Approximate Reasoning for the Semantic Web
Part II
OWL Semantics and Tableau Reasoning

Frank van Harmelen
Pascal Hitzler
Holger Wache

ESSLLI 2006 Summer School
Malaga, Spain, August 2006

Introducing the speaker

- 1998 Diplom (Master) in Mathematics
  - Uni Tübingen (Helmut Salzmann)
- 1999-2001 PhD in Mathematics
  - Cork, Irland (Tony Seda)
  - Formal Aspects of Knowledge Representation
- 2001-2004 Postdoc
  - TU Dresden, Artificial Intelligence (Steffen Hölldobler)
- since 2004 Assistant Professor
  - AIFB Univ. Karlsruhe, Semantic Web (Rudi Studer)
- 2005 Habilitation in Computer Science

Main Interests:
Semantic Web (Knowledge Representation/Logic)
Neural-symbolic Integration
Mathematical Foundations of Artificial Intelligence
Karlsruhe: Location for Semantic Technologies and Applications

Semantic Karlsruhe

Knowledge Management
- B2B, EAI
- Business Intelligence
- Electronic Markets
- eGovernment

Semantic Web Infrastructure
- Ontology Management
- Data, Web & Text Mining
- Peer-to-Peer, Semantic Grid
- Semantic Web Services

Who are we? ... Semantic Web Research Group

AIFB
- Sudhir Agarwal
- Anupriya Ankolekar
- Stephan Bloehdorn
- Sebastian Blohm
- Saartje Brockmans
- Philipp Cimiano
- Peter Haase
- Jens Hartmann
- Pascal Hitzler
- Markus Krötzsch
- Steffen Lamparter
- Holger Lewen
- Yimin Wang
- Juliet Tane
- Thanh Tran Duc
- Tuvshintur Tserendorj
- Christoph Tempich
- Yimin Wang
- Johanna Völker
- Denny Vrandecic

FZI
- Andreas Abecker
- Simone Braun
- Stephan Grimm
- Heiko Haller
- Hans-Jörg Happel
- Mark Hefke
- Ljiljana Stojanovic
- Nenad Stojanovic
- Max Völkel
- Valentin Zacharias

& ~40 people at Ontoprise
Partners and Projects

Semantic Web Layer Cake

- Data
- Ontology
- RDF + rdfschema
- XML + NS + xsmlschema
- Unicode
- URI
- Self-desc. doc.
- Now
- Logic
- OWL++
- Vocabulary
- Trust
- Proof
- Rules
What is Semantics?

Syntax: strings without meaning
Semantics: meaning of syntax

Why logic is so successful:
Semantics can be captured syntactically!

What is semantics? Example programming language

Syntax

computing factorial

intended semantics

FUNCTION f(n:natural):natural;
BEGIN
  IF n=0 THEN f:=1
  ELSE f:=n*f(n-1);
END;

What the program does when executed

f : n ⟷ n!

formal semantics

procedural semantics
Semantics of logics/knowledge representation languages

∀ x (p(x) → q(x))

all humans are mortal

intended semantics

⊨

logical consequence

model theoretic semantics

deducible in a calculus

proof theoretic semantics

Part II contents

1. OWL Model-theoretic Semantics
   a. Description Logics: ALC
   b. OWL as SHOIN(D)
   c. OWL Examples
2. Proof Theory
   a. Reasoning as Satisfiability checking
   b. Tableaux Reasoning
Description Logics, DLs

- FOL (First-Order Logic) fragments
- usually decidable
- "expressive"
- come from semantic networks and frame systems
- close relation with multi-modal logics

- W3C Standard OWL DL is the description logic SHOIN(D)
- We first talk about the simpler ALC

General DL Architecture

Knowledge Base

- Tbox (schema)
  - Man ≡ Human ⊓ Male
  - Happy-Father ≡ Man ⊓ ∃ has-child.Female ⊓ ...

- Abox (data)
  - Happy-Father(John)
  - has-child(John, Mary)
DLs – general remarks

- DLs are a family of logic-based KR formalisms
- DLs characterised by:
  - Different constructors for generating complex class expressions.
  - Axioms for describing properties for roles.

- ALC is the smallest DL which is propositionally closed
  - Conjunction, disjunction, negation are constructors, written as $\land$, $\lor$, $\neg$.
  - Quantifiers used only together with roles:

\[
\text{Man} \land \exists \text{hasChild}.\text{Female} \land \exists \text{hasChild}.\text{Male} \\
\land \forall \text{hasChild}.(\text{Rich} \lor \text{Happy})
\]

Other DL language components

- E.g.
  - Number restrictions (cardinality constraints) for Roles:
    $\geq 3 \text{ hasChild}$, $\leq 1 \text{ hasMother}$
  - Qualified number restrictions:
    $\geq 2 \text{ hasChild}.\text{Female}$, $\leq 1 \text{ hasParent}.\text{Male}$
  - Nominals (definition by extension): \{Italy, France, Spain\}
  - Concrete domains (datatypes): $\text{hasAge.}(\geq 21)$
  - Inverse roles: $\text{hasChild}^{-} \equiv \text{hasParent}$
  - Transitive roles: $\text{hasAncestor} \sqsubseteq \text{hasAncestor}^{+}$
  - Role composition: $\text{hasParent}.\text{hasBrother}(\text{uncle})$
ALC: basic language elements

- basic language components:
  - classes
  - roles
  - individuals

- Professor(RudiStuder)
  - Individual RudiStuder is in class Professor
- affiliation(RudiStuder,AIFB)
  - RudiStuder has affiliation AIFB

ALC: subclass relation

- Professor $\sqsubseteq$ Faculty
  - translates to $(\forall x)(\text{Professor}(x) \rightarrow \text{Faculty}(x))$
  - corresponds to owl:subClassOf

- Professor $\equiv$ Faculty
  - translates to $(\forall x)(\text{Professor}(x) \leftrightarrow \text{Faculty}(x))$
  - corresponds to owl:equivalentClass
**ALC: complex class descriptions**

- conjunction $\land$
- disjunction $\lor$
- negation $\neg$

- Professor $\subseteq (\text{Person} \land \text{Faculty})$
  $\quad \lor (\text{Person} \land \neg \text{PhDStudent})$

$$(\forall x)(\text{Professor}(x) \rightarrow ((\text{Person}(x) \land \text{Faculty}(x)) \lor (\text{Person}(x) \land \neg \text{PhDStudent}(x))))$$

---

**ALC: Quantifiers**

- Exam $\subseteq \forall \text{hasExaminer}.\text{Professor}$
  $$(\forall x)(\text{Exam}(x) \rightarrow (\forall y)(\text{hasExaminer}(x,y) \rightarrow \text{Professor}(y))))$$
  – corresponds to owl:allValuesFrom

- Exam $\subseteq \exists \text{hasExaminer}.\text{Person}$
  $$(\forall x)(\text{Exam}(x) \rightarrow (\exists y)(\text{hasExaminer}(x,y) \land \text{Person}(y))))$$
  – corresponds to owl:someValuesFrom
Modelling in ALC

- owl:nothing: \( \bot \equiv C \sqcap \neg C \)
- owl:thing: \( T \equiv C \sqcup \neg C \)
- owl:disjointWith: \( C \sqcap D \equiv \bot \)
extequivalently: \( C \sqsubseteq \neg D \)
- rdfs:range: \( T \sqsubseteq \forall R.C \)
- rdfs:domain: \( \exists R.T \sqsubseteq C \)

ALC: Syntax

- The following rules generate classes in ALC, where A is an atomic (named) class and R is a role.

\[
\begin{align*}
C, D &\rightarrow A \mid T \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C
\end{align*}
\]

- An ALC TBox consists of assertions (axioms) of the form \( C \sqsubseteq D \) and \( C \equiv D \), where C,D are classes.

- An ALC ABox consists of assertions of the form \( C(a) \) and \( R(a,b) \), where C is a complex class, R is a role and a,b are individuals.

- An ALC-knowledge base consists of an ABox and a TBox.
**ALC: Semantics**

Defined by translating TBox axioms into FOL via the mapping \( \pi \) (shown to the right).

Here, \( C, D \) are complex classes, \( R \) is a role and \( A \) is an atomic class.

\[
\begin{align*}
\pi(C \sqsubseteq D) &= (\forall x)(\pi_x(C) \to \pi_x(D)) \\
\pi(C \equiv D) &= (\forall x)(\pi_x(C) \leftrightarrow \pi_x(D)) \\
\pi_x(A) &= A(x) \\
\pi_x(\neg C) &= \neg \pi_x(C) \\
\pi_x(C \cap D) &= \pi_x(C) \land \pi_x(D) \\
\pi_x(C \cup D) &= \pi_x(C) \lor \pi_x(D) \\
\pi_y(\forall R.C) &= (\forall y)(R(x, y) \to \pi_y(C)) \\
\pi_y(\exists R.C) &= (\exists y)(R(x, y) \land \pi_y(C))
\end{align*}
\]

**DL knowledge bases**

- DL knowledge bases consist of two parts:
  - **TBox**: Axioms containing schema knowledge:
    - \textbf{HappyFather} \equiv \text{Man} \sqcap \exists \text{hasChild.Female} \sqcap \ldots
    - \textbf{Elephant} \sqsubseteq \text{Animal} \sqcap \text{Large} \sqcap \text{Grey}
    - \text{transitive(hasAncestor)}
  
  - **Abox**: Axioms describing data:
    - \textbf{HappyFather}(John)
    - \textbf{hasChild}(John, Mary)

- Distinction between ABox and TBox has no logical significance whatsoever
  ...but it makes some things easier to talk about.
**Simple example**

**Terminological knowledge (TBox):**
Human ⊑ ∃parentOf.Human
Orphan ≡ Human ⊓ ¬∃hasParent.Alive

**Data (ABox):**
Orphan(harrypotter)
hasParent(harrypotter,jamespotter)

Semantics and logical consequences are understood via translation to FOL.

---

**Part II contents**

1. OWL Model-theoretic Semantics
   a. Description Logics: ALC
   b. **OWL as SHOIN(D)**
   c. OWL Examples
2. Proof Theory
   a. Reasoning as Satisfiability checking
   b. Tableaux Reasoning
OWL and ALC

The following OWL DL primitives are expressible in ALC:
• classes, roles, individuals
• class membership, role instances
• \( \top \) and \( \bot \)
• class inclusion, equivalence, and disjointness
• \( \sqcap \), \( \sqcup \)
• \neg
• role restrictions
• range and domain

OWL as SHOIN(D): Individuals

• owl:sameAs
  – equality of individuals
  – DL: \( a=b \)
  – FOL: need extension with equality predicate

• owl:differentFrom
  – inequality of individuals
  – DL: \( a \neq b \)
  – FOL: \( \neg(a=b) \)
OWL as SHOIN(D): nominals

Nominals
- `owl:oneOf`
  - closed class (definition by extension)
  - DL: \( C \equiv \{a, b, c\} \)
  - FOL: \( (\forall x)(C(x) \leftrightarrow (x=a \lor x=b \lor x=c)) \)

OWL as SHOIN(D): number restrictions

Number restrictions need equality predicate

An exam may have at most two examiners.
- DL: Exam \( \subseteq \leq 2 \) hasExaminer
- In FOL: \( (E \ldots \text{Exam}, h \ldots \text{hasExaminer}) \)
  \( (\forall x)(E(x) \rightarrow \neg(\exists x_1)(\exists x_2)(\exists x_3)(x_1 \neq x_2 \land x_2 \neq x_3 \land x_1 \neq x_3 \land h(x, x_1) \land h(x, x_2) \land h(x, x_3))) \)

Similarly for the other number restrictions.
**OWL as SHOIN(D): role constructors**

- **Rdfs:subPropertyOf**
  - **DL:** \( R \sqsubseteq S \)
  - **FOL:** \( (\forall x)(\forall y)(R(x,y) \rightarrow S(x,y)) \)

- similarly for role equivalence

- **Inverse Roles:** \( R \equiv S^- \)
  - **FOL:** \( (\forall x)(\forall y)(R(x,y) \leftrightarrow S(y,x)) \)

- **Transitive Roles:** \( R^+ \sqsubseteq R \)
  - **FOL:** \( (\forall x)(\forall y)(\forall z)(R(x,y) \land R(y,z) \rightarrow R(x,z)) \)

- **Symmetry:** \( R \equiv R^- \)

- **Functionality:** \( T \sqsubseteq \leq 1 R \)

- **Inverse Functionality:** \( T \sqsubseteq \leq 1 R^- \)

---

**datatypes**

- Allow usage of datatypes in the second argument of concrete roles in the ABox.

- Furthermore, a nominal (closed class) can consist of a set of datatype elements.

- It is not possible to express datatypes directly in FOL. But FOL syntax/semantics can be extended to encompass datatypes.
OWL DL as SHOIN(D): overview

Allowed are:
- ALC
- Equality and inequality between individuals
- Nominals
- Number restrictions
- Subroles and role equivalence
- Inverse and transitive roles
- datatypes

Naming conventions for DLs

- ALC: Attribute Language with Complement
- S: ALC + role transitivity
- H: subrole relations
- O: nominals
- I: inverse roles
- N: number restrictions $\leq n \ R$ etc.
  - Q: Qualified number restrictions $\leq n \ R.C$ etc.
- (D): Datatypes
- F: Functional roles

- OWL DL is SHOIN(D)
- OWL Lite is SHIF(D)
Overview syntax for DLs (without datatypes)

### 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: \( R^- \)

---

**Ontology (=Knowledge Base)**

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

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

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

---

**S = ALC + Transitivity**
**OWL DL = SHOIN(D)**

(D: concrete domain)

---

**OWL DL as DL: overview class constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>( C_1 \sqcap \ldots \sqcap C_n )</td>
<td>Human \sqcap \text{Male}</td>
<td>( C_1(x) \land \ldots \land C_n(x) )</td>
</tr>
<tr>
<td>unionOf</td>
<td>( C_1 \sqcup \ldots \sqcup C_n )</td>
<td>Doctor \sqcup \text{Lawyer}</td>
<td>( C_1(x) \lor \ldots \lor C_n(x) )</td>
</tr>
<tr>
<td>complementOf</td>
<td>( \neg C )</td>
<td>\text{\neg Male}</td>
<td>( \neg C(x) )</td>
</tr>
<tr>
<td>oneOf</td>
<td>( {x_1} \sqcup \ldots \sqcup {x_n} )</td>
<td>{john} \sqcup {mary}</td>
<td>( \forall y. P(x, y) \rightarrow C(y) )</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>( \forall P.C )</td>
<td>\text{\forall hasChild.Doctor}</td>
<td>( \exists y. P(x,y) \land C(y) )</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>( \exists P.C )</td>
<td>\text{\exists hasChild.Lawyer}</td>
<td>( \exists y. P(x,y) \sqsubseteq C(y) )</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>( \leq nP )</td>
<td>\leq1hasChild</td>
<td>( \exists n y. P(x, y) )</td>
</tr>
<tr>
<td>minCardinality</td>
<td>( \geq nP )</td>
<td>\geq2hasChild</td>
<td>( \exists n y. P(x, y) )</td>
</tr>
</tbody>
</table>

nesting is allowed:

\( \text{Person} \sqcap \forall \text{hasChild.(Doctor} \sqcup \exists \text{hasChild.Doctor}) \)
**OWL DL as DL: axioms**

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td>Male $\sqsubseteq \neg$ Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>${x_1} \equiv {x_2}$</td>
<td>${\text{President_Bush}} \equiv {\text{G_W_Bush}}$</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \sqsubseteq \neg {x_2}$</td>
<td>${\text{john}} \sqsubseteq \neg {\text{peter}}$</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \sqsubseteq \neg P_2$</td>
<td>hasChild $\equiv$ hasParent $\neg$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor $^+$ $\sqsubseteq$ ancestor $^+$</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$T \sqsubseteq 1P$</td>
<td>$T \sqsubseteq 1\text{hasMother}$</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$T \sqsubseteq 1\neg P$</td>
<td>$T \sqsubseteq 1\text{hasSSN}$</td>
</tr>
</tbody>
</table>

- **General Class Inclusion ($\sqsubseteq$) suffices:**
  
  $C \equiv D \iff (\forall x) (C(x) \leftrightarrow D(x))$

- **Equivalences**
  
  $C \sqsubseteq D \iff (\forall x) (C(x) \rightarrow D(x))$

---

**Computational complexity (worst-case)**

<table>
<thead>
<tr>
<th>OWL variant</th>
<th>data complexity</th>
<th>combined complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL Full</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>OWL DL</td>
<td>unknown</td>
<td>NExptime</td>
</tr>
<tr>
<td>OWL DL without nominals</td>
<td>NP (IJCAI 2005)</td>
<td>Exptime</td>
</tr>
<tr>
<td>OWL Lite</td>
<td>NP</td>
<td>Exptime</td>
</tr>
</tbody>
</table>

- **data complexity:** complexity w.r.t. ABox size
- **combined complexity:** complexity w.r.t. ABox and TBox size
Part II contents

1. OWL Model-theoretic Semantics
   a. Description Logics: ALC
   b. OWL as SHOIN(D)
   c. OWL Examples
2. Proof Theory
   a. Reasoning as Satisfiability checking
   b. Tableaux Reasoning

Examples (abstract syntax)

Class(a:bus_driver complete intersectionOf(a:person 
   restriction(a:drives someValuesFrom (a:bus)))))

\[ \text{bus\_driver} \equiv \text{person} \cap \exists \text{drives\_bus} \]

Class(a:driver complete intersectionOf(a:person 
   restriction(a:drives someValuesFrom (a:vehicle)))))

Class(a:bus partial a:vehicle)

\[ \text{driver} \equiv \text{person} \cap \exists \text{drives\_vehicle} \]

\[ \text{bus} \sqsubseteq \text{vehicle} \]

- A bus driver is a person that drives a bus.
- A bus is a vehicle.
- **A bus driver** drives a vehicle, so must be a driver.

The subclass is inferred due to subclasses being used in existential quantification.
Examples

\[
\text{driver} \equiv \text{person} \cap \exists \text{drives.vehicle}
\]

Class(a:driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:vehicle))))

Class(a:driver partial a:adult)

Class(a:grownup complete intersectionOf(a:adult a:person))

- Drivers are defined as persons that drive cars (complete definition)
- We also know that drivers are adults (partial definition)
- So all drivers must be adult persons (i.e. grownups)

An example of axioms being used to assert additional necessary information about a class. We do not need to know that a driver is an adult in order to recognize one, but once we have recognized a driver, we know that they must be adult.

Part II contents

1. OWL Model-theoretic Semantics
   a. Description Logics: ALC
   b. OWL as SHOIN(D)
   c. OWL Examples
2. Proof Theory
   a. Reasoning as Satisfiability checking
   b. Tableaux Reasoning
Important inference problems

• global consistency of knowledge base \( KB \not\models false \)?
  – Is knowledge base reasonable?
• class consistency \( C \equiv \bot \)?
  – Is class \( c \) well-modeled?
• subsumption \( C \sqsubseteq D \)?
  – structuring the knowledge base
• class equivalence \( C \equiv D \)?
  – Are they actually the same?
• class disjointness \( C \cap D = \bot \)?
  – Do they have common instances?
• class membership \( C(a) \)?
  – Does the instance belong to the class?
• instance retrieval „find all \( X \) with \( C(X) \)“
  – Give me all instances with some given properties.

Decidability of OWL DL reasoning

• Decidability: For every inference problem there's a terminating decision procedure.
• OWL DL is FOL fragment. In principle, one could attempt to use standard FOL algorithms
• But they don't always terminate!

• Problem: Find terminating algorithms.
Reasoning via satisfiability checking

- We will modify standard tableaux algorithms.
  - We will cover ALC only.
- Tableaux algorithms work by showing unsatisfiability.

→ Reduce reasoning to finding inconsistencies in some knowledge base, i.e. we show unsatisfiability of the knowledge base!

Reasoning by satisfiability checking

- class consistency $C \equiv \bot$?
  - $KB \cup \{C(a)\}$ unsatisfiable (a new)
- subsumption $C \sqsubseteq D$?
  - $KB \cup \{C \sqcap \neg D(a)\}$ unsatisfiable (a new)
- class equivalence $C \equiv D$?
  - $C \sqsubseteq D$ and $D \sqsubseteq C$
- class disjointness $C \sqcap D = \bot$?
  - $KB \cup \{(C \sqcap D)(a)\}$ unsatisfiable (a new)
- class membership $C(a)$?
  - $KB \cup \{\neg C(a)\}$ unsatisfiable (a new)
- instance retrieval "find all X with C(X)"
  - Check membership for all known individuals
Part II contents

1. OWL Model-theoretic Semantics
   a. Description Logics: ALC
   b. OWL as SHOIN(D)
   c. OWL Examples
2. Proof Theory
   a. Reasoning as Satisfiability checking
   b. Tableaux Reasoning

ALC Tableaux: contents

• Transformation to negation normal form
• Naive tableau algorithm
• Tableau algorithm with Blocking
Transformation to negation normal form

Let $W$ be a knowledge base

- replace $C \equiv D$ by $C \subseteq D$ and $D \subseteq C$.
- replace $C \subseteq D$ by $\neg C \sqcup D$.
- Apply rules on next slides exhaustively.

resulting knowledge base: $\text{NNF}(W)$

*negation normal form of* $W$.

Negation only occurs directly in front of atomic classes.

\[
\begin{align*}
\text{NNF}(C) &= C, \text{ if } C \text{ atomic} \\
\text{NNF}(\neg C) &= \neg C, \text{ if } C \text{ atomic} \\
\text{NNF}(\neg \neg C) &= \text{NNF}(C) \\
\text{NNF}(C \sqcup D) &= \text{NNF}(C) \sqcup \text{NNF}(D) \\
\text{NNF}(C \sqcap D) &= \text{NNF}(C) \sqcap \text{NNF}(D) \\
\text{NNF}(\neg (C \sqcup D)) &= \text{NNF}(\neg C) \sqcap \text{NNF}(\neg D) \\
\text{NNF}(\neg (C \sqcap D)) &= \text{NNF}(\neg C) \sqcup \text{NNF}(\neg D) \\
\text{NNF}(\forall R.C) &= \forall R.\text{NNF}(C) \\
\text{NNF}(\exists R.C) &= \exists R.\text{NNF}(C) \\
\text{NNF}(\neg \forall R.C) &= \exists R.\text{NNF}(\neg C) \\
\text{NNF}(\neg \exists R.C) &= \forall R.\text{NNF}(\neg C)
\end{align*}
\]

$W$ and $\text{NNF}(W)$ are logically equivalent.
negation normal form: example

\[ P \subseteq (E \cap U) \cup \neg(\forall R.E \cup D). \]

In negation normal form:

\[ \neg P \cup (E \cap U) \cup (\exists R.\neg E \cap \neg D). \]
Naive tableau algorithm

reduce to finding an inconsistency

Idea:
• Given a knowledge base $W$.
• Generate consequences of the form $C(a)$ and $\neg C(a)$, until a contradiction is found.

Simple example

$C(a)$
$(\neg C \land D)(a)$

$\neg C(a)$ is logical consequence:
2nd formula in FOL: $\neg C(a) \land D(a)$

hence $\neg C(a)$

Contradiction has been found.
Another example

\[ C(a) \quad \neg C \lor D \quad \neg D(a) \]

Derive logical consequences:

\[ C(a) \]
\[ \neg D(a) \]
\[ (\neg C \lor D)(a) \]

Now we consider two cases

1. \( \neg C(a) \)
   contradiction
2. \( D(a) \)
   contradiction

Splitting of the tableau into two branches

---

Tableau: definitions

- **Tableau branch**: Finite set of statements of the form \( C(a), \neg C(a), R(a,b) \).
- **Tableau**: Finite set of tableau branches.
- A tableau branch is *closed* if it contains a pair \( C(a) \) and \( \neg C(a) \) of contradictory statements.
- Tableau is *closed* if every branch of it is closed.
Generating a tableau

<table>
<thead>
<tr>
<th>Selection</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C(a) \in W$ (ABox)</td>
<td>Add $C(a)$.</td>
</tr>
<tr>
<td>$R(a, b) \in W$ (ABox)</td>
<td>Add $R(a, b)$.</td>
</tr>
<tr>
<td>$C \in W$ (TBox)</td>
<td>Add $C(a)$ for a known individual $a$.</td>
</tr>
<tr>
<td>$(C \sqcap D)(a) \in A$</td>
<td>Add $C(a)$ and $D(a)$.</td>
</tr>
<tr>
<td>$(C \sqcup D)(a) \in A$</td>
<td>Duplicate branch. Add $C(a)$ to one branch and $D(a)$ to the other.</td>
</tr>
<tr>
<td>$(\exists R.C)(a) \in A$</td>
<td>Add $R(a, b)$ and $C(b)$ for a new individual $b$.</td>
</tr>
<tr>
<td>$(\forall R.C)(a) \in A$</td>
<td>If $R(a, b) \in A$ then add $C(b)$.</td>
</tr>
</tbody>
</table>

If the resulting tableau is closed, then the original knowledge base is unsatisfiable.

Always select only such elements which add new elements to the tableau. If this is impossible, then terminate the algorithm – then the knowledge base is satisfiable.

Example

- P ... Professor
  E ... Person
  U ... University member
  D ... PhD student
- knowledge base: $P \subseteq (E \cap U) \cup (E \cap \neg D)$
  Is $P \subseteq E$ a logical consequence?
- Knowledge base (with query) in NNF:
  $\{\neg P \cup (E \cap U) \cup (E \cap \neg D), (P \cap \neg E)(a)\}$
Example (ctd)

TBox: $\neg P \cup (E \cap U) \cup (E \cap \neg D)$

Tableau:

$(P \cap \neg E)(a)$ (from knowledge base)

$P(a)$

$\neg E(a)$

$\neg P \cup (E \cap U) \cup (E \cap \neg D))(a)$

$\neg P(a)$

$((E \cap U) \cup (E \cap \neg D))(a)$

$(E \cap U)(a)$

$(E \cap \neg D)(a)$

$E(a)$

$U(a)$

$\neg D(a)$

Knowledge base is unsatisfiable, i.e. $P \subseteq E$.

The termination problem

- Single Axiom: $\neg \text{Person} \cup \exists \text{hasParent.} \text{Person}$

  we want to show: $\neg \text{Person}(\text{Bill})$

  Problem occurs due to existential quantification (and minCardinality)
**ALC Tableaux: contents**

- Transformation to negation normal form
- Naive tableau algorithm
- Tableau algorithm with Blocking

**Solving the termination problem**

- The following happened:

  ![Diagram](image1)

- But why not do it this way:

  ![Diagram](image2)

  I.e. reuse old nodes!
  Formal proof required that this suffices!
Tableau with Blocking

- Single Axiom: \( \neg \text{Person} \cup \exists \text{hasParent}. \text{Person} \)
  
  We want to show: \( \neg \text{Person}(\text{Bill}) \)

\[
\begin{align*}
\text{Person}(\text{Bill}) & \quad \cup \\
\neg \text{Person}(\text{Bill}) & \quad \cap \\
\exists \text{hasParent}. \text{Person}(\text{Bill}) & \\
\text{hasParent}(\text{Bill},x_1) & \quad \exists \\
\text{Person}(x_1) & \\
\neg \text{Person} \cup \exists \text{hasParent}. \text{Person}(x_1) & \quad \cap \\
\neg \text{Person}(x_1) & \quad \cup \\
\exists \text{hasParent}. \text{Person}(x_1) & \\
\sigma(\text{Bill}) = \{ \text{Person, } \\
\neg \text{Person} \cup \exists \text{hasParent}. \text{Person, } \\
\exists \text{hasParent}. \text{Person} \} & \\
\sigma(x_1) = \{ \text{Person, } \\
\neg \text{Person} \cup \exists \text{hasParent}. \text{Person, } \\
\exists \text{hasParent}. \text{Person} \} & \\
\sigma(x_1) \subseteq \sigma(\text{Bill}), \text{ so Bill blocks } x_1 & \\
\end{align*}
\]

Blocking

The selection of \( \exists R.C(a) \) in branch A is *blocked* if there is an individual \( b \) with
\[
\{ C | C(a) \in A \} \subseteq \{ C | C(b) \in A \}.
\]

Now two ways of terminating

1. Tableau is closed.  
   Then knowledge base is unsatisfiable.
2. No unblocked selection extends the tableau.  
   Then knowledge base is satisfiable.
Example

- F ... Women
  h ... hasMother
  V ... Bird
  t ... Tweety
- knowledge base \{F \sqsubseteq \exists h.F, V(t)\}
- We want to show that Tweety is *not* a Women, i.e. that \(\neg F(t)\) is a logical consequence.
- It will not be possible to prove this, i.e. Tweety *can* be a Woman.

Example (ctd)

**TBox:** \(\neg F \sqcup \exists h.F\)

**Tableau:**

- \(V(t)\) (from knowledge base)
- \(F(t)\) (negated query in NNF)
- \((\neg F \sqcup \exists h.F)(t)\)
- \(\neg F(t)\) (\(\exists h.F\)(t))
- \(h(t,s)\)
- \(F(s)\)
- \((\neg F \sqcup \exists h.F)(s)\)
- \(\neg F(s)\) (\(\exists h.F\)(s))
  blocked by t
- s and t fall under F, \(\neg F \sqcup h.F, \exists h.F\)
- no other selection possible
Tableaux for OWL DL

• Basic ideas are the same.

• Need more complex blocking rules.

• Instance generation is not very efficient.

• Tableau with blocking is 2NExptime! → worse than necessary!

Tableaux inference engines

• Fact
  – http://www.cs.man.ac.uk/~horrocks/FaCT/
  – SHIQ

• Fact++
  – http://owl.man.ac.uk/factplusplus/
  – SHOIQ(D)

• Pellet
  – SHOIN(D)

• RacerPro
  – http://www.sts.tu-harburg.de/~r.f.moeller/racer/
  – SHIQ(D)
Acknowledgements

For the preparation of these slides, I did not hesitate to reuse any material which I found on the web or on my computer. Some of it is derived from slides by

- last years' ISWWW lecture / Steffen Staab et al.
- Sean Bechhofer, Manchester
- Ian Horrocks, Manchester
- Boris Motik, FZI Karlsruhe
- Alan Rector et al., Manchester (OWL Pizzas)
- Denny Vrandecic, AIFB Karlsruhe
- and possibly some other people for the cases where I couldn't trace the origin of my files …