AIFB

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

AIFB

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

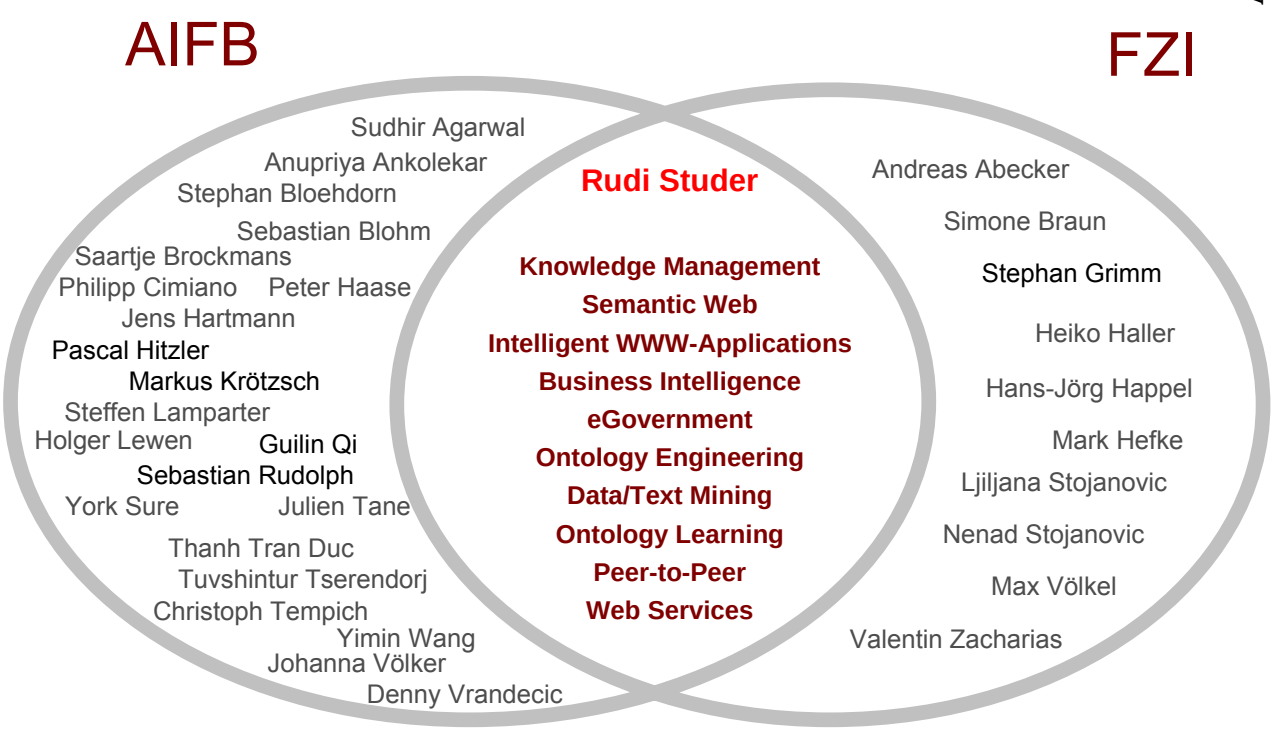
Semantic Karlsruhe

AIFB	FZI	ontoprise
Basic Research Application-oriented Research	Application-oriented Research Know-how Transfer Realizing new Scenarios	Application-oriented Research Product Development Innovative Solutions

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

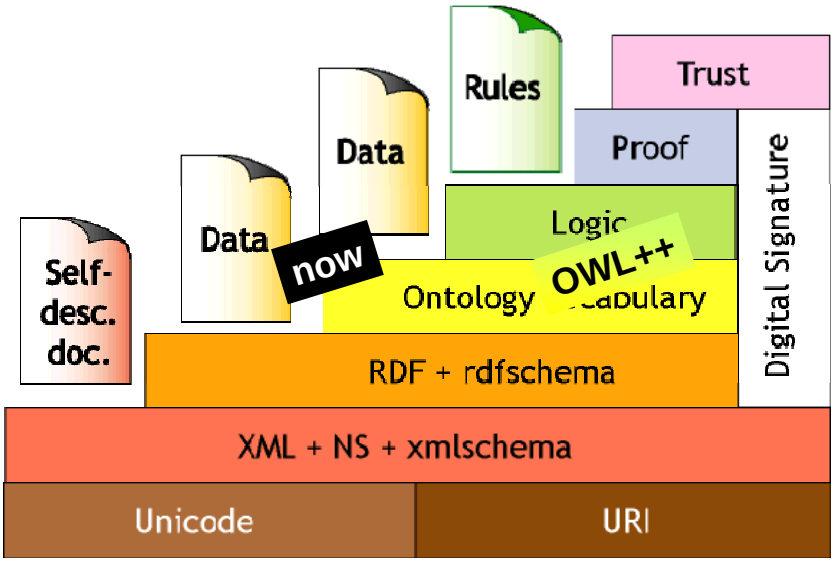


& ~40 people at Ontoprise

Partners and Projects

AIFB

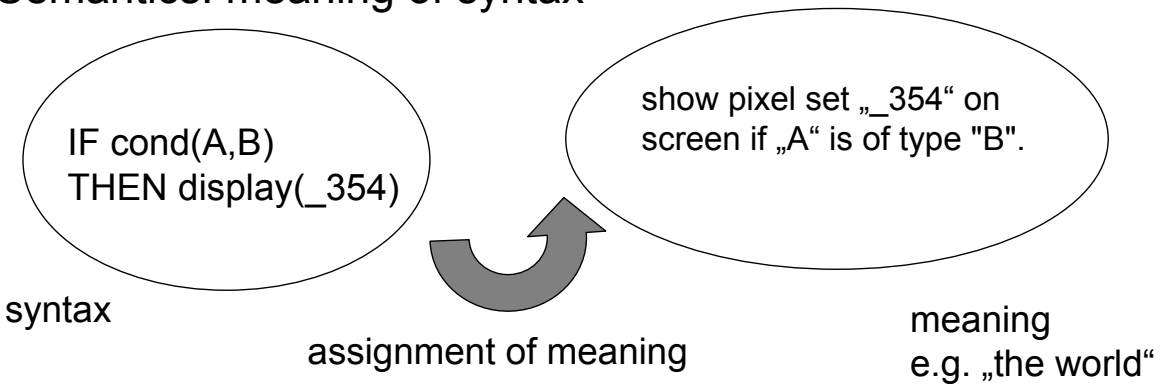
Semantic Web Layer Cake



What is Semantics?

Syntax: strings without meaning

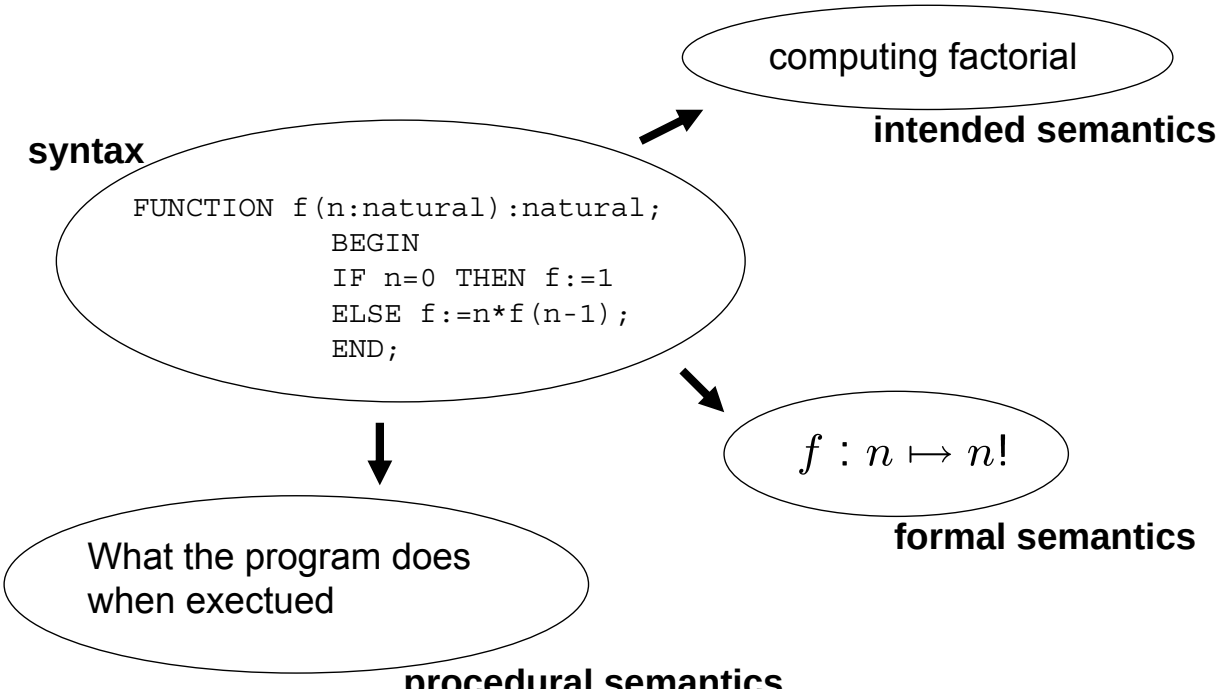
Semantics: meaning of syntax



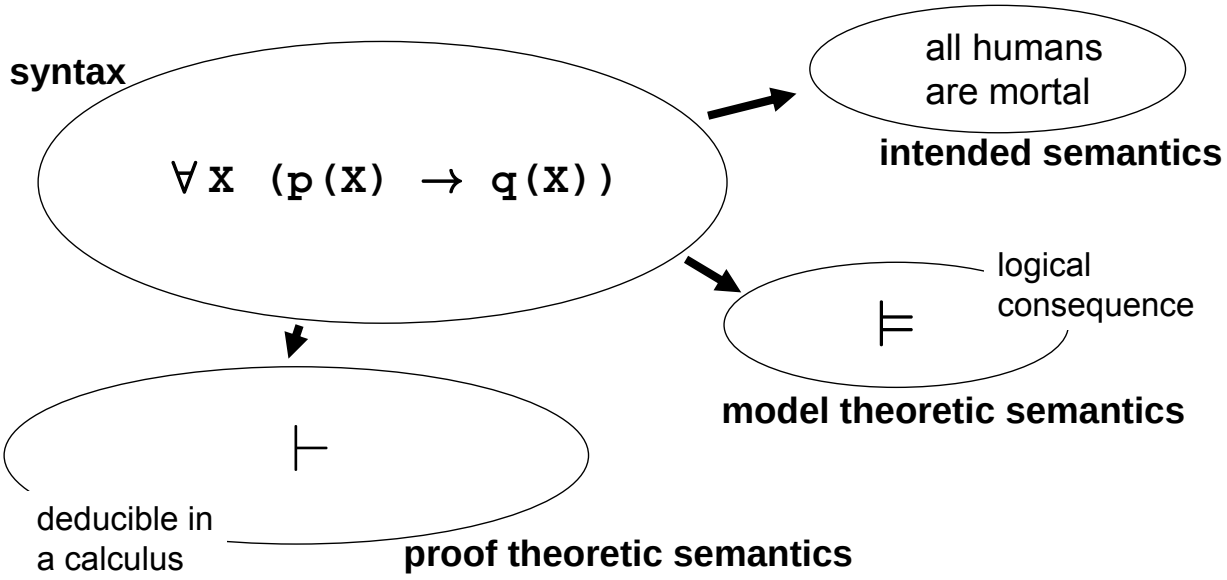
Why logic is so successful:

Semantics can be captured syntactically!

What is semantics? Example programming language



Semantics of logics/knowledge representation languages



Part II contents

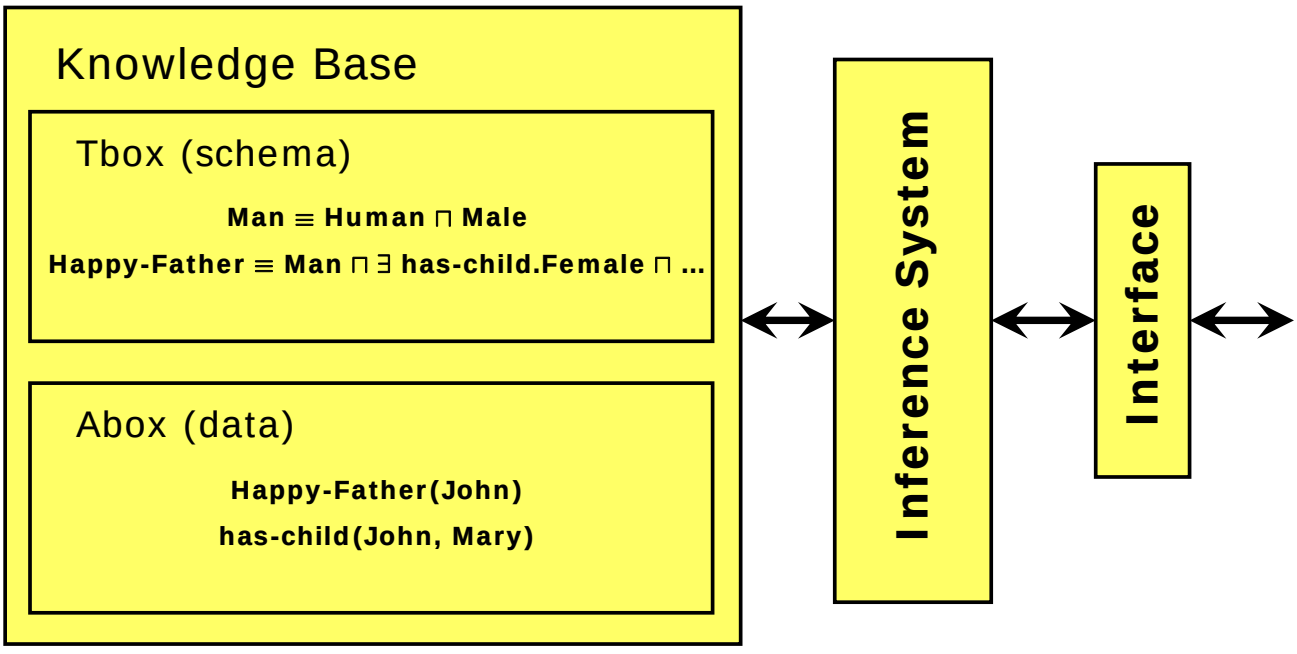
1. OWL Model-theoretic Semantics
   a. **Description Logics: ALC**
   b. OWL as SHOIN(D)
   c. OWL Examples
2. Proof Theory
   a. Reasoning as Satisfiability checking
   b. Tableaux Reasoning

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



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 smalles DL which is propositionally closed
  - Conjunction, disjunction, negation are constructors, written as ⊓, ⊔, ¬.
  - Quantifiers used only together with roles:

Man ⊓ ∃hasChild.Female ⊓ ∃hasChild.Male
⊓ ∀hasChild.(Rich ⊔ Happy)

Other DL language components

- E.g.
  - Number restrictions (cardinality constraints) for Roles:
    ≥3 hasChild, ≤1hasMother
  - Qualified number restrictions:
    ≥2 hasChild.Female, ≤1 hasParent.Male
  - Nominals (definition by extension): {Italy, France, Spain}
  - Concrete domains (datatypes): hasAge.(≥21)

  - Inverse roles: $\text{hasChild}^{-} \equiv \text{hasParent}$
  - Transitive roles: $\text{hasAncestor} \sqsubseteq^{+} \text{hasAncestor}$
  - Role composition: hasParent.hasBrother(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 $\sqcap$
- disjunction $\sqcup$
- negation $\neg$

- $\text{Professor} \sqsubseteq (\text{Person} \sqcap \text{Faculty}) \sqcup (\text{Person} \sqcap \neg\text{PhDStudent})$

$(\forall x)(\text{Professor}(x) \rightarrow ((\text{Person}(x) \wedge \text{Faculty}(x)) \vee (\text{Person}(x) \wedge \neg\text{PhDStudent}(x)))$

ALC: Quantifiers

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

- $\text{Exam} \sqsubseteq \exists \text{hasExaminer.Person}$
  $(\forall x)(\text{Exam}(x) \rightarrow (\exists y)(\text{hasExaminer}(x,y) \wedge \text{Person}(y)))$
  - corresponds to owl:someValuesFrom

Modelling in ALC

- owl:nothing: $\bot \equiv C \sqcap \neg C$
- owl:thing: $\top \equiv C \sqcup \neg C$
- owl:disjointWith: $C \sqcap D \equiv \bot$
  equivalently: $C \sqsubseteq \neg D$
- rdfs:range: $\top \sqsubseteq \forall R.C$
- rdfs:domain: $\exists R.\top \sqsubseteq C$

ALC: Syntax

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

  $C,D \rightarrow A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$
- 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 π (shown to the right).

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

$$\pi(C \sqsubseteq D) = (\forall x)(\pi_x(C) \rightarrow \pi_x(D))$$
$$\pi(C \equiv D) = (\forall x)(\pi_x(C) \leftrightarrow \pi_x(D))$$
$$\pi_x(A) = A(x)$$
$$\pi_x(\neg C) = \neg \pi_x(C)$$
$$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$$
$$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$$
$$\pi_x(\forall R.C) = (\forall y)(R(x,y) \rightarrow \pi_y(C))$$
$$\pi_x(\exists R.C) = (\exists y)(R(x,y) \wedge \pi_y(C))$$
$$\pi_y(A) = \boldsymbol{A(y)}$$
$$\pi_y(\neg C) = \neg \pi_y(C)$$
$$\pi_y(C \sqcap D) = \pi_y(C) \wedge \pi_y(D)$$
$$\pi_y(C \sqcup D) = \pi_y(C) \vee \pi_y(D)$$
$$\pi_y(\forall R.C) = (\forall x)(R(y,x) \rightarrow \pi_x(C))$$
$$\pi_y(\exists R.C) = (\exists x)(R(y,x) \wedge \pi_x(C))$$

DL knowledge bases

- DL knowledge bases consist of two parts:
  - TBox: Axioms containing schema knowledge:
    - **HappyFather ≡ Man ⊓ ∃hasChild.Female ⊓ …**
    - **Elephant ⊑ Animal ⊓ Large ⊓ Grey**
    - **transitive(hasAncestor)**
  - Abox: Axioms describing data:
    - **HappyFather(John)**
    - **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 $\sqsubseteq \exists$parentOf.Human

Orphan $\equiv$ Human $\sqcap \neg\exists$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≠b
  - FOL: ¬(a=b)

OWL as SHOIN(D): nominals

Nominals

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

OWL as SHOIN(D): number restrictions

number restrictions need equality predicate

```
<owl:Class rdf:about="#Exam">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasExaminer"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxcardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

An exam may have *at most two* examiners.

- DL: Exam $\sqsubseteq \leq 2$ hasExaminer
- In FOL: (E… Exam, h…hasExaminer)

  $(\forall x)(E(x) \rightarrow \neg(\exists x1)(\exists x2)(\exists x3)(x1 \neq x2 \wedge x2 \neq x3 \wedge x1 \neq x3 \wedge h(x,x1) \wedge h(x,x2) \wedge h(x,x3)))$

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: $(\forall x)(\forall y)(\forall z)(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$
- Symmetry: $R \equiv R^-$
- Functionality: $\top \sqsubseteq \leq 1\ R$
- Inverse Functionality: $\top \sqsubseteq \leq 1\ R^-$

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)

AIFB

Overview syntax for DLs (without datatypes)

	Concepts	
ALC	Atomic	A, B
ALC	Not	¬C
ALC	And	C ⊓ D
ALC	Or	C ⊔ D
ALC	Exists	∃R.C
ALC	For all	∀R.C
Q (N)	At least	≥n R.C (≥n R)
Q (N)	At most	≤n R.C (≤n R)
O	Nominal	$\{i_1,\ldots,i_n\}$

	Roles	
I	Atomic	R
I	Inverse	R^-

Ontology (=Knowledge Base)

Concept Axioms (TBox)	
Subclass	C ⊑ D
Equivalent	C ≡ D

	Role Axioms (RBox)	
H	Subrole	R ⊑ S
S	Transitivity	Trans(S)

Assertional Axioms (ABox)	
Instance	C(a)
Role	R(a,b)
Same	a = b
Different	a ≠ b

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

AIFB

OWL DL as DL: overview class constructors

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

nesting is allowed:

Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)

AIFB

OWL DL as DL: axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

- General Class Inclusion ($\sqsubseteq$) suffices:
  C $\equiv$ D gdw (C $\sqsubseteq$ D und D $\sqsubseteq$ C)
- Equivalences
  C $\equiv$ D $\Leftrightarrow$ ($\forall$x) (C(x) $\leftrightarrow$ D(x))
  C $\sqsubseteq$ D $\Leftrightarrow$ ($\forall$x) (C(x) $\rightarrow$ D(x))

AIFB

Computational complexity (worst-case)

OWL variant	data complexity	combined complexity
OWL Full	undecidable	undecidable
OWL DL	unknown	NExptime
OWL DL without nominals	NP (IJCAI 2005)	Exptime
OWL Lite	NP	Exptime

data complexity: complexity w.r.t. ABox size

combined complexity: complexity w.r.t. ABox and TBox size

AIFB

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

AIFB

Examples (abstract syntax)

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

$$\mathbf{bus\_driver \equiv person \sqcap \exists drives.bus}$$

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

Class(a:bus partial a:vehicle)

$$\mathbf{driver \equiv person \sqcap \exists drives.vehicle}$$

$$\mathbf{bus \sqsubseteq vehicle}$$

- 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

driver ≡ person ⊓ ∃drives.vehicle

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

driver ⊑ adult

Class(a:driver partial a:adult)

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

grownup ≡ adult ⊓ 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

AIFB

Important inference problems

- global consistency of knowledge base — KB ⊨ **false**?
  - Is knowledge base reasonable?
- class consistency — C ≡ ⊥?
  - Is class c well-modeled?
- subsumption — C ⊑ D?
  - structuring the knowledge base
- class equivalence — C ≡ D?
  - Are they actually the same?
- class disjointness — C ⊓ D = ⊥?
  - 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.

AIFB

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.

AIFB

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!

AIFB

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

resulting knowledge base: NNF(W)

negation normal form of W.

Negation only occurs directly in front of atomic classes.

$NNF(C) = C$, if C atomic

$NNF(\neg C) = \neg C$, if C atomic

$NNF(\neg\neg C) = NNF(C)$

$NNF(C \sqcup D) = NNF(C) \sqcup NNF(D)$

$NNF(C \sqcap D) = NNF(C) \sqcap NNF(D)$

$NNF(\neg(C \sqcup D)) = NNF(\neg C) \sqcap NNF(\neg D)$

$NNF(\neg(C \sqcap D)) = NNF(\neg C) \sqcup NNF(\neg D)$

$NNF(\forall R.C) = \forall R.NNF(C)$

$NNF(\exists R.C) = \exists R.NNF(C)$

$NNF(\neg\forall R.C) = \exists R.NNF(\neg C)$

$NNF(\neg\exists R.C) = \forall R.NNF(\neg C)$

W and NNF(W) are logically equivalent.

negation normal form: example

$P \sqsubseteq (E \sqcap U) \sqcup \neg(\forall R.E \sqcup D)$.

In negation normal form:

$\neg P \sqcup (E \sqcap U) \sqcup (\exists R.\neg E \sqcap \neg D)$.

ALC Tableaux: contents

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

Naive tableau algorithm

reduce to finding an inconsistency

Idea:

- Given a knowledge base W.
- Generate consequences of the form C(a) and ¬C(a), until a contradiction is found.

Simple example

C(a)
(¬C ⊓ D)(a)

¬C(a) is logical consequence:

2nd formula in FOL: ¬C(a)∧ D(a)
hence ¬C(a)

Contradiction has been found.

Another example

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

Derive logical consequences:

C(a)

$\neg D(a)$

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

Now we consider two cases

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

Splitting of the tableau into two *brances*

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

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

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

Always select only such elments 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 ⊑ (E ⊓ U) ⊔ (E ⊓ ¬D)
  Is P ⊑ E a logical consequence?

- Knowledge base (with query) in NNF:
  {¬P ⊔ (E ⊓ U) ⊔ (E ⊓ ¬D), (P ⊓ ¬E)(a)}

AIFB

Example (ctd)

TBox: $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$

Tableau:

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

$P(a)$

$\neg E(a)$

$(\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

$\neg P(a)$ $((E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

$(E \sqcap U)(a)$ $(E \sqcap \neg D)(a)$

$E(a)$ $E(a)$

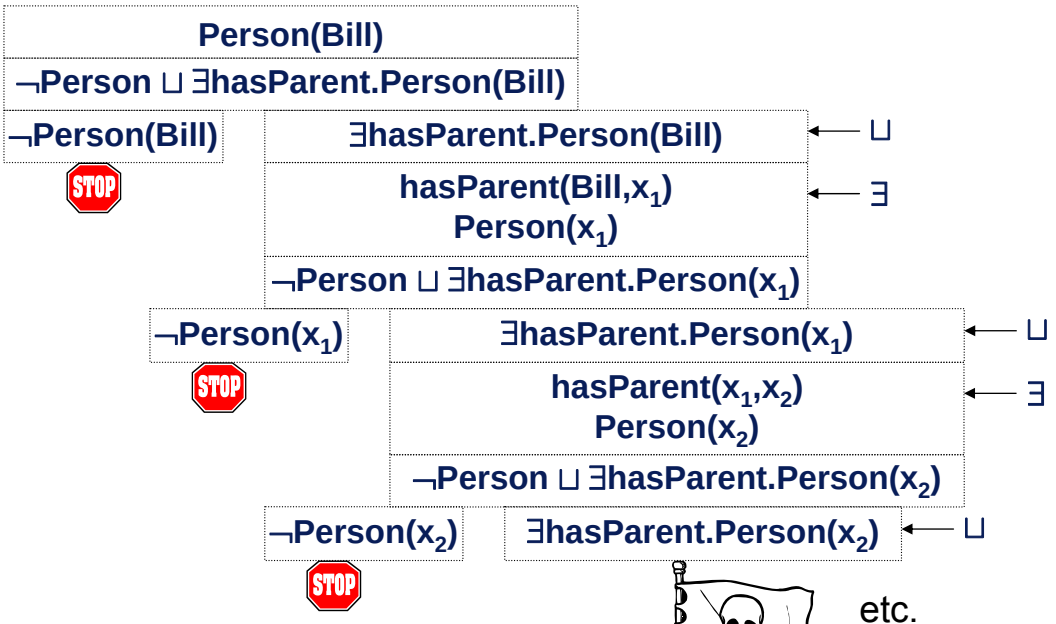
$U(a)$ $\neg D(a)$

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

AIFB

The termination problem

- Single Axiom: $\neg$Person $\sqcup$ $\exists$hasParent.Person
  we want to show: $\neg$Person(Bill)



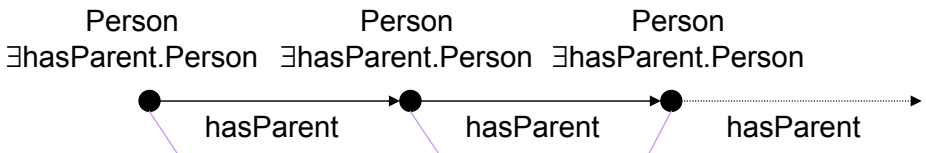
Problem occurs due to existential quantification (and minCardinality)

ALC Tableaux: contents

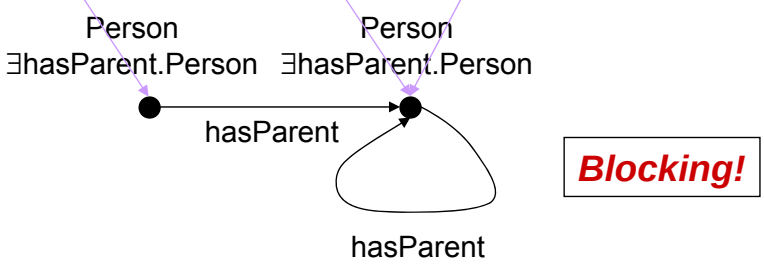
- Transformation to negation normal form
- Naive tableau algorithm
- **Tableau algorithm with Blocking**

Solving the termination problem

- The following happened:



- But why not do it this way:



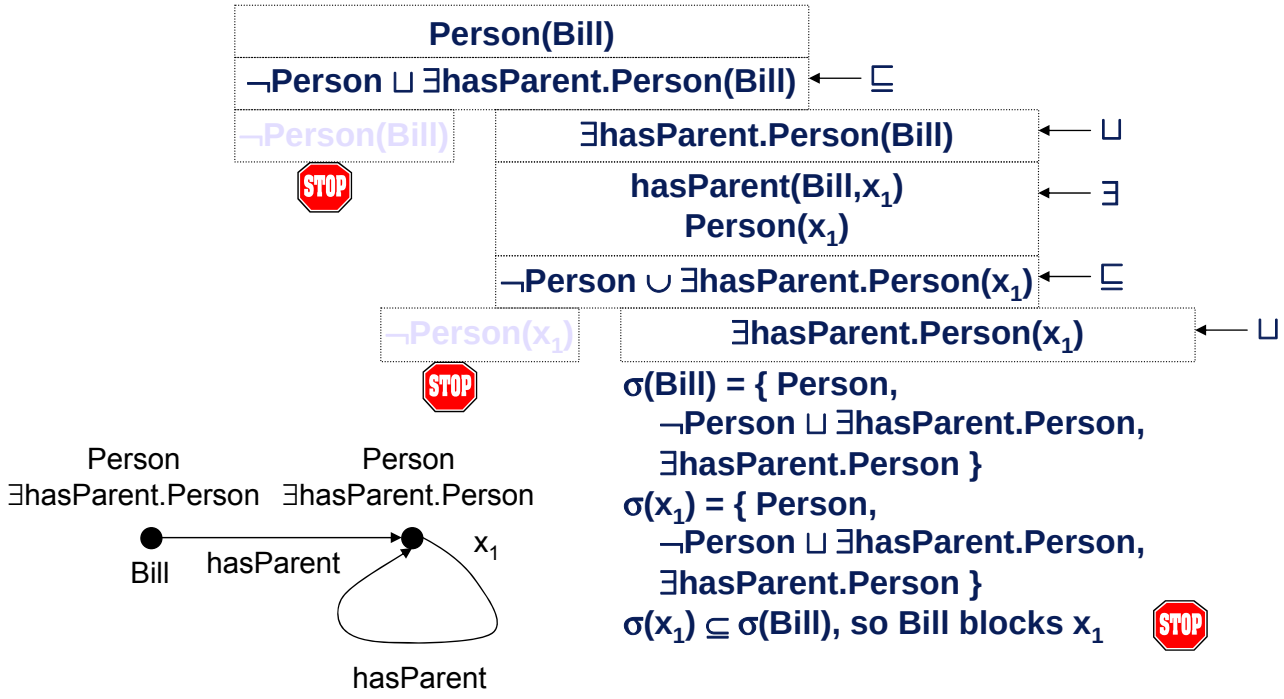
I.e. reuse old nodes!
Formal proof required that this suffices!

AIFB

Tableau with Blocking

- Single Axiom: ¬Person ⊔ ∃hasParent.Person

  We want to show: ¬Person(Bill)



AIFB

Blocking

The selection of (∃R.C)(a) in branch A is *blocked* if there is an individual b with

{C | C(a) ∈ A} ⊆ {C | C(b) ∈ 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 ⊑ ∃h.F, V(t)}
- We want to show that Tweety is *not* a Women, i.e. that ¬F(t) is a logical consequence.
- It will not be possible to prove this, i.e. Tweety *can* be a Woman.

Example (ctd)

TBox: ¬F⊔ ∃h.F

Tableau:

V(t) (from knowledge base)

F(t) (negated query in NNF)

(¬F⊔ ∃h.F)(t)

¬F(t) (∃h.F)(t)

h(t,s)

F(s)

(¬F⊔ ∃h.F)(s)

¬F(s) (∃h.F)(s)

blocked by t

s and t fall under

F, ¬F ⊔ h.F, ∃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!
  $\rightarrow$ 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
  - http://www.mindswap.org/2003/pellet/index.shtml
  - SHOIN(D)
- RacerPro
  - http://www.sts.tu-harburg.de/~r.f.moeller/racer/
  - SHIQ(D)

Acknowledgements

For the preperation 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 …