Beschreibungslogiken (Description Logics)

- sind eine Familie von logik-basierten Wissensrepräsentationssprachen
- stammen von semantischen Netzen und KL-ONE ab.
- beschreiben die Welt mit Konzepten (Klassen), Rollen (Relationen) und Individuen.

- haben eine formale (typischerweise modell-theoretische) Semantik.
  - Sie sind entscheidbare Fragmente der PL1
  - und eng verwandt mit aussagenlogischen Modal- und Temporallogiken.
- bieten Inferenzmechanismen für zentrale Probleme.
  - Korrekte und vollständige Entscheidungsverfahren existieren.
  - Hoch-effiziente Implementierungen existieren.

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

**Literatur**


**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.
What is the Problem?

Consider a typical web page:

Markup consists of:
- rendering information (e.g., font size and colour)
- Hyper-links to related content

Semantic content is accessible to humans but not (easily) to computers...

What information can we see...

WWW2002
The eleventh international world wide web conference
Sheraton waikiki hotel
Honolulu, hawaii, USA
7-11 may 2002
1 location 5 days learn interact
Registered participants coming from
australia, canada, chile denmark, france, germany, ghana, hong kong, india, ireland, italy, japan, malta, new zealand, the netherlands, norway, singapore, switzerland, the united kingdom, the united states, vietnam, zaire
Register now
On the 7th May Honolulu will provide the backdrop of the eleventh international world wide web conference. This prestigious event ...
Speakers confirmed
Tim berners-lee
Tim is the well known inventor of the Web, ...
Ian Foster
Ian is the pioneer of the Grid, the next generation internet ...

Solution: XML markup with “meaningful” tags?

What information can a machine see...
Recall: Logics and Model Theory

But What About…

Recall: Logics and Model Theory

Machine sees…

Recall: Logics and Model Theory

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., $T \models A \subseteq B$ 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 referred 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.
Recall: Logics and Model Theory

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)

Model

World

Recall: Logics and Model Theory

\( \{a, b, \ldots\} \subseteq \Delta \times \Delta \)

Model

Interpretation

World

Recall: Logics and Model Theory

DL Architecture

DL Knowledge Base

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 \( \land \ldots \)
  - 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>:hasChild

Separation has no logical significance

- But may be conceptually and implementationally convenient

Knowledge Base

Tbox (schema)

Man = Human \( \land \) Male
Happy-Father = Man \( \land \exists \) has-child Female \( \land \ldots \)

Abox (data)

John : Happy-Father
<John, Mary> : has-child

Inference System

Interface

DL Architecture
Interpretation function $\mathcal{I}$ extends to concept expressions in the obvious way, i.e.:

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

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

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

$$\{x\}^\mathcal{I} = \{x^\mathcal{I}\}$$

$$(\exists R.C)^\mathcal{I} = \{x \mid \exists y.\langle x, y \rangle \in R^\mathcal{I} \land y \in C^\mathcal{I}\}$$

$$(\forall R.C)^\mathcal{I} = \{x \mid \forall y.\langle x, y \rangle \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I}\}$$

$$(\leq n R)^\mathcal{I} = \{x \mid \#\{y \mid \langle x, y \rangle \in R^\mathcal{I}\} \leq n\}$$

$$(\geq n R)^\mathcal{I} = \{x \mid \#\{y \mid \langle x, y \rangle \in R^\mathcal{I}\} \geq n\}$$

**DL Knowledge Bases (Ontologies)**

A DL Knowledge Base is of the form $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

- $\mathcal{T}$ (Tbox) is a set of axioms of the form:
  - $C \subseteq D$ (concept inclusion)
  - $C \equiv D$ (concept equivalence)
  - $R \subseteq S$ (role inclusion)
  - $R \equiv S$ (role equivalence)
  - $R^+ \subseteq R$ (role transitivity)

- $\mathcal{A}$ (Abox) is a set of axioms of the form
  - $x \in D$ (concept instantiation)
  - $\langle x, y \rangle \in R$ (role instantiation)

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 $\mathcal{I}$ satisfies (models) an axiom $A$ ($\mathcal{I} \models A$):

- $\mathcal{I} \models C \subseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$
- $\mathcal{I} \models C \equiv D$ if $C^\mathcal{I} = D^\mathcal{I}$
- $\mathcal{I} \models R \subseteq S$ if $R^\mathcal{I} \subseteq S^\mathcal{I}$
- $\mathcal{I} \models R^+ \subseteq R$ if $(R^+)^\mathcal{I} \subseteq R^\mathcal{I}$
- $\mathcal{I} \models x \in D$ if $x^\mathcal{I} \in D^\mathcal{I}$
- $\mathcal{I} \models \langle x, y \rangle \in R$ if $(x^\mathcal{I}, y^\mathcal{I}) \in R^\mathcal{I}$

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

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

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

**Inference Tasks**

Knowledge is correct (captures intuitions)

- $C$ subsumes D w.r.t. $\mathcal{K}$ iff for every model $\mathcal{I}$ of $\mathcal{K}$, $C^\mathcal{I} \subseteq D^\mathcal{I}$

Knowledge is minimally redundant (no unintended synonyms)

- $C$ is equivalent to D w.r.t. $\mathcal{K}$ iff for every model $\mathcal{I}$ of $\mathcal{K}$, $C^\mathcal{I} = D^\mathcal{I}$

Knowledge is meaningful (classes can have instances)

- $C$ is satisfiable w.r.t. $\mathcal{K}$ iff there exists some model $\mathcal{I}$ of $\mathcal{K}$ s.t. $C^\mathcal{I} \neq \emptyset$

Querying knowledge

- $x$ is an instance of $C$ w.r.t. $\mathcal{K}$ iff for every model $\mathcal{I}$ of $\mathcal{K}$, $x^\mathcal{I} \in C^\mathcal{I}$
- $\langle x, y \rangle$ is an instance of $R$ w.r.t. $\mathcal{K}$ iff for every model $\mathcal{I}$ of $\mathcal{K}$, $(x^\mathcal{I}, y^\mathcal{I}) \in R^\mathcal{I}$

Knowledge base consistency

- A KB $\mathcal{K}$ is consistent iff there exists some model $\mathcal{I}$ of $\mathcal{K}$. 
Syntax für DLs (ohne concrete domains)

**Concepts**
- Atomic: $A, B$
- Not: $\neg C$
- And: $C \land D$
- Or: $C \lor D$
- Exists: $\exists R.C$
- For all: $\forall R.C$
- At least: $\geq n R.C$ ($\geq n R$)
- At most: $\leq n R.C$ ($\leq n R$)
- Nominal: $(i_1, \ldots, i_n)$

**Roles**
- Atomic: $R$
- Inverse: $R^{-}$

$S = ALC + \text{Transitivity}$

**Ontology (= Knowledge Base)**

**Concept Axioms (TBox)**
- Subclass: $C \subseteq D$
- Equivalent: $C \equiv D$

**Role Axioms (RBox)**
- Subrole: $R \subseteq S$
- Transitivity: $\text{Trans}(D)$

**Assertional Axioms (ABox)**
- Instance: $C(a)$
- Role: $R(a,b)$
- Same: $a = b$
- Different: $a \neq b$

**ALC**

$\text{Q (N)}$

$\text{Concept Axioms (TBox)}$
- $C \subseteq D$
- $C \equiv D$

$\text{Role Axioms (RBox)}$
- $R \subseteq S$
- $\text{Trans}(D)$

$\text{Assertional Axioms (ABox)}$
- $C(a)$
- $R(a,b)$
- $a = b$
- $a \neq b$

$S = ALC + \text{Transitivity}$

$\text{OWL DL = SHOIN(D)}$ (D: concrete domain)

**Examples**
- Person $\sqcap$ Female
- Person $\sqcap \exists \text{attends.Course}$
- Person $\forall \text{attends.(Course} \rightarrow \neg \text{Easy})$
- Person $\exists \text{teaches.(Course} \sqcap \forall \text{attended-by.(Bored} \sqcup \text{Sleeping))}$

**Interpretations**
Semantics based on interpretations $(\Delta^X, \mathcal{I})$, where
- $\Delta^X$ is a non-empty set (the domain)
- $\mathcal{I}$ is the interpretation function mapping
  - each concept name $A$ to a subset $A^\mathcal{I}$ of $\Delta^X$
  - each role name $R$ to a binary relation $R^\mathcal{I}$ over $\Delta^X$.  

Intuition: interpretation is complete description of the world
Technically: interpretation is first-order structure with only unary and binary predicates
Example

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 \( A \triangleq C \) iff \( A^\mathcal{I} = C^\mathcal{I} \)
\( \mathcal{I} \) is model of \( \mathcal{T} \) if it satisfies all definitions in \( \mathcal{T} \)

E.g.: Lecturer \( \triangleq \) Person \( \sqcap \exists \text{teaches}.\text{Course} \)
Yields two kinds of concept names: defined and primitive

Semantics of Complex Concepts

\( (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I} \)  \( (C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I} \)  \( (C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I} \)

\( (\exists R.C)^\mathcal{I} = \{ d \mid \text{there is an } e \in \Delta^\mathcal{I} \text{ with } (d, e) \in R^\mathcal{I} \text{ and } e \in C^\mathcal{I} \} \)

\( (\forall R.C)^\mathcal{I} = \{ d \mid \text{for all } e \in \Delta^\mathcal{I}, (d, e) \in R^\mathcal{I} \text{ implies } e \in C^\mathcal{I} \} \)

TBox: Example

TBoxes are used as ontologies:

Woman \( \triangleq \) Person \( \sqcap \) Female
Man \( \triangleq \) Person \( \sqcap \) \( \neg \)Woman
Lecturer \( \triangleq \) Person \( \sqcap \exists \text{teaches}.\text{Course} \)
Student \( \triangleq \) Person \( \sqcap \exists \text{attends}.\text{Course} \)
BadLecturer \( \triangleq \) Person \( \sqcap \forall \text{teaches}.(\text{Course} \rightarrow \text{Boring}) \)
A TBox restricts the set of admissible interpretations.

Lecturer \triangleq \text{Person} \cap \exists \text{teaches}.\text{Course}

Student \triangleq \text{Person} \cap \exists \text{attends}.\text{Course}

**Reasoning Tasks — Classification**

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

- Woman \triangleq \text{Person} \cap \text{Female}
- Man \triangleq \text{Person} \cap \neg \text{Woman}
- MaleLecturer \triangleq \text{Man} \cap \exists \text{teaches}.\text{Course}

Can be computed using multiple subsumption tests

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

**Reasoning Tasks — Subsumption**

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

if

\[ C^{\downarrow} \subseteq D^{\downarrow} \] 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 \text{Person} \cap \exists \text{teaches}.\text{Course}

Student \triangleq \text{Person} \cap \exists \text{attends}.\text{Course}

Then:

Lecturer \cap \exists \text{attends}.\text{Course} \sqsubseteq^\mathcal{T} \text{Student}
A Concept Hierarchy

Excerpt from a process engineering ontology

Reasoning Tasks — Satisfiability

A primitive interpretation for TBox $\mathcal{T}$ interprets

- 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

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

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

Definitorial TBoxes

Acyclic TBoxes

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

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

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

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 $A^\mathcal{T} := C^\mathcal{T}$ for all $A \equiv C \in \mathcal{T}$
Acyclic TBoxes II

For reasoning, acyclic TBox can be eliminated

- to decide $C \subseteq_T D$ with $T$ acyclic,
  - expand $T$
  - replace defined concept names in $C, D$ with their definition
  - decide $C \sqsubseteq D$
- analogously for satisfiability

May yield an exponential blow-up:

$$A_0 \models \forall r.A_1 \land \forall s.A_1$$
$$A_1 \models \forall r.A_2 \land \forall s.A_2$$
$$\ldots$$
$$A_{n-1} \models \forall r.A_n \land \forall s.A_n$$

ABoxes II

ABoxes describe a snapshot of the world

An ABox is a finite set of assertions

\[\alpha : C\]  (\(\alpha\) individual name, \(C\) concept)  
\[(a, b) : R\]  (\(a, b\) individual names, \(R\) role name)

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

Interpretations $\mathcal{I}$ map each individual name $\alpha$ to an element of $\Delta^2$.

$\mathcal{I}$ satisfies an assertion

\[\alpha : C\]  if  
\[\alpha^x \in C^x\]  
\[(a, b) : R\]  if  
\[(a^x, b^x) \in R^x\]

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

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 \land \forall attended-by.Sleeping \sqsubseteq Boring

Interpretation $\mathcal{I}$ is model of general TBox $T$ if

\[C^\mathcal{I} \sqsubseteq D^\mathcal{I}\] for all $C \sqsubseteq D \in T$.

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

e.g. Student \land \exists has-favourite.SoccerTeam \models Student \land \exists has-favourite.Beer

Note: $C \sqsubseteq D$ equivalent to $\top \models C \rightarrow D$

ABoxes

Note:

- interpretations describe the state if the world in a \textbf{complete} way
- ABoxes describe the state if the world in an \textbf{incomplete} way

\[(uli, dl-course) : taught-by\]  uli : Female

does \textbf{not} imply

dl-course : \forall taught-by.Female

An ABox has many models!

An ABox constrains the set of admissible models similar to a TBox
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 $C$, does $\alpha^2 \in C^2$ hold in all models of $\mathcal{A}$ and $\mathcal{T}$?

(written $\mathcal{A}, \mathcal{T} \models \alpha : C$)

The two tasks are interreducible:

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

Example for ABox Reasoning

<table>
<thead>
<tr>
<th>ABox</th>
<th>dumbo : Mammal</th>
<th>t14 : Trunk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>q23 : Darkgrey</td>
<td>(dumbo, t14) : bodypart</td>
</tr>
<tr>
<td></td>
<td>(dumbo, q23) : color</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dumbo : $\forall$color.Lightgrey</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TBox</th>
<th>Elephant $\models$ Mammal $\sqcap \exists$bodypart.Trunk $\sqcap \forall$color.Grey</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Grey $\models$ Lightgrey $\sqcup$ Darkgrey</td>
</tr>
<tr>
<td></td>
<td>$\bot$ $\models$ Lightgrey $\sqcup$ Darkgrey</td>
</tr>
</tbody>
</table>

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

2. Tableau algorithms for $\mathcal{ALC}$ and extensions

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

- $\checkmark$ general TBoxes and
- $\checkmark$ 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:** $C_0 \supseteq C_1$ and $C_1 \subseteq C_0$

**NNF:** negation occurs only in front of concept names

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

\[
\neg(C \cap D) \iff \neg C \cup \neg D \\
\neg(C \cup D) \iff \neg C \cap \neg D \\
\neg\forall R.C \iff \exists R.\neg C \\
\neg\exists R.C \iff \forall R.\neg C
\]

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 $A \cap \exists R.B \cap \forall R.\neg B$ is satisfiable...

$A \cap \exists R.B \cap \forall R.\neg B \supseteq \exists S.E$

Our tableau algorithm works on a completion tree which

- represents a model $\mathcal{I}$: nodes represent elements of $\Delta^x$
  - each node $x$ is labelled with concepts $\mathcal{L}(x) \subseteq \text{sub}(C_0)$
  - $C \in \mathcal{L}(x)$ is read as "$x$ should be an instance of $C$"

- edges represent role successorship
  - each edge $(x, y)$ is labelled with a role-name from $C_0$
  - $R \in \mathcal{L}((x, y))$ is read as "$(x, y)$ should be in $R^x$"

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

- is expanded using completion rules

Completion rules for $\mathcal{ALC}$

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

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

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

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

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

We only apply rules if their application does “something new”

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

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

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

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

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

\( \sqcup \)-rule: if \( C_1 \sqcup C_2 \in L(x) \) 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 our tableau algorithm**

**Lemma:** Let \( C_0 \) an 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 ALC.
2. Satisfiability (and subsumption) in ALC is decidable in PSpace.
3. ALC has the finite model property
   i.e., every satisfiable concept has a finite model.
4. ALC has the tree model property
   i.e., every satisfiable concept has a tree model.
5. ALC has the finite tree model property
   i.e., every satisfiable concept has a finite tree model.

**Extend tableau algorithm to 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 \( \mathcal{T} \) iff \( \mathcal{I} \) satisfies each concept equation in \( \mathcal{T} \)
- \( C_0 \) is satisfiable w.r.t. \( \mathcal{T} \) iff there is a model \( \mathcal{I} \) of \( \mathcal{T} \) with \( C_0^\mathcal{I} \neq \emptyset \)

**Goal – Lemma:** Let \( C_0 \) an ALC-concept and \( \mathcal{T} \) be a an ALC-TBox. Then
1. the algorithm terminates when applied to \( \mathcal{T} \) 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. \( \mathcal{T} \).
We extend our tableau algorithm by adding a new completion rule:

- remember that nodes represent elements of $\Delta^I$ and
- if $C \sqsubseteq D \in T$, then for each element $x$ in a model $I$ of $T$
  - if $x \in C^I$, then $x \in D^I$
  - hence $x \in (\neg C)^I$ or $x \in D^I$
  - $x \in (\neg C \cup D)^I$
  - $x \in (\text{NNF}(\neg C \cup D))^I$

for $\text{NNF}(E)$ the negation normal form of $E$.

My attempt:

- $C \sqsubseteq D \in T$ then for each element $x$ in a model $I$ of $T$
  - if $x \in C^I$, then $x \in D^I$
  - hence $x \in (\neg C)^I$ or $x \in D^I$
  - $x \in (\neg C \cup D)^I$
  - $x \in (\text{NNF}(\neg C \cup D))^I$

for $\text{NNF}(E)$ the negation normal form of $E$.

---

**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 $x$.

---

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

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

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

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

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

- $T$-rule: if $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2 \in T$ and $\text{NNF}(\neg \mathcal{C}_1 \sqcup \mathcal{C}_2) \not\subseteq \mathcal{L}(x)$
  then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\text{NNF}(\neg \mathcal{C}_1 \sqcup \mathcal{C}_2)\}$

---

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

- $x$ is directly blocked if it has an ancestor $y$ with $\mathcal{L}(x) \subseteq \mathcal{L}(y)$
- 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

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

$\leadsto$ completion rules for $\mathcal{ALC}$ w.r.t. TBoxes
A tableau algorithm for $\mathcal{ALC}$ with general TBoxes

\begin{itemize}
  \item $\land$-rule: if $C_1 \cap C_2 \in \mathcal{L}(x)$, \{\text{\textit{C}}_1, C_2\} \not\subset \mathcal{L}(x)$, and $x$ is not blocked
  
  then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$

  \item $\lor$-rule: if $C_1 \cup C_2 \in \mathcal{L}(x)$, \{\text{\textit{C}}_1, C_2\} \cap \mathcal{L}(x) = \emptyset$, and $x$ is not blocked
  
  then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

  \item $\exists$-rule: if $\exists S.C \in \mathcal{L}(x)$, $x$ has no $S$-successor $y$ with $C \in \mathcal{L}(y)$, and $x$ is not blocked
  
  then create a new node $y$ with $\mathcal{L}(\langle x, y \rangle) = \{S\}$ and $\mathcal{L}(y) = \{C\}$

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

  \item $T$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$, $\text{\textit{NNF}}(\neg C_1 \sqcup C_2) \not\in \mathcal{L}(x)$ and $x$ is not blocked
  
  then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\text{\textit{NNF}}(\neg C_1 \sqcup C_2)\}$
\end{itemize}

### Tableaux Rules for $\mathcal{ALC}$

\begin{table}[h]
\begin{tabular}{|c|c|}
  \hline
  $x \bullet \{C_1 \cap C_2, \ldots\}$ & $\neg \land$ & $x \bullet \{C_1 \cap C_2, C_1, C_2, \ldots\}$ \\
  \hline
  $x \bullet \{C_1 \cup C_2, \ldots\}$ & $\lor$ & $x \bullet \{C_1 \cup C_2, C, \ldots\}$ for $C \in \{C_1, C_2\}$ \\
  \hline
  $x \bullet \{\exists R.C, \ldots\}$ & $\exists$ & $x \bullet \{\exists R.C, \ldots\}$
  
  $R$ & $y \bullet \{C\}$ \\
  \hline
  $x \bullet \{\forall R.C, \ldots\}$ & $\forall$ & $x \bullet \{\forall R.C, \ldots\}$
  
  $R$ & $y \bullet \{C, \ldots\}$ \\
  \hline
\end{tabular}
\end{table}

### Tableaux Rule for Transitive Roles

Where $R$ is a transitive role (i.e., $(R^2)^+ = R^2$)

- No longer naturally terminating (e.g., if $C = \exists R.\top$)
- Need blocking
  - Simple blocking suffices for $\mathcal{ALC}$ 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 $\exists S.C \cap \forall S. (\neg C \cup \neg D) \cap \exists R.C \cap \forall R.(\exists R.C)$ where $R$ is a transitive role
Tableaux Algorithm — Example

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

$L(w) = \{\exists S.C \cap \forall S.(\neg C \cup \neg D) \cap \exists R.C \cap \forall R.(\exists R.C)\}$
Test satisfiability of $\exists S.C \land \forall S. (\neg C \lor \neg D) \land \exists R.C \land \forall R.(\exists R.C)$ where $R$ is a transitive role.

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

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

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

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

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.

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

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

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

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

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

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

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

$L(x) = \{C, \neg C \sqcup \neg D\}$
Test satisfiability of $\exists S.C \cap \forall S. (\neg C \cup \neg D) \cap \exists R.C \cap \forall R. (\exists R.C)$ where $R$ is a transitive role

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

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

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

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

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

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

Reasoning with Expressive Description Logics – p. 7/27

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

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

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

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

$L_z = \{C\}$

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

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

$L(x) = \{C, (\neg C \lor \neg D), \neg D\} \quad \text{(2)}$

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

\begin{array}{c}
\text{blocked} \\
L(z) = \{C, \exists R.C, \forall R. (\exists R.C)\}
\end{array}$

Concept is satisfiable: $T$ corresponds to model

Properties of our tableau algorithm for ALC with TBoxes

Lemma: Let $T$ be a general ALC-Tbox and $C_0$ an 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 ALC-concept w.r.t. TBoxes is decidable
2. ALC with TBoxes has the finite model property
3. ALC with TBoxes has the tree model property
A tableau algorithm for $\mathcal{ALC}$ with general TBoxes: Summary

The tableau algorithm presented here

- decides satisfiability of $\mathcal{ALC}$-concepts w.r.t. TBoxes, and thus also
- decides subsumption of $\mathcal{ALC}$-concepts w.r.t. TBoxes
- uses blocking to ensure termination, and
- is non-deterministic due to the $\rightarrow_{U}$-rule
- in the worst case, it builds a tree of depth exponential in the size of the input, and thus of double exponential size. Hence it runs in (worst case) $2\text{NExpTime}$,
- can be implemented in various ways,
  - order/priorities of rules
  - data structure
  - etc.
- is amenable to optimisations – more on this next week

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 $\mathcal{SHIQ}$ 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

Challenges

- **Increased expressive power**
  - Existing DL systems implement (at most) $\mathcal{SHIQ}$
  - OWL extends $\mathcal{SHIQ}$ 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

Resources

- Slides from this talk
  [http://www.cs.man.ac.uk/~horrocks Slides/Innsbruck-tutorial/]
- FaCT system (open source)
  [http://www.cs.man.ac.uk/~FaCT/]
- OilEd (open source)
  [http://oiled.man.ac.uk/]
- OIL
  [http://www.ontoknowledge.org/oil/]
- W3C Web-Ontology (WebOnt) working group (OWL)
  [http://www.w3.org/2001/sw/WebOnt/]
- **DL Handbook**, Cambridge University Press
  [http://books.cambridge.org/0521781760.htm]