

Intelligente Systeme im World Wide Web

Web Ontology Language OWL

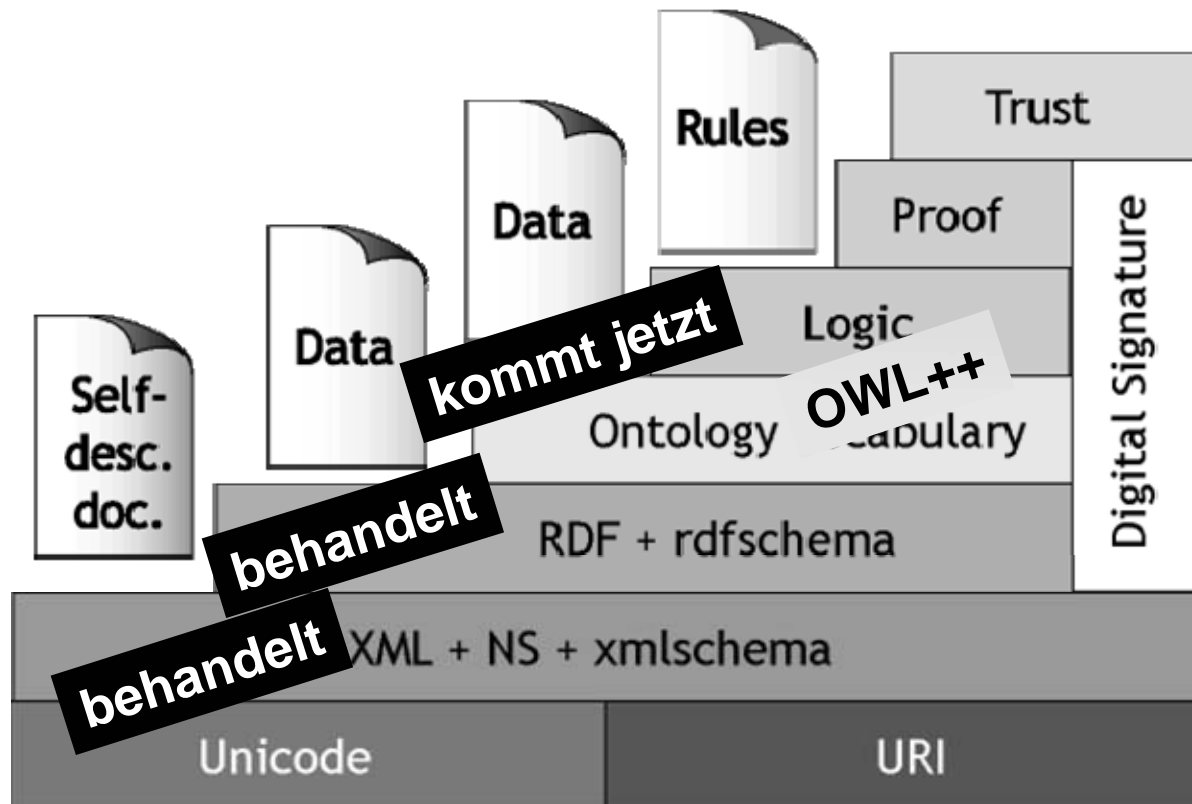
Folien zur Vorlesung im Sommersemester 2005

Pascal Hitzler & York Sure

Institut für angewandte Informatik und Formale
Beschreibungsverfahren (AIFB)

Universität Karlsruhe (TH)

Die Semantic Web Schichttorte



Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. Beweistheoretische Semantik
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

Ontologie – philosophisch

- Ontologie einer Theorie A: notwendige Bedingungen ans Sein oder das Seiende für die Gültigkeit der Theorie A
- Ontologie ist die Lehre vom Sein qua Sein
- Fragestellungen:
 - Was ist das ‚Sein‘? Was bedeutet ‚sein‘? (Intensionale Definition)
 - Was hat ‚Sein‘? (Extensionale Definition)
 - Wie unterteilt sich das Seiende?

Aristoteles (Sokrates), Thomas von Aquin, Descartes, Kant, Hegel, Wittgenstein, Heidegger, Quine, ...

Ontologie – informatisch

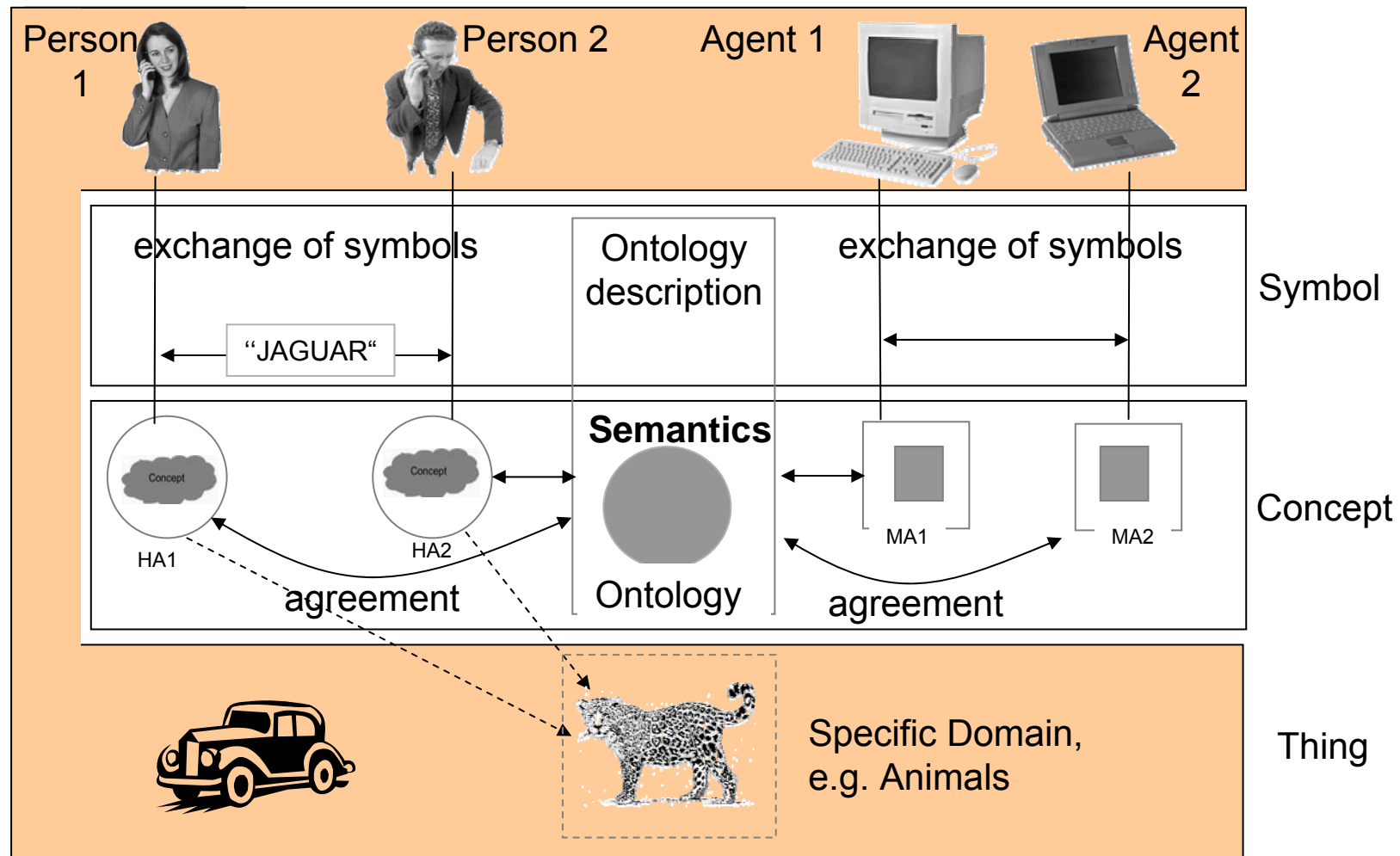
Gruber 93:

An Ontology is a

formal specification
of a shared
conceptualization
of a domain of interest

⇒ machine-understandable
⇒ group of people
⇒ about concepts
⇒ between general
description and individual
use

Ontologie und Kommunikation



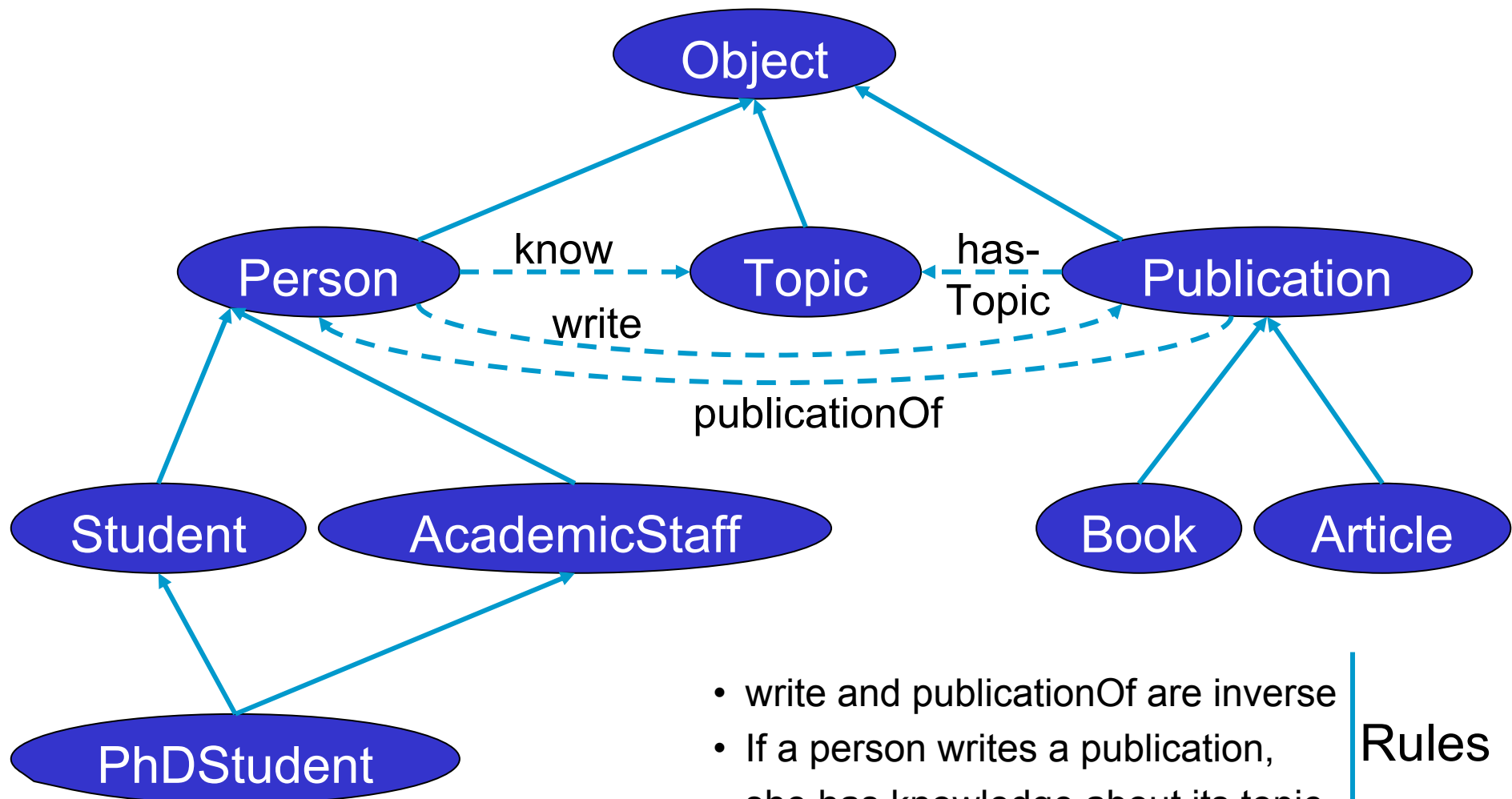
Ontologie – praktisch. Einige Anforderungen

- Begriffshierarchie (Taxonomie): Klassen, Begriffe
- binäre Relationen zwischen Begriffen: Properties, Roles
- Vererbung: is-a etc.

- Datentypen (z.B. Zahlen): concrete domains
- Eigenschaften von Relationen (z.B. range, transitive)
- logische Ausdrucksmittel

- **Semantik!**

Einfache Ontologie: Beispiel



- write and publicationOf are inverse
- If a person writes a publication, she has knowledge about its topic

Rules

Inhalt

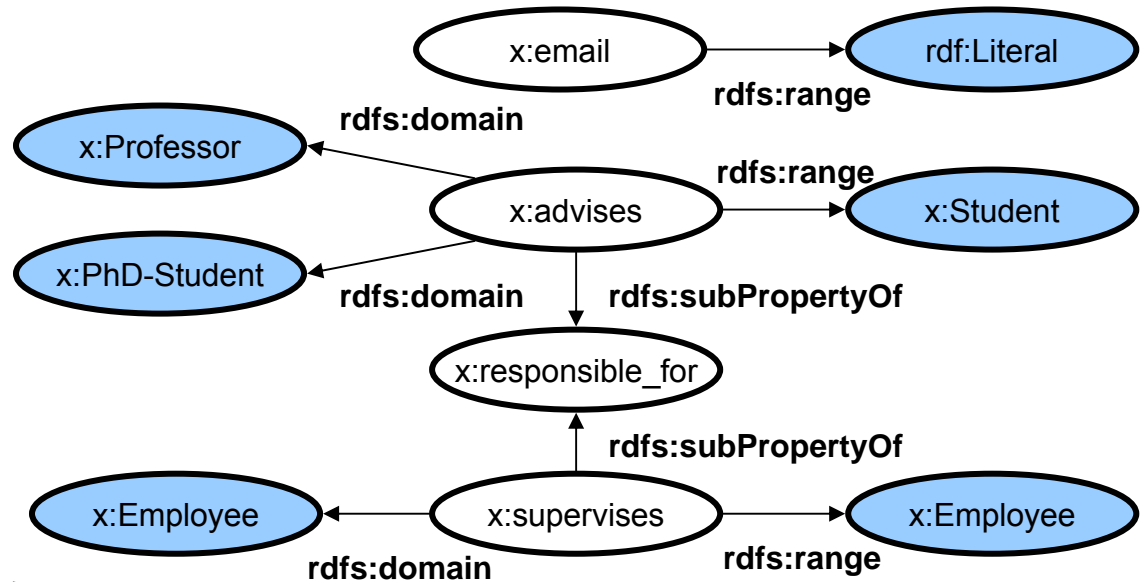
- I. Was ist „Ontologie“?
- II. **RDF(S) und FOL als Ontologiesprachen**
- III. **OWL**
 1. **Syntax und Modelltheoretische Semantik**
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. **Beweistheoretische Semantik**
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

FOL als Ontologiesprache

- Warum nicht einfach FOL für Ontologien nehmen?
 - FOL kann alles
 - Assembler auch!**
 - FOL ist
 - sehr ausdrucksstark
 - deshalb unhandlich bei der Modellierung
 - schlecht geeignet um Konsens bei der Modellierung zu finden
 - Beweistheoretisch sehr komplex
- ⇒ Suche geeignetes Fragment von FOL

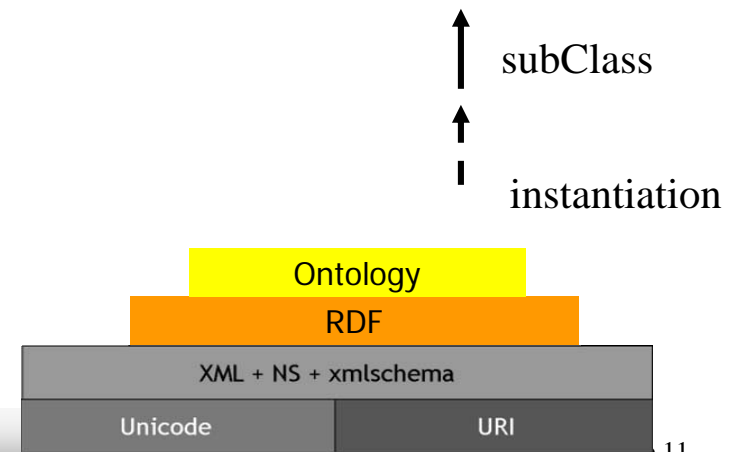
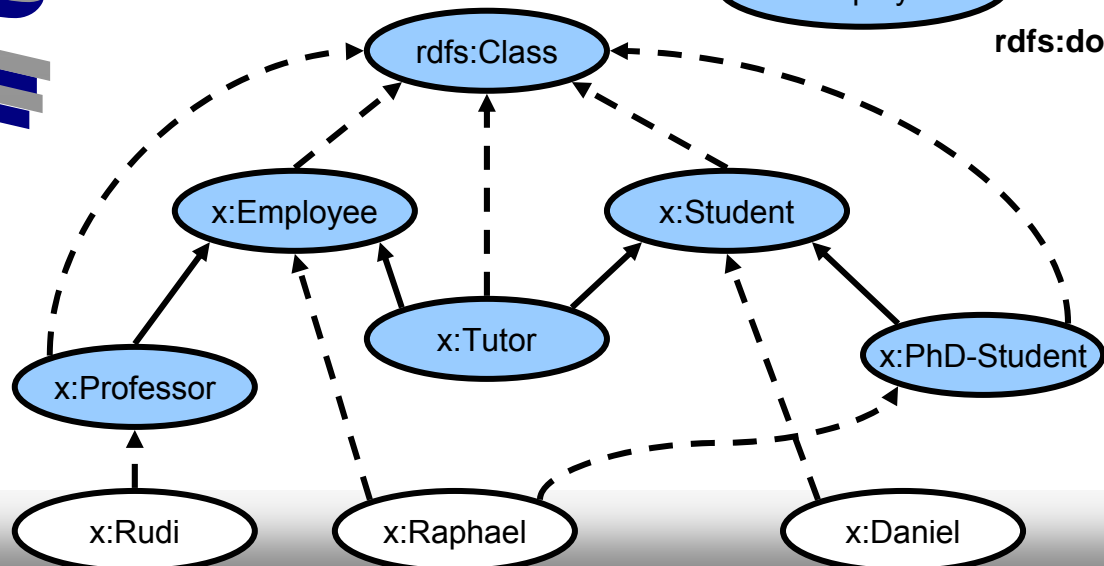
RDF Schema – Einfache Ontologien

Declaration of properties



Declaration of classes

RDF



Formal Model of RDF III

- The entailment process terminates on any finite RDF graph
 → **only finitely many possible triples can be formed from a given finite vocabulary.**

- Example Graph (Single Triple):

[foo bar baz].

- Closure (for mentioned rules only !):

1.	foo bar baz .	Source
2.	foo rdf:type rdfs:Resource .	Rule 4a on (1)
3.	baz rdf:type rdfs:Resource .	Rule 4a on (1)
4.	bar rdf:type rdf:Property .	Rule 1 on (1)
5.	rdf:type rdf:type rdf:Property .	Rule 1 on (4)
6.	rdf:type rdfs:subPropertyOf rdf:type.	Rule 5b on (5)
7.	bar rdfs:subPropertyOf bar.	Rule 5b on (4)
8.	rdfs:subPropertyOf rdf:type rdf:Property	Rule 1 on (6)

RDF Schema als Ontologiesprache

- Geeignet für einfache Ontologien
 - Für komplexere Modellierungen ungeeignet
 - → „Need for Expressivity!“

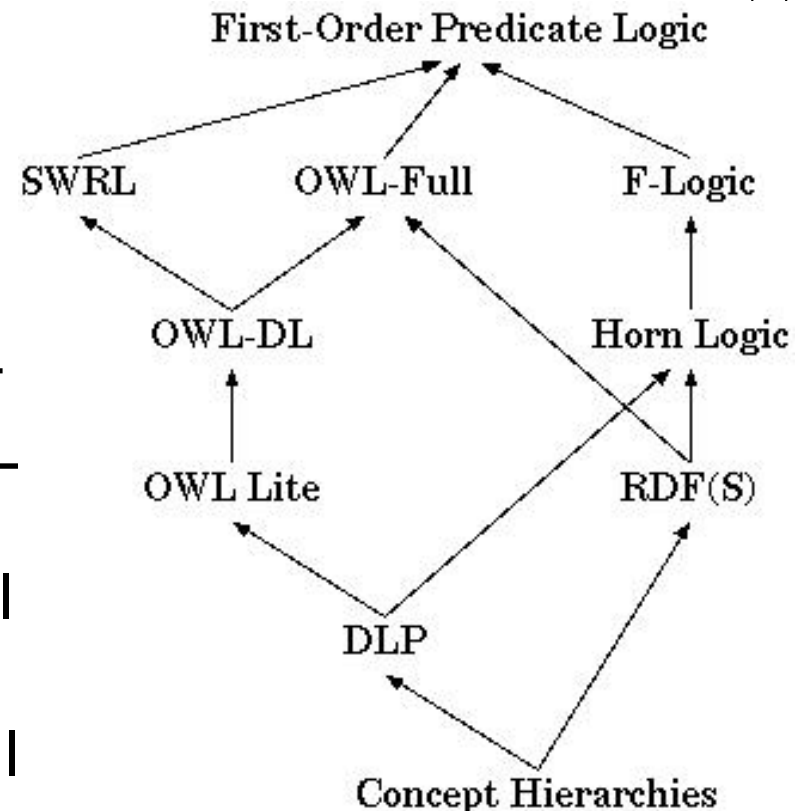
 - Mächtigere Sprachen:
 - OWL
 - F-Logic
 - Regelerweiterungen von OWL
- } werden wir behandeln

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. **OWL**
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. Beweistheoretische Semantik
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

OWL – Allgemeines

- W3C Recommendation seit 2004
- Semantisches Fragment von FOL
- Drei Varianten:
 $\text{OWL Lite} \subseteq \text{OWL DL} \subseteq \text{OWL Full}$
- Keine Reifikation in OWL DL
 RDFS ist Fragment von OWL Full
- OWL DL ist entscheidbar
- OWL DL = SHOIN(D) (Beschreibungslogik)
- W3C-Dokumente (Vorlesungswebseite) enthalten Details, die hier nicht alle angesprochen werden können.



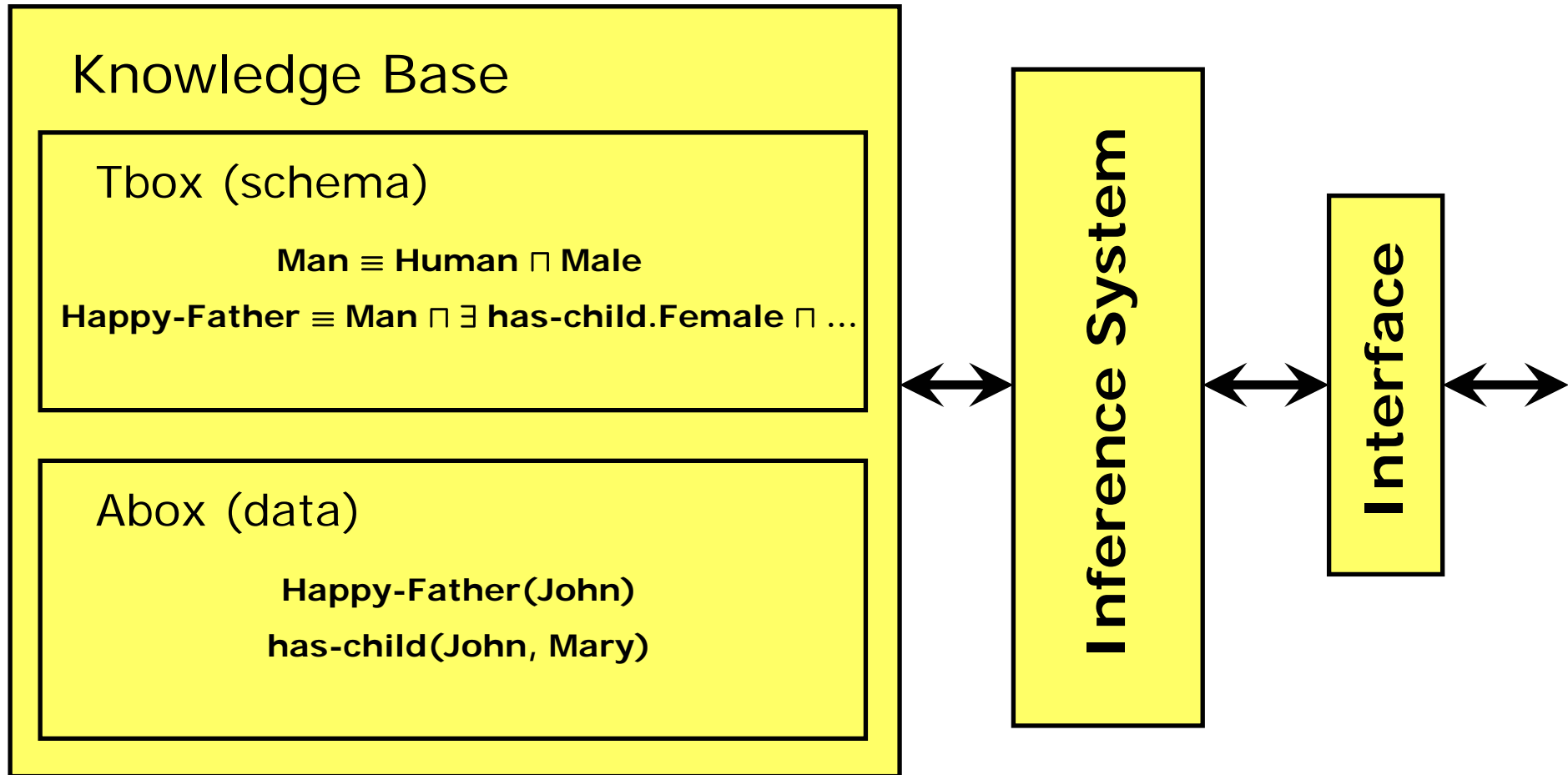
Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. **Syntax und Modelltheoretische Semantik**
 - a. **Beschreibungslogiken: SHOIN(D)**
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. **Beweistheoretische Semantik**
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

Beschreibungslogiken (Description Logics, DLs)

- Fragmente von FOL
 - meist entscheidbar
 - ausdrucksstark
 - entwickelt aus semantischen Netzwerken
 - enge Beziehungen zu Modallogiken
-
- W3C Standard OWL DL ist die Beschreibungslogik SHOIN(D)

Allgemeine DL Architektur



DLs – allgemeiner Aufbau

- DLs sind eine Familie logikbasierter Formalismen zur Wissensrepräsentation
- Spezielle Sprachen v.a. charakterisiert durch:
 - Konstruktoren für komplexe Konzepte und Rollen aus einfacheren.
 - Menge von Axiomen um Fakten über Konzepte, Rollen und Individuen auszudrücken.
- ALC ist die kleinste DL, die aussagenlogisch abgeschlossen ist
 - \wedge, \vee, \neg sind Konstruktoren, geschrieben \sqcap, \sqcup, \neg .
 - Quantoren schränken Rollenbereiche ein:

Man $\sqcap \exists \text{hasChild.Female} \sqcap \exists \text{hasChild.Male}$
 $\sqcap \forall \text{hasChild.}(\text{Rich} \sqcup \text{Happy})$

Weitere DL Konzept- und Rollenkonstruktoren

- Andere Konstruktoren sind z.B.
 - Number restrictions (cardinality constraints) für Rollen:
 - ≥ 3 hasChild, ≤ 1 hasMother
 - Qualified number restrictions:
 - ≥ 2 hasChild.Female, ≤ 1 hasParent.Male
 - Nominals (definition by extension): {Italy, France, Spain}
 - Concrete domains (datatypes): hasAge.(≥ 21)

 - Inverse roles: hasChild⁻ \equiv hasParent
 - Transitive roles: hasAncestor* (descendant)
 - Role composition: hasParent.hasBrother(uncle)

DL Wissensbasen

- DL Wissensbasen bestehen normalerweise aus 2 Teilen:
 - TBox: Axiome, die die Struktur der zu modellierenden Domäne beschreiben (konzeptionelles Schema):
 - **HappyFather** \equiv **Man** \sqcap \exists **hasChild.Female** \sqcap ...
 - **Elephant** \sqsubseteq **Animal** \sqcap **Large** \sqcap **Grey**
 - **transitive(hasAncestor)**
 - ABox: Axiome, die konkrete Situationen (Daten) beschreiben:
 - **HappyFather(John)**
 - **hasChild(John, Mary)**
- Unterscheidung TBox/ABox hat keine logische Bedeutung ... ist aber konzeptionell einfacher.

Syntax für DLs (ohne concrete domains)

Hitzler & Sure, 2005



Concepts		
ALC	Atomic	A, B
	Not	$\neg C$
	And	$C \sqcap D$
	Or	$C \sqcup D$
	Exists	$\exists R.C$
	For all	$\forall R.C$
Q(N)	At least	$\geq n R.C$ ($\geq n R$)
	At most	$\leq n R.C$ ($\leq n R$)
O	Nominal	$\{i_1, \dots, i_n\}$

Roles		
—	Atomic	R
	Inverse	R^-

Ontology (=Knowledge Base)	
Concept Axioms (TBox)	
Subclass	$C \sqsubseteq D$
Equivalent	$C \equiv D$
Role Axioms (RBox)	
\sqsubseteq Subrole	$R \sqsubseteq S$
\mathcal{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)

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. **OWL als SHOIN(D)**
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. Beweistheoretische Semantik
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

OWL DL als DL: Klassenkonstruktoren

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male	$C_1(x) \wedge \dots \wedge C_n(x)$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer	$C_1(x) \vee \dots \vee C_n(x)$
complementOf	$\neg C$	\neg Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} \sqcup {mary}	$x = x_1 \vee \dots \vee x = x_n$
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor	$\forall y.P(x, y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	\exists hasChild.Lawyer	$\exists y.P(x, y) \wedge C(y)$
maxCardinality	$\leq_n P$	≤ 1 hasChild	$\exists^{\leq n} y.P(x, y)$
minCardinality	$\geq_n P$	≥ 2 hasChild	$\exists^{\geq n} y.P(x, y)$

Beliebig komplexes Schachteln von Konstruktoren erlaubt:

Person $\sqcap \forall$ hasChild.(Doctor $\sqcup \exists$ hasChild.Doctor)

■ OWL DL als DL: Axiome

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	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1$ hasSSN ⁻

- General Class Inclusion (\sqsubseteq) genügt:
 $C \equiv D \text{ gdw } (C \sqsubseteq D \text{ und } D \sqsubseteq C)$
- Offensichtliche FOL-Äquivalenzen
 $C \equiv D \Leftrightarrow (\forall x) (C(x) \leftrightarrow D(x))$
 $C \sqsubseteq D \Leftrightarrow (\forall x) (C(x) \rightarrow D(x))$

OWL/DL Beispiel

Terminologisches Wissen (*TBox*):

Human $\sqsubseteq \exists \text{parentOf.Human}$

Orphan $\equiv \text{Human} \sqcap \neg \exists \text{childOf.Alive}$

Wissen um Individuen (*ABox*):

Orphan(harrypotter)

ParentOf(jamespotter,harrypotter)

Semantik und logische Konsequenzen klar, da
übersetzbar nach FOL.

Modelltheoretische Semantik – direkt

Concepts	
A	Subset of Δ^I
$\neg C$	$\Delta^I \setminus C^I$
$C \cap D$	$\{x \mid x \in C^I \text{ und } x \in D^I\}$
$C \sqcup D$	$\{x \mid x \in C^I \text{ oder } x \in D^I\}$
$\exists R.C$	$\{x \mid (x, y) \in R^I \text{ und } y \in C^I\}$
$\forall R.C$	$\{x \mid \text{wenn } (x, y) \in R^I \text{ dann } y \in C^I\}$
$\geq n R.C$	$\{x \mid \#\{(x, y) \in R^I \text{ und } y \in C^I\} \geq n\}$
$\leq n R.C$	$\{x \mid \#\{(x, y) \in R^I \text{ und } y \in C^I\} \leq n\}$
$\{i_1, \dots, i_n\}$	$\{i_1^I, \dots, i_n^I\}$

Roles	
R	Subset of $\Delta \times \Delta$
R^-	$\{(y, x) \mid (x, y) \in R^I\}$

Ontology (=Knowledge Base)

Concept Axioms (TBox)

$C \sqsubseteq D$	$C^I \subseteq D^I$
$C \equiv D$	$C^I \equiv D^I$

Role Axioms (manchmal: RBox)

$R \sqsubseteq S$	$R^I \subseteq S^I$
-------------------	---------------------

Assertional Axioms (ABox)

$C(a)$	$a^I \in C^I$
$R(a, b)$	$(a^I, b^I) \in R^I$
$a = b$	$a^I = b^I$
$a \neq b$	$a^I \neq b^I$

Concrete Domains

- Strings und Integers (von Standard verlangt)
- Andere Datentypen können auch unterstützt werden.
- Beschränkung auf entscheidbare Prädikate über der Concrete Domain.

- Jede Concrete Domain muss im Rahmen eines OWL-Systems separat implementiert werden.
Analogie: built-ins (aber nicht prozedural).

OWL Lite

- Einfaches Fragment, bessere Komplexität.
- Nicht verwendet werden dürfen:
 - owl:unionOf
 - owl:complementOf
 - owl:oneOf
 - owl:hasValue
 - owl:disjointWith
- Eingeschränkt verwendet werden dürfen:
 - owl:intersectionOf
 - owl:minCardinality
 - owl:maxCardinality
 - owl:cardinality

OWL Full

- ist OWL DL vereinigt mit RDFS
- RDF ist i.A. in OWL Full, nicht in OWL DL
- Intuition:
 - OWL Full erlaubt reification.
 - OWL Full ist kein “schönes“ Fragment von FOL
 - OWL Full ist nicht entscheidbar.

Reification – Beispiel

Manchmal möchte man z.B. Aussagen über
Klassen*namen* machen:

Klasse: father.

(konkrete) Rollen: `germanClassName(father, „Vater“)`
 `frenchClassName(father, „père“)`
 `englishClassName(father, „father“)`

→ in OWL DL nicht ausdrückbar!

Komplexitäten (worst-case)

OWL Variante	Datenkomplexität	Kombinierte Komplexität
OWL Full	unentscheidbar	unentscheidbar
OWL DL	unbekannt	NExptime
OWL DL ohne Nominals	NP (neues Resultat IJCAI 2005!)	Exptime
OWL Lite	NP	Exptime

Datenkomplexität: nur bezüglich ABox

Kombinierte Komplexität: bezüglich ABox und TBox

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. **Serialisierungen**
 - d. Wissensmodellierung in OWL
 2. Beweistheoretische Semantik
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

Serialisierungen/verschiedene Syntax

- OWL RDF Syntax W3C offiziell!
- OWL Abstract Syntax W3C offiziell!
siehe nächster Abschnitt

- OWL XML Syntax W3C Dokument

- DL Schreibweise sehr verbreitet für
wissenschaftliche Arbeiten
- FOL Schreibweise unüblich

- Für die Implementierung und das Testen von KAON2
wurde z.B. eine funktionale Schreibweise entwickelt.

Lisp-artig

Beispiel: RDF Syntax

Person $\sqcap \forall \text{hasChild} . (\text{Doctor} \sqcup \exists \text{hasChild} . \text{Doctor})$:

```

<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild" />
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor" />
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild" />
            <owl:someValuesFrom rdf:resource="#Doctor" />
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

```

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. **Wissensmodellierung in OWL**
 2. Beweistheoretische Semantik
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

Wissensmodellierung in OWL

Beispielontologie und –schlussfolgerungen unter
<http://owl.man.ac.uk/2003/why/latest/#2>

- Dient auch als Beispiel für OWL Abstract Syntax.

```
Namespace(a = <http://cohse.semanticweb.org/ontologies/people#>)
```

```
Ontology(
```

```
  ObjectProperty(a:drives)
```

```
  ObjectProperty(a:eaten_by)
```

```
  ObjectProperty(a:eats inverseOf(a:eaten_by) domain(a:animal))
```

```
  ...
```

```
  Class(a:adult partial annotation(rdfs:comment "Things that are adult."))
```

```
  Class(a:animal partial restriction(a:eats someValuesFrom (owl:Thing)))
```

```
  Class(a:animal_lover complete intersectionOf(restriction(a:has_pet  
    minCardinality(3)) a:person))
```

```
...)
```

Wissensmodellierung: Beispiele

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

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) **driver \equiv person \sqcap \exists drives.vehicle**

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.

Wissensmodellierung: Beispiele

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

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

Class(a:driver partial a:adult)

$$\text{driver} \sqsubseteq \text{adult}$$

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

$$\text{grownup} \equiv \text{adult} \sqcap \text{person}$$

- 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 (e.g. 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.

$$\exists \text{partof. animal} \sqcup \text{animal} \not\equiv \text{plant} \sqcup \exists \text{partof. plant}$$

Wissensmodellierung: Beispiele

Class(a:cow partial a:vegetarian)

DisjointClasses(unionOf(restriction(a:part_of someValuesFrom (a:animal)) a:animal) unionOf(a:plant restriction(a:part_of someValuesFrom (a:plant))))

Class(a:vegetarian complete intersectionOf(restriction(a:eats allValuesFrom (complementOf(restriction(a:part_of someValuesFrom (a:animal)))))) restriction(a:eats allValuesFrom (complementOf(a:animal))) a:animal))

Class(a:mad_cow complete intersectionOf(a:cow restriction(a:eats someValuesFrom (intersectionOf(restriction(a:part_of someValuesFrom (a:sheep)) a:brain))))))

Class(a:sheep partial a:animal restriction(a:eats allValuesFrom (a:grass)))

- Cows are naturally vegetarians
- A mad cow is one that has been eating sheeps brains
- Sheep are animals

Thus a mad cow has been eating part of an animal, which is inconsistent with the definition of a vegetarian

Wissensmodellierung: Beispiele

```
Individual(a:Walt type(a:person) value(a:has_pet a:Huey)
  value(a:has_pet a:Louie) value(a:has_pet a:Dewey))
```

```
Individual(a:Huey type(a:duck))
```

```
Individual(a:Dewey type(a:duck))
```

```
Individual(a:Louie type(a:duck))
```

```
DifferentIndividuals(a:Huey a:Dewey a:Louie)
```

```
Class(a:animal_lover complete intersectionOf(a:person
  restriction(a:has_pet minCardinality(3))))
```

```
ObjectProperty(a:has_pet domain(a:person) range(a:animal))
```

- Walt has pets Huey, Dewey and Louie.
- Huey, Dewey and Louie are all distinct individuals.
- Walt has at least three pets and is thus an animal lover.

Note that in this case, we don't actually need to include person in the definition of animal lover (as the domain restriction will allow us to draw this inference).

Wissensmodellierung: OWA vs. CWA

OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

	<i>Are all children of Bill male?</i>	<i>No idea, since we do not know all children of Bill.</i>	<i>If we assume that we know everything about Bill, then all of his children are male.</i>
child(Bill,Bob)			
Man(Bob)	$? \models \forall \text{child.Man(Bill)}$	DL answers don't know	Prolog yes
≤ 1 child.T(Bill)	$? \models \forall \text{child.Man(Bill)}$	yes	<i>Now we know everything about Bill's children.</i>

■ Wissensmodellierung: Domain und Range

- ObjectProperty(xyz:has_topping
 domain(xyz:Pizza)
 range(xyz:Pizza_topping))

$$\top \sqsubseteq \forall \text{has_topping}^-. \text{Pizza}$$

$$\top \sqsubseteq \forall \text{has_topping}. \text{Pizza_topping}$$
- Class(xyz:Ice_cream_cone partial
 restriction(xyz:has_topping someValuesFrom (xyz:Ice_cream)))

$$\text{Ice_cream_cone} \sqsubseteq \exists \text{has_topping}. \text{Ice_cream}$$
- Wenn Ice_cream_cone und Pizza *nicht* disjunkt sind:
 - Ice_cream_cone wird als Pizza klassifiziert
 - ...aber: Ice_cream wird *nicht* als Pizza_topping klassifiziert
 - Konsequenzen: *alle* Ice_cream_cones sind Pizzas,
und *manche* Ice_cream ist ein Pizza_topping

Wissensmodellierung: Some Research Challenges

- Schließen mit
 - uncertainty (fuzzy, probabilistisch)
 - Inkonsistenzen (parakonsistent)
 - Regeln/Rules (machen wir noch)
 - weitere KI-Paradigmen (nichtmonotones Schließen, Präferenzen, ...)
- Maintenance (updates, Infrastruktur, etc)
- Skalierbarkeit des Schließens
- ...

Laufende Forschung untersucht u.a. obige Punkte!

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. **Beweistheoretische Semantik**
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

Wichtige Inferenzprobleme

- Konsistenz der Wissensbasis KB \models **false**?
 - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz C \equiv \perp ?
 - Muss Klasse C leer sein?
- Klasseninklusion (Subsumption) C \sqsubseteq D?
 - Strukturierung der Wissensbasis
- Klassenäquivalenz C \equiv D?
 - Sind zwei Klassen eigentlich dieselbe?
- Klassendisjunktheit C \sqcap D = \perp ?
 - Sind zwei Klassen disjunkt?
- Klassenzugehörigkeit C(a)?
 - Ist Individuum a in der Klasse C?
- Instanzgenerierung (Retrieval) „alle X mit C(X) finden“
 - Finde alle (bekannt!) Individuen zur Klasse C.

Entscheidbarkeit von OWL DL

- **Entscheidbarkeit:** zu jedem Inferenzproblem gibt es einen immer terminierenden Algorithmus.
- OWL DL ist Fragment von FOL, also könnten (im Prinzip) FOL-Inferenzalgorithmen (Resolution, Tableaux) verwendet werden.
- Diese terminieren aber nicht immer!
- **Problem:** Finde immer terminierende Algorithmen!
Keine „naiven“ Lösungen in Sicht!

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. Beweistheoretische Semantik
 - a. **Rückführung von Reasoning auf Unerfüllbarkeit**
 - b. Klassische Beweiser: Tableaux
 - c. State-of-the-Art Beweiser via Resolution

Rückführung auf Unerfüllbarkeit

- Wir werden Tableau- und Resolutionsverfahren für OWL DL abwandeln.
 - Tableau- und Resolutionsverfahren zeigen Unerfüllbarkeit einer Theorie.
- Rückführung der Inferenzprobleme auf das Finden von Inkonsistenten in der Wissensbasis, d.h. zeigen der Unerfüllbarkeit der Wissensbasis!

Rückführung auf Unerfüllbarkeit/Konsistenz

- **Klassenkonsistenz** $C \equiv \perp$ gdw
 - $KB \cup \{C(a)\}$ unerfüllbar (a neu)
- **Klasseninklusion (Subsumption)** $C \sqsubseteq D$ gdw
 - $KB \cup \{C \sqcap \neg D\}$ unerfüllbar
- **Klassenäquivalenz** $C \equiv D$ gdw
 - $C \sqsubseteq D$ und $D \sqsubseteq C$
- **Klassendisjunktheit** $C \sqcap D = \perp$ gdw
 - $KB \cup \{(C \sqcap D)(a)\}$ unerfüllbar (a neu)
- **Klassenzugehörigkeit** $C(a)$ gdw
 - $KB \cup \{\neg C(a)\}$ unerfüllbar
- **Instanzgenerierung (Retrieval)** alle $C(X)$ finden
 - Schwerer, dies gut zu implementieren!

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. Beweistheoretische Semantik
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. **Klassische Beweiser: Tableaux**
 - c. State-of-the-Art Beweiser via Resolution

Naive Tableauregeln für SHIQ: Erzeugen einer ABox

Hitzler & Sure, 2005



33

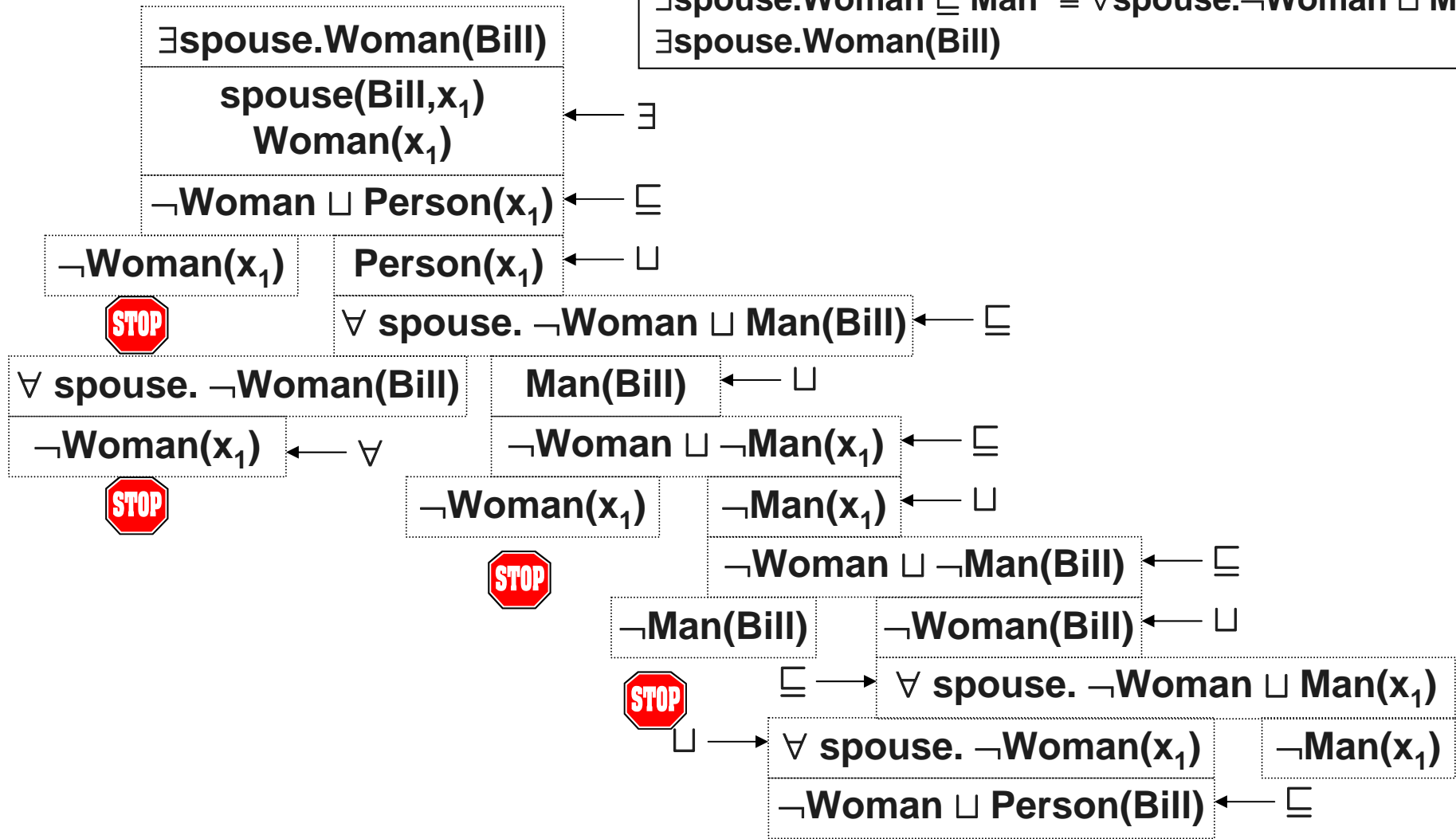
	Existing ABox statement	How ABox is changed
\sqcap	$C \sqcap D(s)$	Add $C(s)$ and $D(s)$
\sqcup	$C \sqcup D(s)$	Check either $C(s)$ or $D(s)$
\exists	$\exists R.C(s)$	Add $R(s, x)$ and $C(x)$ where x is new
\forall	$\forall R.C(s), R(s, t)$	Add $C(t)$
\geq	$\geq n R.C(s)$	Add $R(s, x_i), C(x_i)$ until $\#\{x \mid R(s, x)\} < n$
\leq	$\leq n R.C(s), \#\{x \mid R(s, x)\} > n$	Check all combinations $x_i = x_j$
	TBox Axiom	How ABox is changed
\sqsubseteq	$C \sqsubseteq D$	Add $\text{NNF}(\neg C) \sqcup \text{NNF}(D)(s)$ for any s
	When am I done?	
	$\neg A(s), A(s)$	Clash (don't expand any more)

Leichte Abwandlung des FOL Verfahrens.

NNF: Negationsnormalform

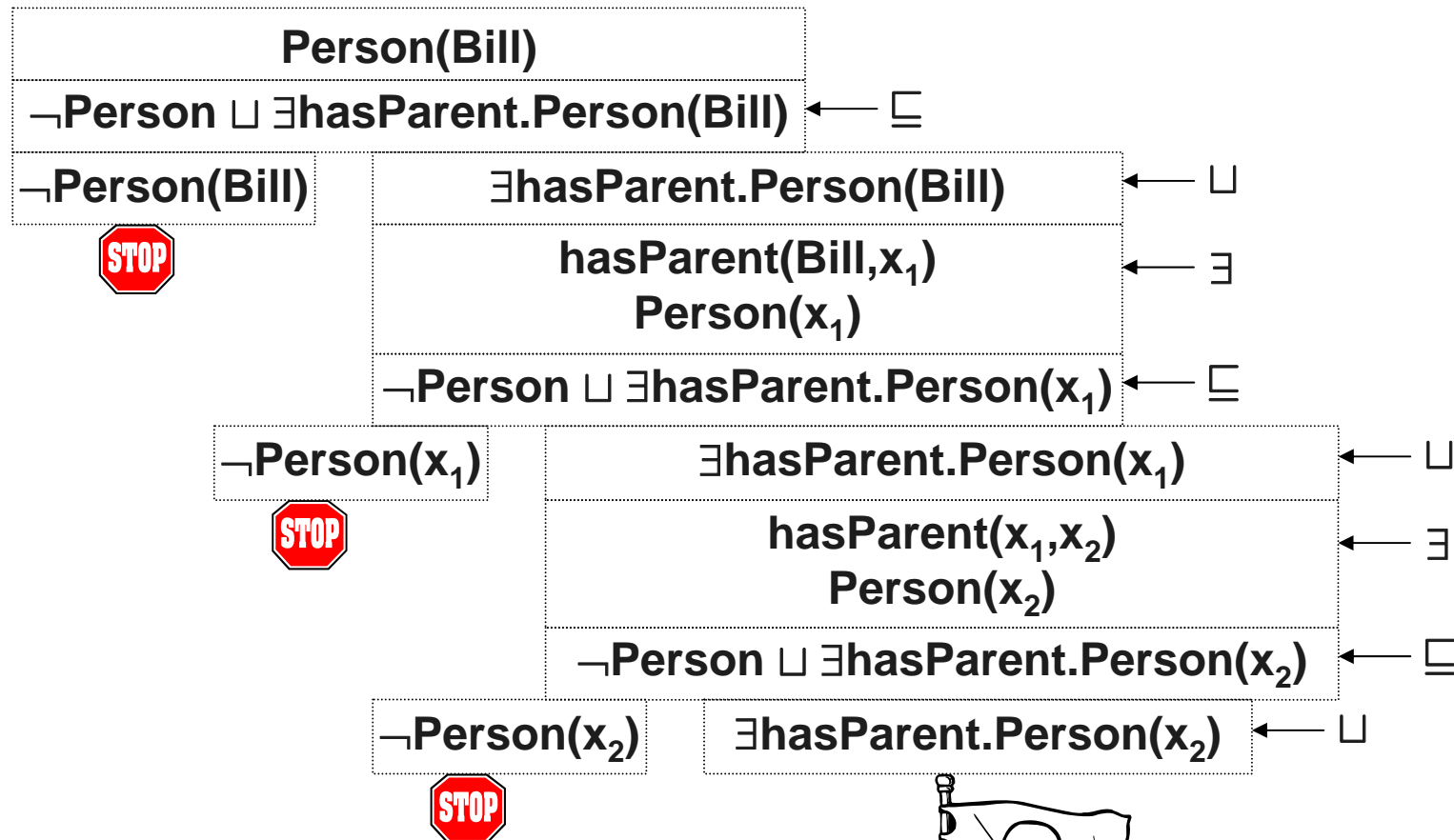
Beispiel

$Woman \sqcap Man \sqsubseteq \perp$	$\equiv \neg Woman \sqcup \neg Man$
$Woman \sqsubseteq Person$	$\equiv \neg Woman \sqcup Person$
$\exists spouse.Woman \sqsubseteq Man$	$\equiv \forall spouse.\neg Woman \sqcup Man$
$\exists spouse.Woman(Bill)$	



Das Terminierungsproblem

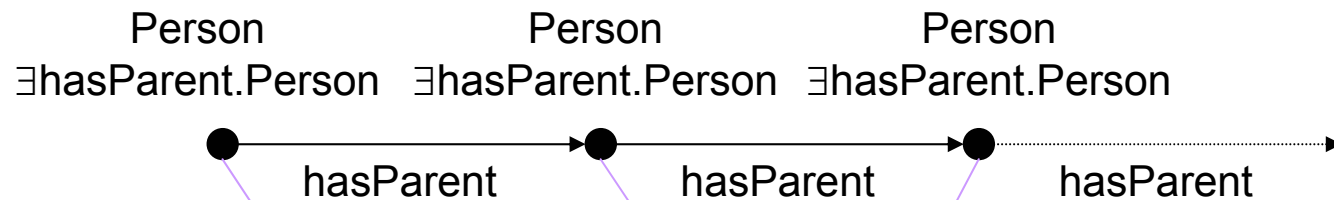
- Einziges Axiom: $\text{Person} \sqsubseteq \exists \text{hasParent. Person}$



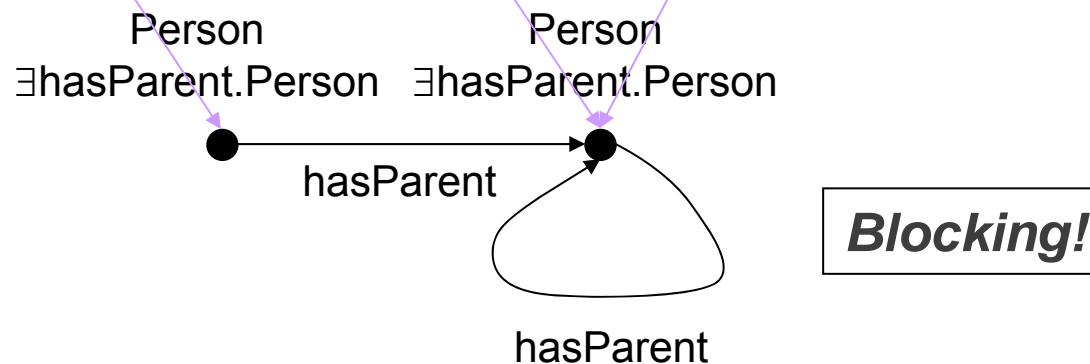
Problem tritt auf durch
Existenzquantoren und minCardinality

Lösung des Terminierungsproblems

- Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:

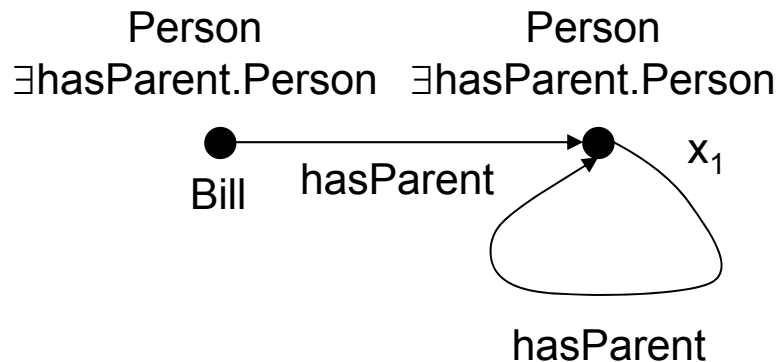
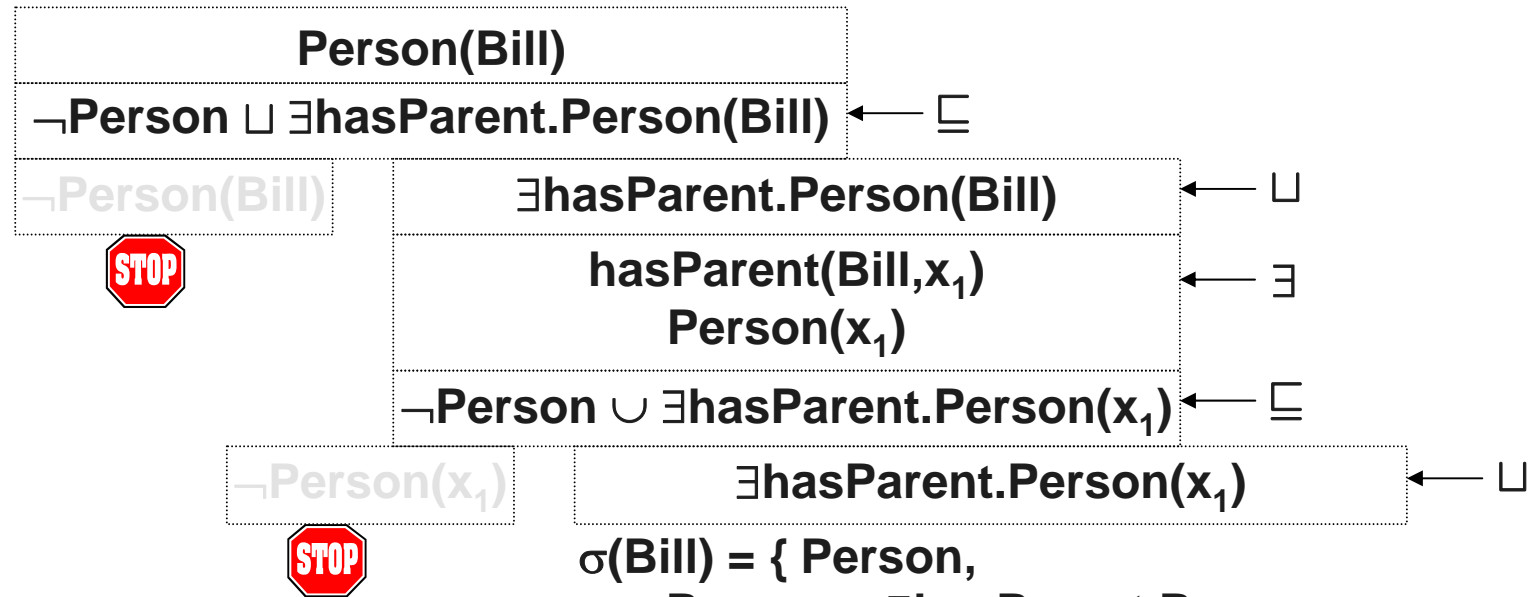


D.h. Wiederverwendung alter Knoten!

Es muss natürlich formal nachgewiesen werden, dass das ausreicht!

Tableau mit Blocking

- Einziges Axiom: $\text{Person} \sqsubseteq \exists \text{hasParent. Person}$



$\sigma(\text{Bill}) = \{ \text{Person}, \neg \text{Person} \sqcup \exists \text{hasParent. Person}, \exists \text{hasParent. Person} \}$

$\sigma(x_1) = \{ \text{Person}, \neg \text{Person} \sqcup \exists \text{hasParent. Person}, \exists \text{hasParent. Person} \}$

$\sigma(x_1) \subseteq \sigma(\text{Bill})$, so Bill blocks x_1 (marked STOP)

Tableauregeln mit Blocking (SHIQ)

	Existing ABox statement	How ABox is changed
\sqcap	$C \sqcap D(s)$	Add $C(s)$ and $D(s)$
\sqcup	$C \sqcup D(s)$	Check either $C(s)$ or $D(s)$
\exists	$\exists R.C(s)$, s is not blocked	Add $R(s, x)$ and $C(x)$ where x is new
\forall	$\forall R.C(s), R(s, t)$	Add $C(t)$
\geq	$\geq n R.C(s)$, s is not blocked	Add $R(s, x_i), C(x_i)$ until $\#\{x \mid R(s, x)\} < n$
\leq	$\leq n R.C(s), \#\{x \mid R(s, x)\} > n$	Check all combinations $x_i = x_j$
	TBox Axiom	How ABox is changed
\sqsubseteq	$C \sqsubseteq D$	Add $\text{NNF}(\neg C) \sqcup \text{NNF}(D)(s)$ for any s
	Remarks	
	$\neg A(s), A(s)$	Clash (don't expand any more)
	$\sigma(s) = \{ C \mid C(s) \}$	$\sigma(s)$ = set of concepts where s appears
	p blocks s	$\sigma(s) \subseteq \sigma(p)$

Tableauverfahren für OWL DL ohne Nominals

- D.h. für SHIN(D):
- Concrete Domains werden auf FOL zurückgeführt.
- Checks für Blocking werden komplizierter.
- Grundidee dieselbe!

- Schlechte Unterstützung von Instanzgenerierung.

- Tableau mit Blocking ist 2NExptime!
→ schlechter als nötig!

Tableaux-Beweiser

- Fact
 - <http://www.cs.man.ac.uk/~horrocks/FaCT/>
 - SHIQ
- Fact++
 - <http://owl.man.ac.uk/factplusplus/>
 - OWL Lite
- Pellet
 - <http://www.mindswap.org/2003/pellet/index.shtml>
 - SHIN(D)
- Racer
 - <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>
 - SHIQ(D)

Inhalt

- I. Was ist „Ontologie“?
- II. RDF(S) und FOL als Ontologiesprachen
- III. OWL
 1. Syntax und Modelltheoretische Semantik
 - a. Beschreibungslogiken: SHOIN(D)
 - b. OWL als SHOIN(D)
 - c. Serialisierungen
 - d. Wissensmodellierung in OWL
 2. Beweistheoretische Semantik
 - a. Rückführung von Reasoning auf Unerfüllbarkeit
 - b. Klassische Beweiser: Tableaux
 - c. **State-of-the-Art Beweiser via Resolution**

KAON2 OWL Reasoner

- Völlig neuer Ansatz
- Theorie entwickelt vor allem in Zusammenarbeit von
 - Ulrich Hustadt, Liverpool
 - Boris Motik, Karlsruhe
 - Ulrike Sattler, Manchester
- Die Entwicklung der Algorithmen war nur durch substantielle theoretische (logische) Grundlagenarbeit möglich.
- Implementierung des Grundsystems durch
 - Boris Motik (Dissertation)
- Wir besprechen die grundsätzliche Vorgehensweise.
- <http://kaon2.semanticweb.org>

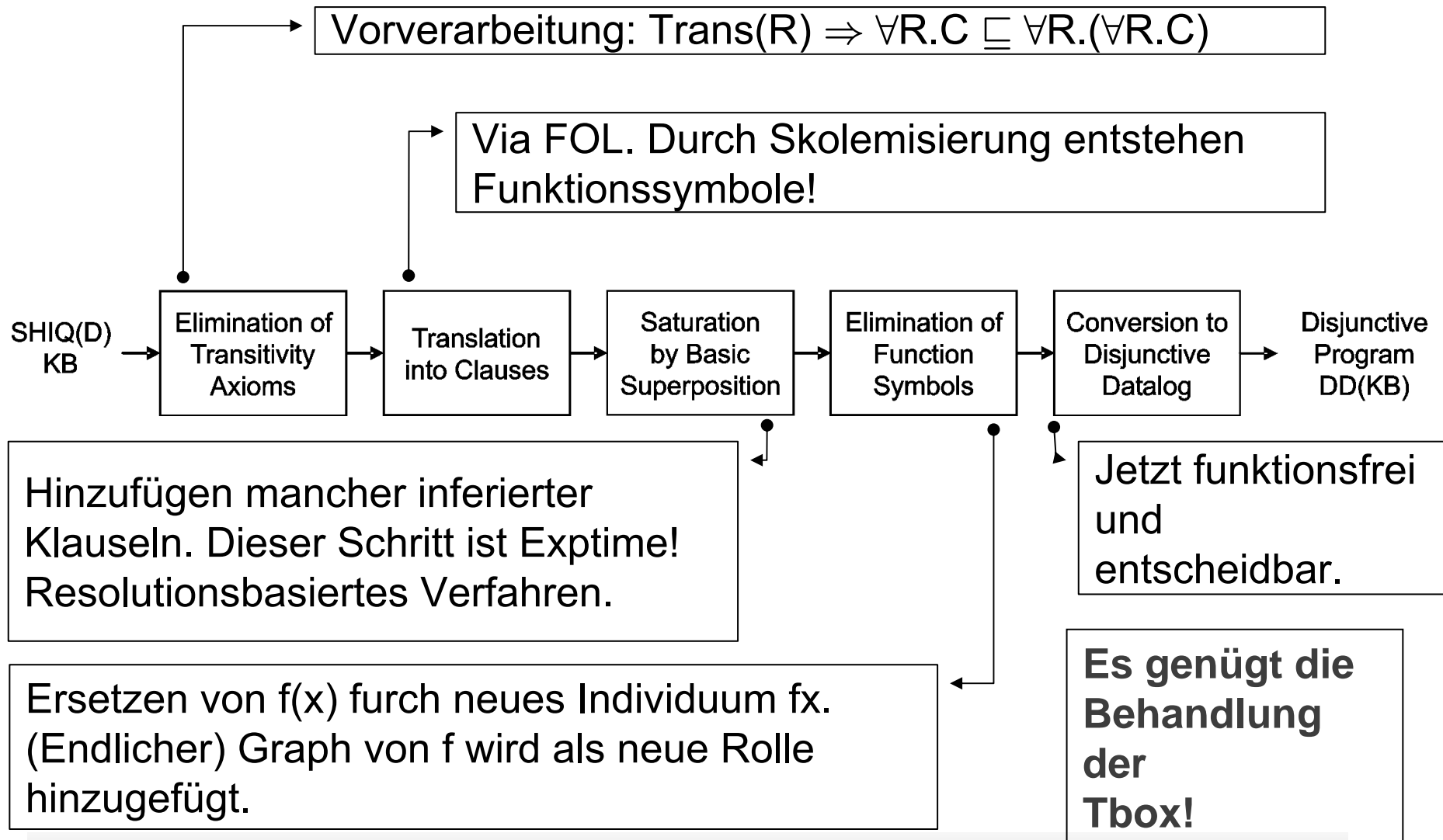
KAON2: Grundideen

- ABox-Reasoning (Instanzgenerierung) ist wichtiger für die Praxis als TBox-Reasoning.
- Resolutionsverfahren ist hervorragend für Instanzgenerierung geeignet (siehe Prolog).
- → Resolutionsbeweiser für OWL DL?
 - Naive Versuche via FOL schlagen fehl.
 - Grund: Transformation in FOL ergibt Existenzquantoren, die in Funktionssymbole skolemisiert werden müssen.
 - Terminierung mit Funktionssymbolen nicht erzwingbar!

KAON2: Vorgehensweise für Terminierung

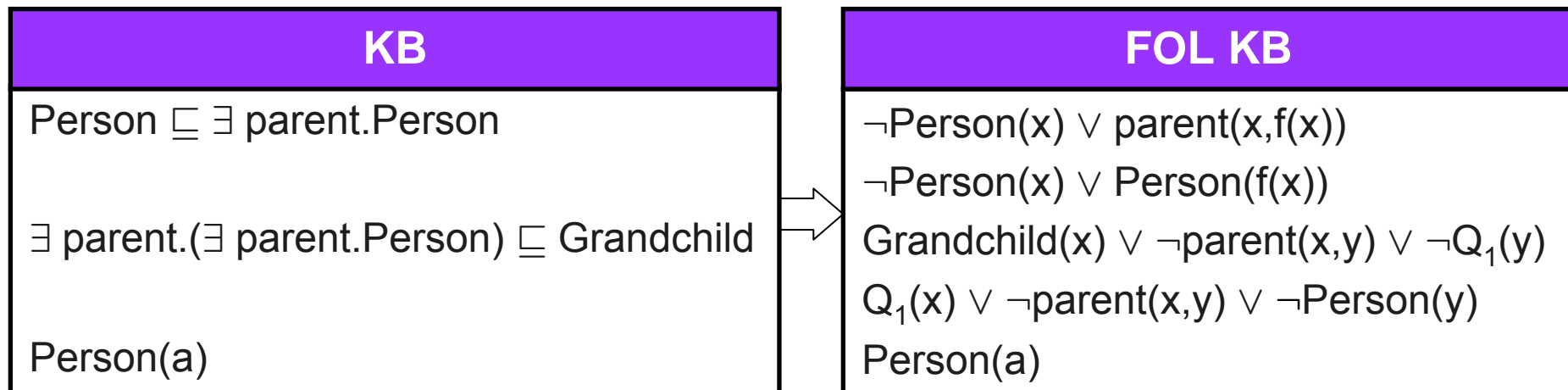
- Endlich viele neue durch Existenzquantor erzeugte Individuen reichen für alle Konsequenzen aus.
 - Wieviele und welche?
- Zunächst Behandlung der TBox: Ziehen aller (benötigten) logischen Konsequenzen.
 - Endliche Menge!
 - Neugenerierung von Individuen via Existenzquantoren dann nicht mehr nötig.
- Existenzquantoren (d.h. Skolemfunktionen) können dann entfernt werden!

Transformationsalgorithmus von KAON2

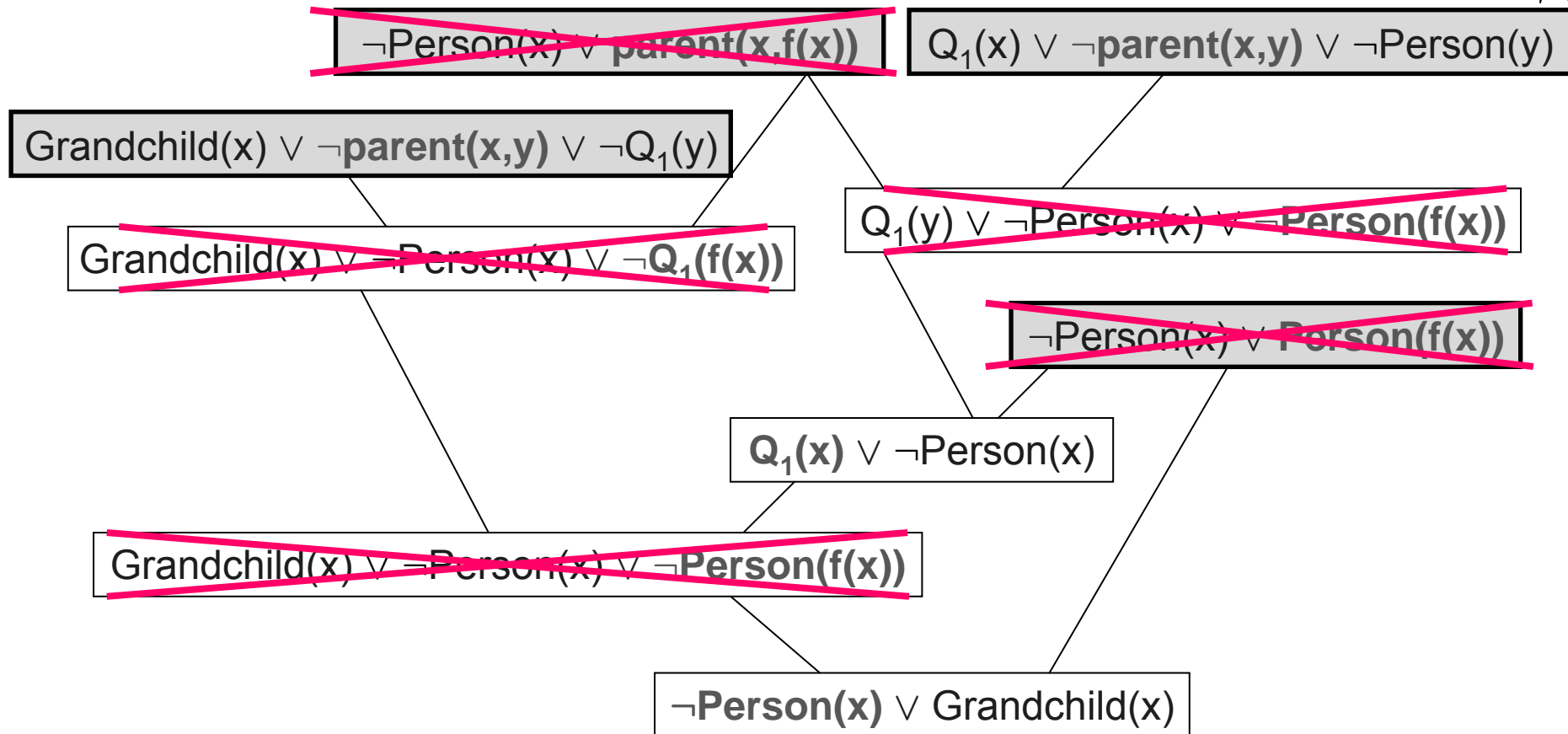


Einfaches Beispiel für Transformation

structural transformation & clausification



Saturierung



Translate to Datalog

Ergebnis: Datalog Programm

KB
$\text{Person} \sqsubseteq \exists \text{parent}.\text{Person}$ $\exists \text{parent}.\text{Grandchild} \sqsubseteq \text{Grandchild}$ $\text{Person}(a)$
DD(KB)
$Q_1(x) \leftarrow \text{parent}(x,y), \text{Person}(y)$ $\text{Grandchild}(x) \leftarrow \text{parent}(x,y), Q_1(y)$ $Q_1(x) \leftarrow \text{Person}(x)$ $\text{Grandchild}(x) \leftarrow \text{Person}(x)$ $\text{Person}(a)$

KAON2: Inferenzmechanismus

1. Übersetzung der TBox in funktionsfreie Klauseln.
(Exptime!)
 2. Hinzufügen der ABox.
 3. Inferenzprobleme in Konsistenzcheck umwandeln.
 4. Konsistenzcheck mit Standardmethoden für funktionsfreie Klauseln (z.B. magic sets).
(NP-vollständig!)
- TBox braucht nur einmal behandelt zu werden!
 - Algorithmus ist worst-case optimal!
 - Datenkomplexität ist NP!

Nominals

- Nominals heben die Kombinierte Komplexität von Exptime auf NExptime!
- Keines der aktuellen Systeme unterstützt Nominals, da keine sinnvollen Algorithmen vorlagen.
- Brandneues Ergebnis:
I. Horrocks and U. Sattler: A Tableaux Decision Procedure for SHOIQ. In *Proceedings of Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-05*, 2005. To appear.

Literatur

- F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2002. (Manche Kapitel)
- S. Staab, R. Studer (eds.): Handbook on Ontologies, Springer, 2004. (Manche Kapitel)
- W3C Dokumente (siehe Vorlesungswebseite)
- Literatur auf <http://kaon2.semanticweb.org>.

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 year's 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 ...