \sqcup C_2) \notin \mathcal{L}(x) \quad \text{then set } \mathcal{L}(x) = \mathcal{L}(x) \cup \{\text{NNF}(\neg C_1 \sqcup C_2)\}
\end{align*}
A tableau algorithm for $\mathcal{ALC}$ with general TBoxes

Example: Consider satisfiability of $C$ w.r.t. $\{C \sqsubseteq \exists R.C\}$

Tableau algorithm no longer terminates!

Reason: size of concepts no longer decreases along paths in a completion tree

Observation: most nodes on this path look the same and we keep repeating ourselves

Regain termination with a “cycle-detection” technique called blocking

Intuitively, whenever we find a situation where $y$ has to satisfy stronger constraints than $x$, we freeze $x$, i.e., block rules from being applied to $x$.

A tableau algorithm for $\mathcal{ALC}$ with general TBoxes: Blocking

- $x$ is directly blocked if it has an ancestor $y$ with $L(x) \subseteq L(y)$
- in this case and if $y$ is the “closest” such node to $x$, we say that $x$ is blocked by $y$
- a node is blocked if it is directly blocked or one of its ancestors is blocked

$\oplus$ restrict the application of all rules to nodes which are not blocked

$\rightarrow$ completion rules for $\mathcal{ALC}$ w.r.t. TBoxes

Tableaux Rules for $\mathcal{ALC}$

<table>
<thead>
<tr>
<th>Rule</th>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqcap$-rule:</td>
<td>$x \cdot {C_1 \sqcap C_2, \ldots}$</td>
<td>$\rightarrow \neg$</td>
</tr>
<tr>
<td></td>
<td>$x \cdot {C_1 \sqcup C_2, \ldots}$</td>
<td>$\rightarrow \sqcup$</td>
</tr>
<tr>
<td>$\exists$-rule:</td>
<td>$x \cdot {\exists R.C, \ldots}$</td>
<td>$\rightarrow \exists$</td>
</tr>
<tr>
<td></td>
<td>$y \cdot {\ldots}$</td>
<td>$\rightarrow \forall$</td>
</tr>
<tr>
<td>$\forall$-rule:</td>
<td>$x \cdot {\forall R.C, \ldots}$</td>
<td>$\rightarrow \forall$</td>
</tr>
</tbody>
</table>
**Tableaux Rule for Transitive Roles**

\[
\begin{array}{c|c}
\{ \forall R.C, \ldots \} & \rightarrow_{\forall R.} \\
R & \{ \forall R.C, \ldots \}
\end{array}
\]

Where \( R \) is a transitive role (i.e., \( (R^+) = R^T \))

☞ No longer naturally terminating (e.g., if \( C = \exists R. \top \))

☞ Need blocking

- Simple blocking suffices for \( ALC \) plus transitive roles
- I.e., do not expand node label if ancestor has superset label
- More expressive logics (e.g., with inverse roles) need more sophisticated blocking strategies

**Tableaux Algorithm — Example**

Test satisfiability of \( \exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C) \) where \( R \) is a transitive role

\[ \mathcal{L}(w) = \{ \exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C) \} \]
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S.(\neg C \sqcup \neg D) \land \exists R.C \land \forall R.(\exists R.C)$ where $R$ is a transitive role

$\mathcal{L}(w) = \{\exists S.C, \forall S.(-C \sqcup -D), \exists R.C, \forall R.(\exists R.C)\}$

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S.(\neg C \sqcup \neg D) \land \exists R.C \land \forall R.(\exists R.C)$ where $R$ is a transitive role

$\mathcal{L}(w) = \{\exists S.C, \forall S.(-C \sqcup -D), \exists R.C, \forall R.(\exists R.C)\}$

$\mathcal{L}(x) = \{C\}$

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S.(\neg C \sqcup \neg D) \land \exists R.C \land \forall R.(\exists R.C)$ where $R$ is a transitive role

$\mathcal{L}(w) = \{\exists S.C, \forall S.(-C \sqcup -D), \exists R.C, \forall R.(\exists R.C)\}$

$\mathcal{L}(x) = \{C\}$

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S.(\neg C \sqcup \neg D) \land \exists R.C \land \forall R.(\exists R.C)$ where $R$ is a transitive role

$\mathcal{L}(w) = \{\exists S.C, \forall S.(-C \sqcup -D), \exists R.C, \forall R.(\exists R.C)\}$

$\mathcal{L}(x) = \{C\}$
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S. \neg (C \sqcup \neg D) \land \exists R.C \land \forall R. (\exists R.C)$ where $R$ is a transitive role.

$L(w) = \{ \exists S.C, \forall S. (\neg C \sqcup \neg D), \exists R.C, \forall R. (\exists R.C) \}$

$L(x) = \{ C, \neg C \sqcup \neg D \}$

$L(w) = \{ \exists S.C, \forall S. (\neg C \sqcup \neg D), \exists R.C, \forall R. (\exists R.C) \}$

$L(x) = \{ C, (\neg C \sqcup \neg D), \neg C \}$

Reasoning with Expressive Description Logics – p. 7/27
Test satisfiability of $\exists S.C \land \forall S. (\neg C \sqcup \neg D) \land \exists R.C \land \forall R. (\exists R.C)$} where $R$ is a transitive role

\[ \mathcal{L}(w) = \{ \exists S.C, \forall S. (\neg C \sqcup \neg D), \exists R.C, \forall R. (\exists R.C) \} \]

\[ \mathcal{L}(x) = \{ C, \neg C \sqcup \neg D \} \]

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S. (\neg C \sqcup \neg D) \land \exists R.C \land \forall R. (\exists R.C)$} where $R$ is a transitive role

\[ \mathcal{L}(w) = \{ \exists S.C, \forall S. (\neg C \sqcup \neg D), \exists R.C, \forall R. (\exists R.C) \} \]

\[ \mathcal{L}(x) = \{ C, \neg C \sqcup \neg D \}, \neg D \} \]

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S. (\neg C \sqcup \neg D) \land \exists R.C \land \forall R. (\exists R.C)$} where $R$ is a transitive role

\[ \mathcal{L}(w) = \{ \exists S.C, \forall S. (\neg C \sqcup \neg D), \exists R.C, \forall R. (\exists R.C) \} \]

\[ \mathcal{L}(x) = \{ C, \neg C \sqcup \neg D \}, R.C \}

\[ \mathcal{L}(y) = \{ C \} \]
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S.\neg(C \sqcup D) \land \exists R.C \land \forall R.(\exists R.C)$ where $R$ is a transitive role.

$L(w) = \{\exists S.C, \forall S.\neg(C \sqcup D), \exists R.C, \forall R.(\exists R.C)\}$

$L(x) = \{C, (\neg C \sqcup \neg D), \neg D\}$

$L(y) = \{C\}$
Tableaux Algorithm — Example

Test satisfiability of $\exists S. C \land \forall S. (\neg C \sqcup \neg D) \land \exists R. C \land \forall R. (\exists R. C)$ where $R$ is a transitive role

$L(w) = \{\exists S. C, \forall S. (\neg C \sqcup \neg D), \exists R. C, \forall R. (\exists R. C)\}$

$L(x) = \{C, (\neg C \sqcup \neg D), \neg D\}$

$L(y) = \{C, \exists R. C, \forall R. (\exists R. C)\}$

$L(z) = \{C\}$

Tableaux Algorithm — Example

Test satisfiability of $\exists S. C \land \forall S. (\neg C \sqcup \neg D) \land \exists R. C \land \forall R. (\exists R. C)$ where $R$ is a transitive role

$L(w) = \{\exists S. C, \forall S. (\neg C \sqcup \neg D), \exists R. C, \forall R. (\exists R. C)\}$

$L(x) = \{C, (\neg C \sqcup \neg D), \neg D\}$

$L(y) = \{C, \exists R. C, \forall R. (\exists R. C)\}$

$L(z) = \{C\}$

Concept is satisfiable: $T$ corresponds to model
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \land \forall S. (\neg C \sqcup \neg D) \land \exists R.C \land \forall R.(\exists R.C)$} where $R$ is a transitive role

Concept is **satisfiable**: $T$ corresponds to model

Properties of our tableau algorithm for $\mathcal{ALC}$ with TBoxes

**Lemma**: Let $T$ be a general $\mathcal{ALC}$-Tbox and $C_0$ an $\mathcal{ALC}$-concept. Then

1. the algorithm terminates when applied to $T$ and $C_0$ and
2. the rules can be applied such that they generate a clash-free and complete completion tree iff $C_0$ is satisfiable w.r.t. $T$.

**Corollary**: 1. Satisfiability of $\mathcal{ALC}$-concept w.r.t. TBoxes is decidable
2. $\mathcal{ALC}$ with TBoxes has the finite model property
3. $\mathcal{ALC}$ with TBoxes has the tree model property

Challenges

☞ **Increased expressive power**
  - Existing DL systems implement (at most) $SHIQ$
  - OWL extends $SHIQ$ with datatypes and nominals

☞ **Scalability**
  - Very large KBs
  - Reasoning with (very large numbers of) individuals

☞ **Other reasoning tasks**
  - Querying
  - Matching
  - Least common subsumer
  - ... 

☞ **Tools and Infrastructure**
  - Support for large scale ontological engineering and deployment
Summary

☞ **Description Logics** are family of logical KR formalisms

☞ **Applications** of DLs include DataBases and **Semantic Web**
  - Ontologies will provide vocabulary for semantic markup
  - OWL web ontology language based on $SHIQ$ DL
  - Set to become W3C standard (OWL) & already widely adopted
  - Use of DL provides formal foundations and reasoning support

☞ **DL Reasoning** based on tableau algorithms

☞ **Highly Optimised** implementations used in DL systems

☞ **Challenges** remain
  - Reasoning with full OWL language
  - (Convincing) demonstration(s) of scalability
  - New reasoning tasks
  - Development of (high quality) tools and infrastructure

---

Resources

Slides from this talk

FaCT system (open source)
  - [http://www.cs.man.ac.uk/FaCT/](http://www.cs.man.ac.uk/FaCT/)

OIlEd (open source)
  - [http://oiled.man.ac.uk/](http://oiled.man.ac.uk/)

OIL
  - [http://www.ontoknowledge.org/oil/](http://www.ontoknowledge.org/oil/)

W3C Web-Ontology (WebOnt) working group (OWL)
  - [http://www.w3.org/2001/sw/WebOnt/](http://www.w3.org/2001/sw/WebOnt/)

**DL Handbook**, Cambridge University Press
  - [http://books.cambridge.org/0521781760.htm](http://books.cambridge.org/0521781760.htm)