

Intelligente Systeme im World Wide Web

Web Ontology Language OWL: Semantik

Folien zur Vorlesung im Sommersemester 2007

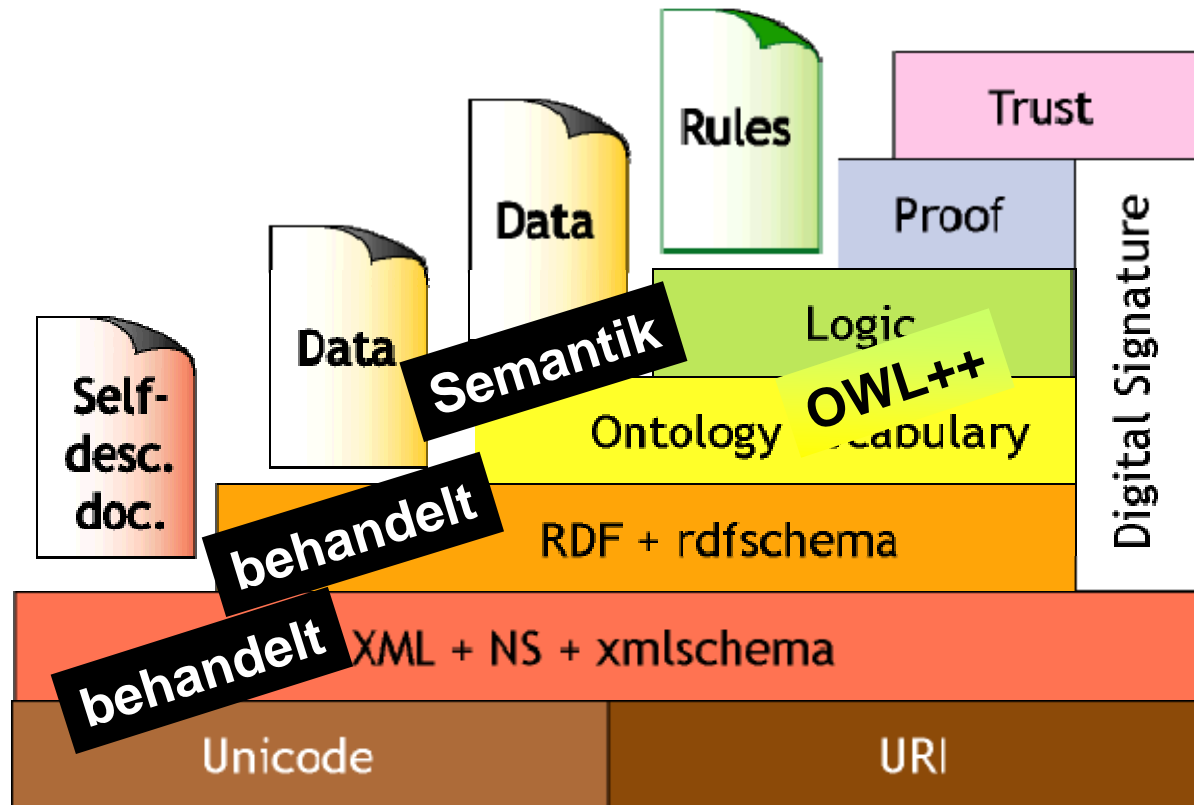
PD Dr. Pascal Hitzler

Dr. Sebastian Rudolph, Dr. Raphael Volz

Institut für Angewandte Informatik und Formale
Beschreibungsverfahren (AIFB)

Universität Karlsruhe (TH)

Die Semantic Web Schichttorte



Inhalt der nächsten 3 Sitzungen

- I. OWL – Syntax und allgemeines Verständnis
- II. Logik (Wiederholung)
- III. OWL – Semantische Grundlagen
 - a. Beschreibungslogiken
 - b. Beweistheorie
- IV. Ontologiesprache F-Logik

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

Inhalt

OWL – Semantische Grundlagen

1. Modelltheoretische Semantik

a. **Beschreibungslogiken: ALC**

b. OWL als SHOIN(D)

c. Serialisierungen

d. Wissensmodellierung in OWL

2. Beweistheoretische Semantik

a. Rückführung von Reasoning auf Unerfüllbarkeit

b. Klassische Beweiser: Tableaux

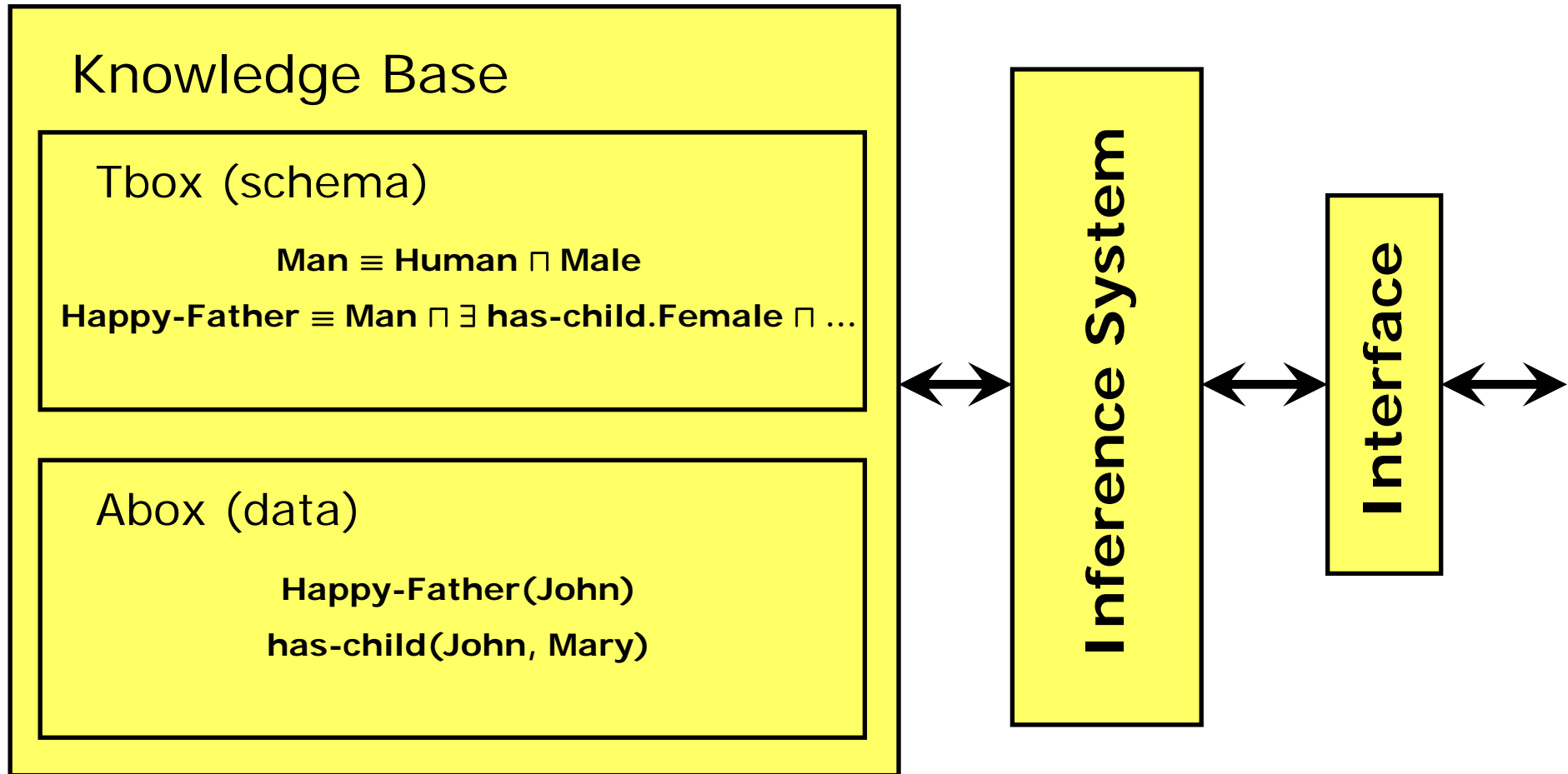
c. State-of-the-Art Beweiser via Resolution

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)
- Wir besprechen zunächst die einfachere ALC

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
 - Konjunktion, Disjunktion, Negation sind Konstruktoren, geschrieben \sqcap , \sqcup , \neg .
 - Quantoren schränken Rollenbereiche ein:

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

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 \sqsubseteq^+ hasAncestor
 - Role composition: hasParent.hasBrother(uncle)

ALC: Grundbausteine

- Grundbausteine:
 - Klassen
 - Rollen
 - Individuen
- Professor(RudiStuder)
 - Individuum RudiStuder ist in Klasse Professor
- Zugehoerigkeit(RudiStuder,AIFB)
 - RudiStuder ist dem AIFB zugehörig

ALC: Subklassenbeziehungen

- Professor \sqsubseteq Fakultaetsmitglied
 - entspricht $(\forall x)(\text{Professor}(x) \rightarrow \text{Fakultaetsmitglied}(x))$
 - entspricht owl:subClassOf
- Professor \equiv Fakultaetsmitglied
 - entspricht $(\forall x)(\text{Professor}(x) \leftrightarrow \text{Fakultaetsmitglied}(x))$
 - entspricht owl:equivalentClass

ALC: komplexe Klassenbeziehungen

- Konjunktion \sqcap entspricht owl:intersectionOf
- Disjunktion \sqcup entspricht owl:unionOf
- Negation \neg entspricht owl:complementOf
- Professor \sqsubseteq (Person \sqcap Unversitaetsangehoeriger)
 \sqcup (Person \sqcap \neg Doktorand)

$$(\forall x)(\text{Professor}(x) \rightarrow \\ ((\text{Person}(x) \wedge \text{Unversitaetsangehoeriger}(x)) \\ \vee (\text{Person}(x) \wedge \neg \text{Doktorand}(x))))$$

ALC: Quantoren

- $\text{Pruefung} \sqsubseteq \forall \text{hatPruefer. Professor}$
 $(\forall x)(\text{Pruefung}(x) \rightarrow (\forall y)(\text{hatPruefer}(x,y) \rightarrow \text{Professor}(y)))$
 - entspricht owl:allValuesFrom
- $\text{Pruefung} \sqsubseteq \exists \text{hatPruefer. Person}$
 $(\forall x)(\text{Pruefung}(x) \rightarrow (\exists y)(\text{hatPruefer}(x,y) \wedge \text{Person}(y)))$
 - entspricht owl:someValuesFrom

Modellierung in ALC

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

ALC: Syntax

- Folgende Syntaxregeln erzeugen Klassen in ALC. Dabei ist A eine atomare Klasse und R eine Rolle.
$$C, D \rightarrow A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$
- Eine *ALC TBox* besteht aus Aussagen der Form $C \sqsubseteq D$ und $C \equiv D$, wobei C, D Klassen sind.
- Eine *ALC ABox* besteht aus Aussagen der Form $C(a)$ und $R(a,b)$, wobei C eine komplexe Klasse, R eine Rolle und a, b Individuen sind.
- Eine *ALC-Wissensbasis* besteht aus einer ABox und einer TBox.

ALC: Semantik

Übersetzung von TBox-Aussagen in die Prädikatenlogik mittels der Abbildung π (rechts).

Dabei sind C,D komplexe Klassen, R eine Rolle und A eine atomare Klasse.

$$\pi(C \sqsubseteq D) = (\forall x)(\pi_x(C) \rightarrow \pi_x(D))$$

$$\pi(C \equiv D) = (\forall x)(\pi_x(C) \leftrightarrow \pi_x(D))$$

$$\pi_x(A) = A(x)$$

$$\pi_x(\neg C) = \neg \pi_x(C)$$

$$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$$

$$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$$

$$\pi_x(\forall R.C) = (\forall y)(R(x, y) \rightarrow \pi_y(C))$$

$$\pi_x(\exists R.C) = (\exists y)(R(x, y) \wedge \pi_y(C))$$

$$\pi_y(A) = \mathbf{A(y)}$$

$$\pi_y(\neg C) = \neg \pi_y(C)$$

$$\pi_y(C \sqcap D) = \pi_y(C) \wedge \pi_y(D)$$

$$\pi_y(C \sqcup D) = \pi_y(C) \vee \pi_y(D)$$

$$\pi_y(\forall R.C) = (\forall x)(R(y, x) \rightarrow \pi_x(C))$$

$$\pi_y(\exists R.C) = (\exists x)(R(y, x) \wedge \pi_x(C))$$

DL Wissensbasen

- DL Wissensbasen bestehen 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.

Einfaches Beispiel

Terminologisches Wissen (*TBox*):

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

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

Wissen um Individuen (*ABox*):

Orphan(harrypotter)

hasParent(harrypotter, jamespotter)

Semantik und logische Konsequenzen klar, da
übersetzbar nach FOL.

Inhalt

OWL – Semantische Grundlagen

1. Modelltheoretische Semantik
 - a. Beschreibungslogiken: ALC
 - 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 und ALC

Folgende OWL DL Sprachelemente sind in ALC repräsentierbar:

- Klassen, Rollen, Individuen
- Klassenzugehörigkeit, Rolleninstanzen
- owl:Thing und owl:Nothing
- Klasseninklusion, -äquivalenz, -disjunktheit
- owl:intersectionOf, owl:unionOf
- owl:complementOf
- Rollenrestriktionen
- rdfs:range und rdfs:domain

OWL als SHOIN(D): Individuen

- owl:sameAs
 - DL: $a=b$
 - FOL: Erweiterung durch Gleichheitsprädikat
- owl:differentFrom
 - DL: $a \neq b$
 - FOL: $\neg(a=b)$

OWL als SHOIN(D): abgeschlossene Klassen

Abgeschlossene Klassen

- owl:oneOf
 - DL: $C \equiv \{a,b,c\}$
 - FOL: $(\forall x) (C(x) \leftrightarrow (x=a \vee x=b \vee x=c))$
- owl:hasValue
 - Darstellbar mittels owl:someValuesFrom und owl:oneOf

OWL als SHOIN(D): Zahlenrestriktionen

Zahlenrestriktionen mittels Gleichheitsprädikat

```

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

```

Eine Prüfung kann *höchstens zwei* Prüfer haben.

- DL: $\text{Pruefung} \sqsubseteq \leq 2 \text{ hatPruefer}$
- In FOL: $(P \dots \text{Prüfung}, h \dots \text{hatPruefer})$
 $(\forall x)(P(x) \rightarrow \neg(\exists x_1)(\exists x_2)(\exists x_3)(x_1 \neq x_2 \wedge x_2 \neq x_3 \wedge x_1 \neq x_3 \wedge h(x,x_1) \wedge h(x,x_2) \wedge h(x,x_3)))$

Entsprechend für die anderen Zahlenrestriktionen

OWL als SHOIN(D): Rollenkonstruktoren

- Rdfs:subPropertyOf
 - DL: $R \sqsubseteq S$
 - FOL: $(\forall x)(\forall y)(R(x,y) \rightarrow S(x,y))$
- Entsprechend Rollenäquivalenz
- Inverse Rollen: $R \equiv S^{-}$
 - FOL: $(\forall x)(\forall y)(R(x,y) \leftrightarrow S(y,x))$
- Transitive Rollen: $R^+ \sqsubseteq R$
 - FOL: $(\forall x)(\forall y)(\forall z)(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$
- Symmetrie: $R \equiv R^{-}$
- Funktionalität: $\top \sqsubseteq \leq 1 R$
- Inverse Funktionalität: $\top \sqsubseteq \leq 1 R^{-}$

Datentypen

- Erlaubt ist die Verwendung von Datentypen im zweiten Argument konkreter Rollen in der ABox.
- Ausserdem kann eine Menge konkreter Daten eine abgeschlossene Klasse bilden
- Datentypen lassen sich nicht ohne Weiteres in FOL ausdrücken. Man kann die FOL Semantik aber entsprechend erweitern.

OWL DL als SHOIN(D): Überblick

Erlaubt sind:

- ALC
- Gleichheit und Ungleichheit zwischen Individuen
- Abgeschlossene Klassen
- Zahlenrestriktionen
- Subrollen und Rollenäquivalenz
- Inverse und transitive Rollen
- Datentypen

Bezeichner für Beschreibungslogiken

- ALC: Attribute Language with Complement
- S: ALC + Rollentransitivität
- H: Subrollenbeziehung
- O: abgeschlossene Klassen
- I: inverse Rollen
- N: Zahlenrestriktionen $\leq n$ R etc.
 - Q: Qualifizierende Zahlenrestriktionen $\leq n$ R.C etc.
- (D): Datentypen
- F: Funktionale Rollen

- OWL DL ist SHOIN(D)
- OWL Lite ist SHIF(D)

Übersicht Syntax für DLs (ohne Datentypen)

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)

OWL DL als DL: Übersicht Klassenconstructoren

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 Constructoren erlaubt:

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

OWL DL als DL: Übersicht 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:
 $C \equiv D$ gdw ($C \sqsubseteq D$ und $D \sqsubseteq C$)
- Offensichtliche FOL-Äquivalenzen
 $C \equiv D \Leftrightarrow (\forall x) (C(x) \leftrightarrow D(x))$
 $C \sqsubseteq D \Leftrightarrow (\forall x) (C(x) \rightarrow D(x))$

Komplexitäten (worst-case)

OWL Variante	Datenkomplexität	Kombinierte Komplexität
OWL Full	unentscheidbar	unentscheidbar
OWL DL	unbekannt	NExptime
OWL DL ohne Nominals	NP	Exptime
OWL Lite	NP	Exptime

Datenkomplexität: nur bezüglich ABox

Kombinierte Komplexität: bezüglich ABox und TBox

Inhalt

OWL – Semantische Grundlagen

1. Modelltheoretische Semantik
 - a. Beschreibungslogiken: ALC
 - 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: DL und 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

OWL – Semantische Grundlagen

1. Modelltheoretische Semantik
 - a. Beschreibungslogiken: ALC
 - 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.

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

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.

	Are all children of Bill male?	No idea, since we do not know all children of Bill.	If we assume that we know everything about Bill, then all of his children are male.
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	Now we know everything about Bill's children.

■ 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

OWL – Semantische Grundlagen

1. Modelltheoretische Semantik

- a. Beschreibungslogiken: ALC
- 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

- Globale 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

OWL – Semantische Grundlagen

1. Modelltheoretische Semantik
 - a. Beschreibungslogiken: ALC
 - 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.
 - Genauer: Wir werden nur ALC behandeln.
 - 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
 - $KB \cup \{C \sqcap \neg D(a)\}$ unerfüllbar (a neu)
- **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 (a neu)
- **Instanzgenerierung (Retrieval)** alle $C(X)$ finden
 - Prüfe Klassenzugehörigkeit für alle Individuen.
 - Schwerer, dies gut zu implementieren!

Inhalt

OWL – Semantische Grundlagen

1. Modelltheoretische Semantik
 - a. Beschreibungslogiken: ALC
 - 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

ALC Tableauverfahren: Inhalt

- **Transformation in Negationsnormalform**
- Naives Tableauverfahren
- Tableauverfahren mit Blocking

Transformation in Negationsnormalform

Gegeben eine Wissensbasis W .

- Ersetze $C \equiv D$ durch $C \sqsubseteq D$ und $D \sqsubseteq C$.
- Ersetze $C \sqsubseteq D$ durch $\neg C \sqcup D$.
- Wende die Regeln auf der folgenden Folie an, bis es nicht mehr geht.

Resultierende Wissensbasis: $NNF(W)$

Negationsnormalform von W .

Negation steht nur noch direkt vor atomaren Klassen.

$\text{NNF}(C) = C$, falls C atomar ist

$\text{NNF}(\neg C) = \neg C$, falls C atomar ist

$\text{NNF}(\neg\neg C) = \text{NNF}(C)$

$\text{NNF}(C \sqcup D) = \text{NNF}(C) \sqcup \text{NNF}(D)$

$\text{NNF}(C \sqcap D) = \text{NNF}(C) \sqcap \text{NNF}(D)$

$\text{NNF}(\neg(C \sqcup D)) = \text{NNF}(\neg C) \sqcap \text{NNF}(\neg D)$

$\text{NNF}(\neg(C \sqcap D)) = \text{NNF}(\neg C) \sqcup \text{NNF}(\neg D)$

$\text{NNF}(\forall R.C) = \forall R.\text{NNF}(C)$

$\text{NNF}(\exists R.C) = \exists R.\text{NNF}(C)$

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

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

W und $\text{NNF}(W)$ sind logisch äquivalent.

Negationsnormalform: Beispiel

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

In Negationsnormalform:

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

ALC Tableauverfahren: Inhalt

- Transformation in Negationsnormalform
- **Naives Tableauverfahren**
- Tableauverfahren mit Blocking

Naives Tableauverfahren

Rückführung auf Unerfüllbarkeit/Widerspruch

Idee:

- Gegeben Wissensbasis W .
- Erzeugen von Konsequenzen der Form $C(a)$ und $\neg C(a)$, bis Widerspruch gefunden.

Einfaches Beispiel

$C(a)$

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

$\neg C(a)$ ist logische Konsequenz:

2. Formel in FOL: $\neg C(a) \wedge D(a)$

daraus folgt u.a. $\neg C(a)$

Widerspruch ist gefunden.

Weiteres Beispiel

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

Ableitung von Konsequenzen:

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

Nun Fallunterscheidung

1. $\neg C(a)$

Widerspruch

2. $D(a)$

Widerspruch

Teilen des Tableaus in zwei *Zweige*.

Tableau: Definitionen

- *Tableauzweig*:
Endliche Menge von Aussagen der Form
 $C(a)$, $\neg C(a)$, $R(a,b)$.
- *Tableau*: Endliche Menge von Tableauzweigen.
- Tableauzweig ist *abgeschlossen* wenn er ein Paar widersprüchlicher Aussagen $C(a)$ und $\neg C(a)$ enthält.
- Tableau ist *abgeschlossen*, wenn jeder Zweig von ihm abgeschlossen ist.

Aufbau eines Tableaus

Auswahl	Aktion
$C(a) \in W$ (ABox)	Füge $C(a)$ hinzu.
$R(a, b) \in W$ (ABox)	Füge $R(a, b)$ hinzu.
$C \in W$ (TBox)	Füge $C(a)$ für ein bekanntes Individuum a hinzu.
$(C \sqcap D)(a) \in A$	Füge $C(a)$ und $D(a)$ hinzu.
$(C \sqcup D)(a) \in A$	Dupliziere den Zweig. Füge zum einen Zweig $C(a)$ und zum anderen Zweig $D(a)$ hinzu.
$(\exists R.C)(a) \in A$	Füge $R(a, b)$ und $C(b)$ für neues Individuum b hinzu.
$(\forall R.C)(a) \in A$	Falls $R(a, b) \in A$, so füge $C(b)$ hinzu.

Ist das resultierende Tableau abgeschlossen, so ist die ursprüngliche Wissensbasis unerfüllbar.

Man wählt dabei immer nur solche Elemente aus, die auch wirklich zu neuen Elementen im Tableau führen. Ist dies nicht möglich, so terminiert der Algorithmus und die Wissensbasis ist erfüllbar.

Beispiel

- P ... Professor
E ... Person
U ... Universitätsangehöriger
D ... Doktorand
- Wissensbasis: $P \sqsubseteq (E \sqcap U) \sqcup (E \sqcap \neg D)$
Ist $P \sqsubseteq E$ logische Konsequenz?
- Wissensbasis (mit Anfrage) in NNF:
 $\{\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D), (P \sqcap \neg E)(a)\}$

Beispiel (Fortsetzung)

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

Tableau:

$(P \sqcap \neg E)(a)$ (aus Wissensbasis)

$P(a)$

$\neg E(a)$

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

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

$(E \sqcap U)(a)$

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

$E(a)$

$E(a)$

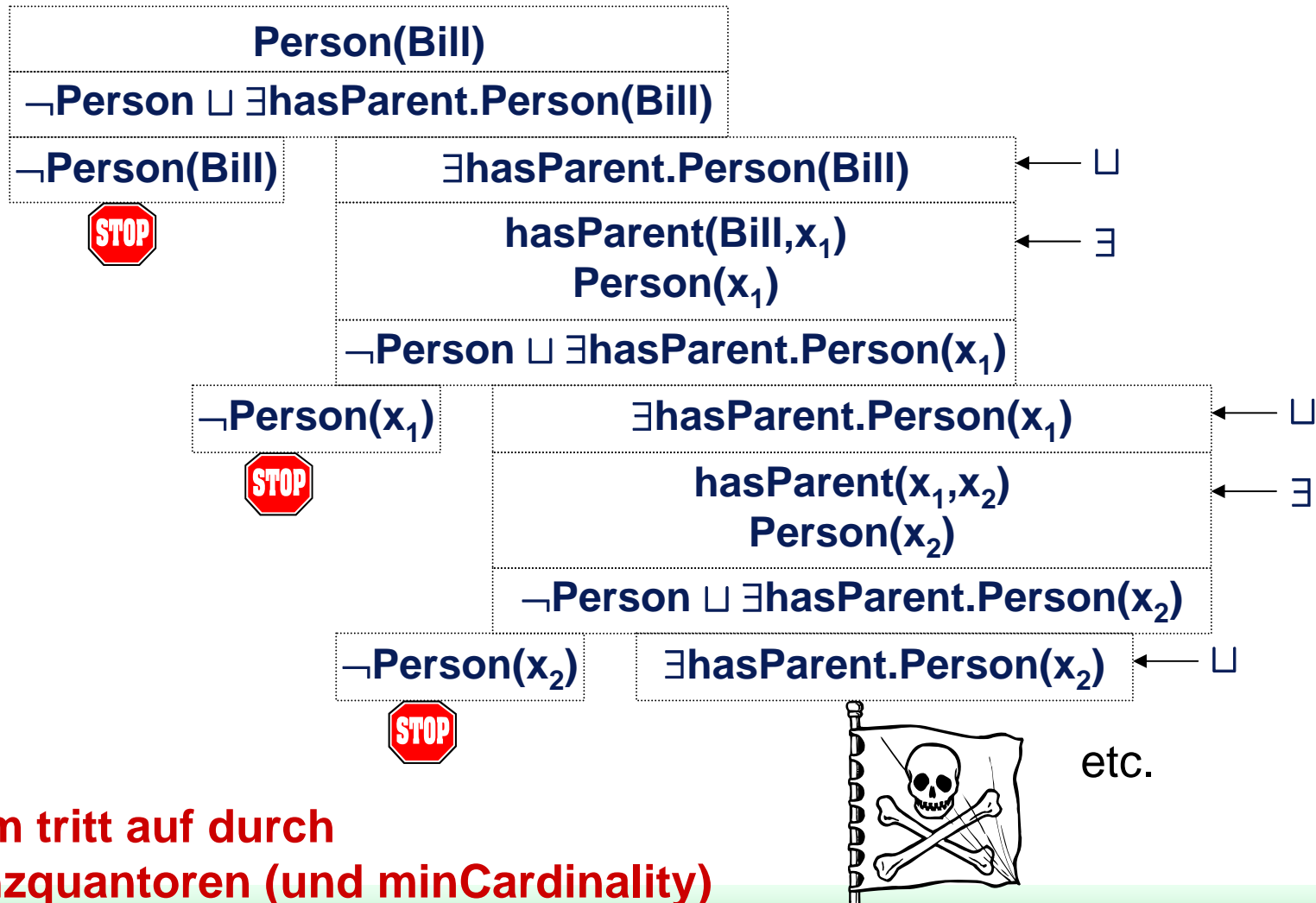
$U(a)$

$\neg D(a)$

D.h. Wissensbasis ist unerfüllbar, d.h. $P \sqsubseteq E$.

Das Terminierungsproblem

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$
Abzuleiten: $\neg \text{Person}(\text{Bill})$



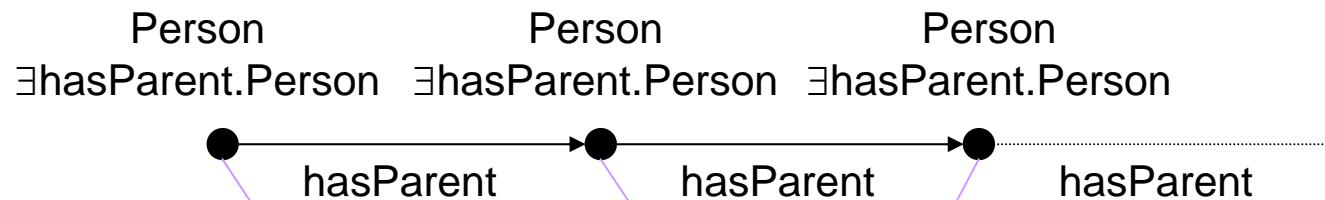
Problem tritt auf durch
Existenzquantoren (und minCardinality)

ALC Tableauverfahren: Inhalt

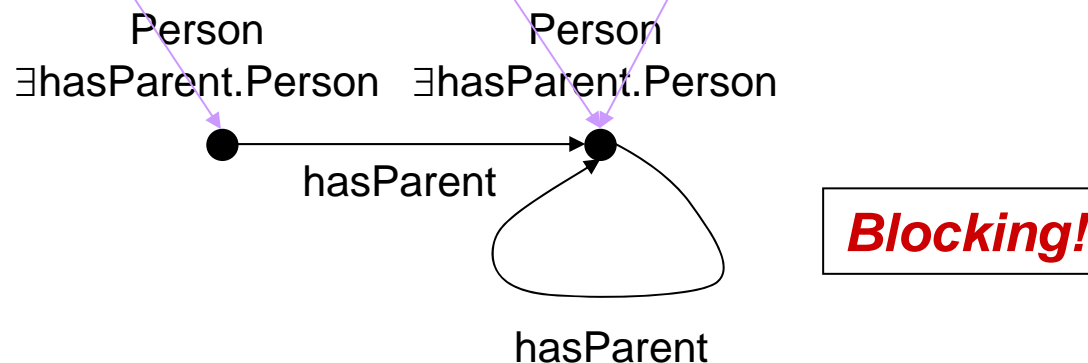
- Transformation in Negationsnormalform
- Naives Tableauverfahren
- **Tableauverfahren mit Blocking**

Lösung des Terminierungsproblems

- Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:

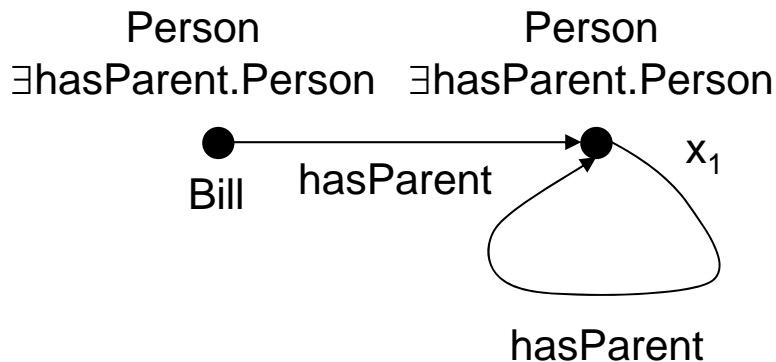
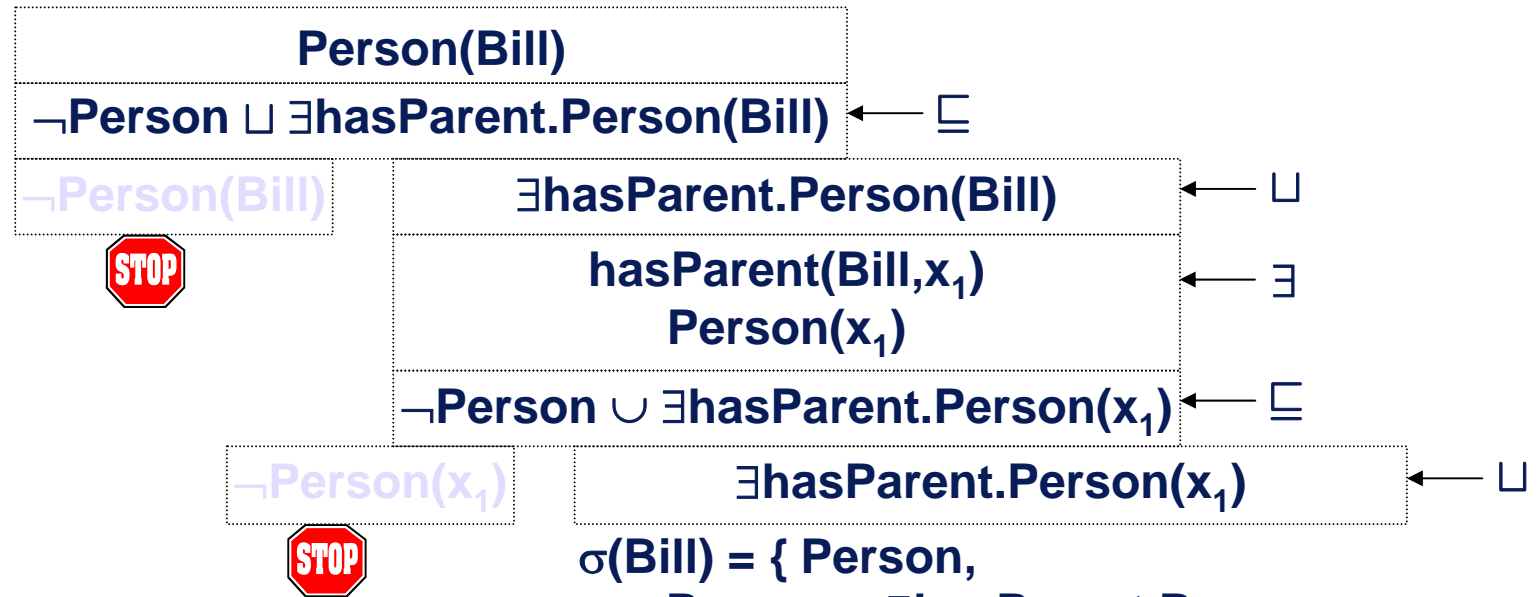


D.h. Wiederverwendung alter Knoten!

Es muss natürlich formal nachgewiesen werden, dass das ausreicht!

Tableau mit Blocking

- Einziges Axiom: $\neg\text{Person} \sqcup \exists\text{hasParent}.\text{Person}$
 Abzuleiten: $\neg\text{Person}(\text{Bill})$



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



Blocking

Die Auswahl von $(\exists R.C)(a)$ im Tableauzweig A ist *blockiert*, falls es ein Individuum b gibt, so dass $\{C \mid C(a) \in A\} \subseteq \{C \mid C(b) \in A\}$ ist.

Zwei Möglichkeiten der Terminierung:

1. Abschluss des Tableaus.
Dann Wissensbasis unerfüllbar.
2. Keine ungeblockte Auswahl führt zu Erweiterung.
Dann Wissensbasis erfüllbar.

Beispiel

- F ... Frau
h ... hatMutter
V ... Vogel
t ... Tweety
- Wissensbasis $\{F \sqsubseteq \exists h.F, V(t)\}$
- Wir wollen zeigen, dass Tweety *keine* Frau ist, d.h. dass $\neg F(t)$ logische Konsequenz ist.
- Dies wird uns nicht gelingen.
D.h. Tweety *kann* eine Frau sein.

Beispiel (Fortsetzung)

TBox: $\neg F \sqcup \exists h.F$

Tableau:

$V(t)$ (aus Wissensbasis)

$F(t)$ (negierte Anfrage in NNF)

$(\neg F \sqcup \exists h.F)(t)$

$\neg F(t)$ $(\exists h.F)(t)$

$h(t,s)$

$F(s)$

$(\neg F \sqcup \exists h.F)(s)$

$\neg F(s)$ $(\exists h.F)(s)$

geblockt durch t

Sowohl s als auch t fallen unter

$F, \neg F \sqcup \exists h.F, \exists h.F$

Keine andere Auswahl möglich.

Tableauverfahren für OWL DL

- Die Grundidee ist dieselbe!
- Kompliziertere Blockingregeln müssen verwendet werden.
- 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/>
 - 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)

Inhalt

OWL – Semantische Grundlagen

1. 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>

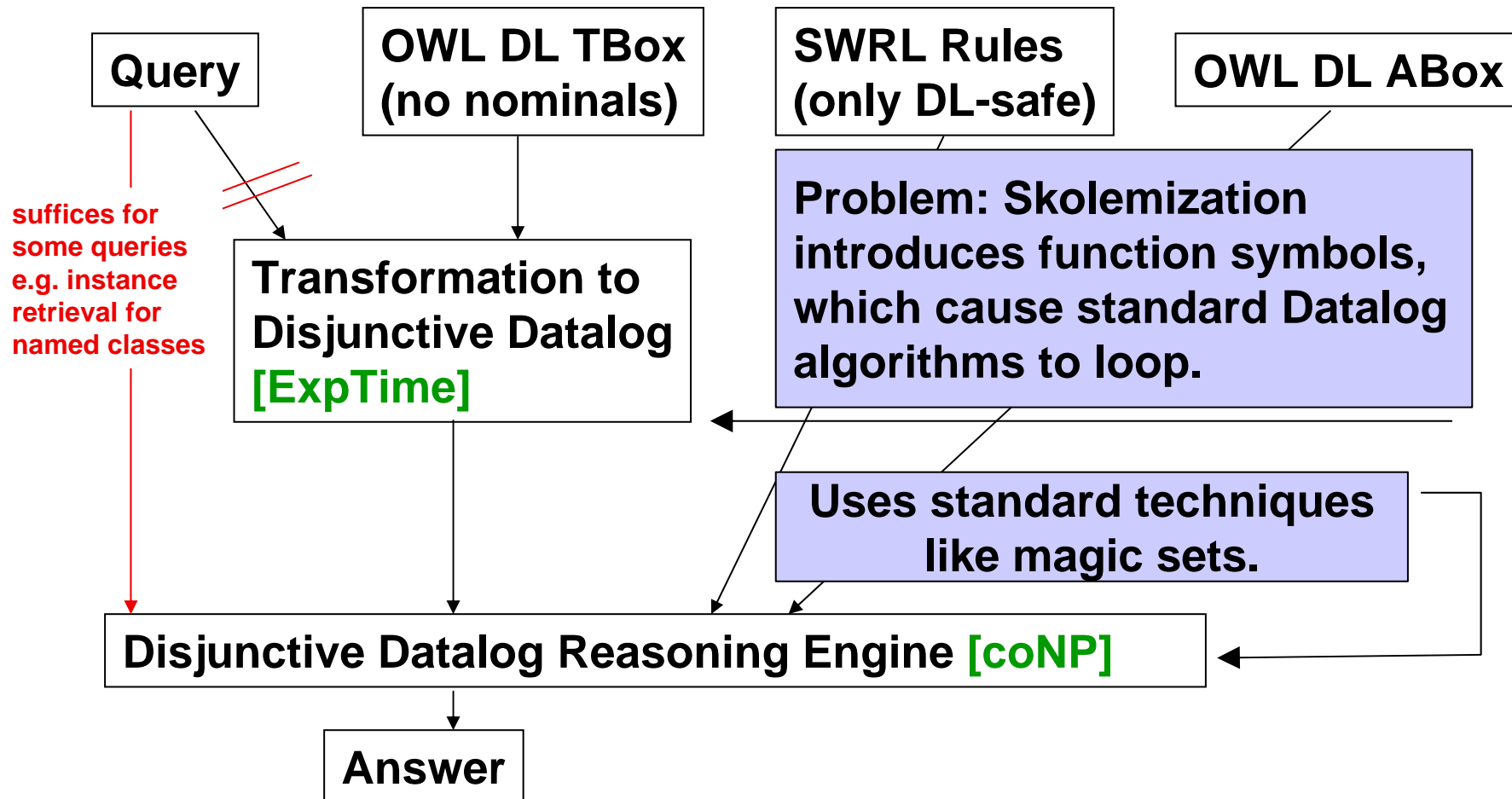
KAON2: Grundideen

- 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!

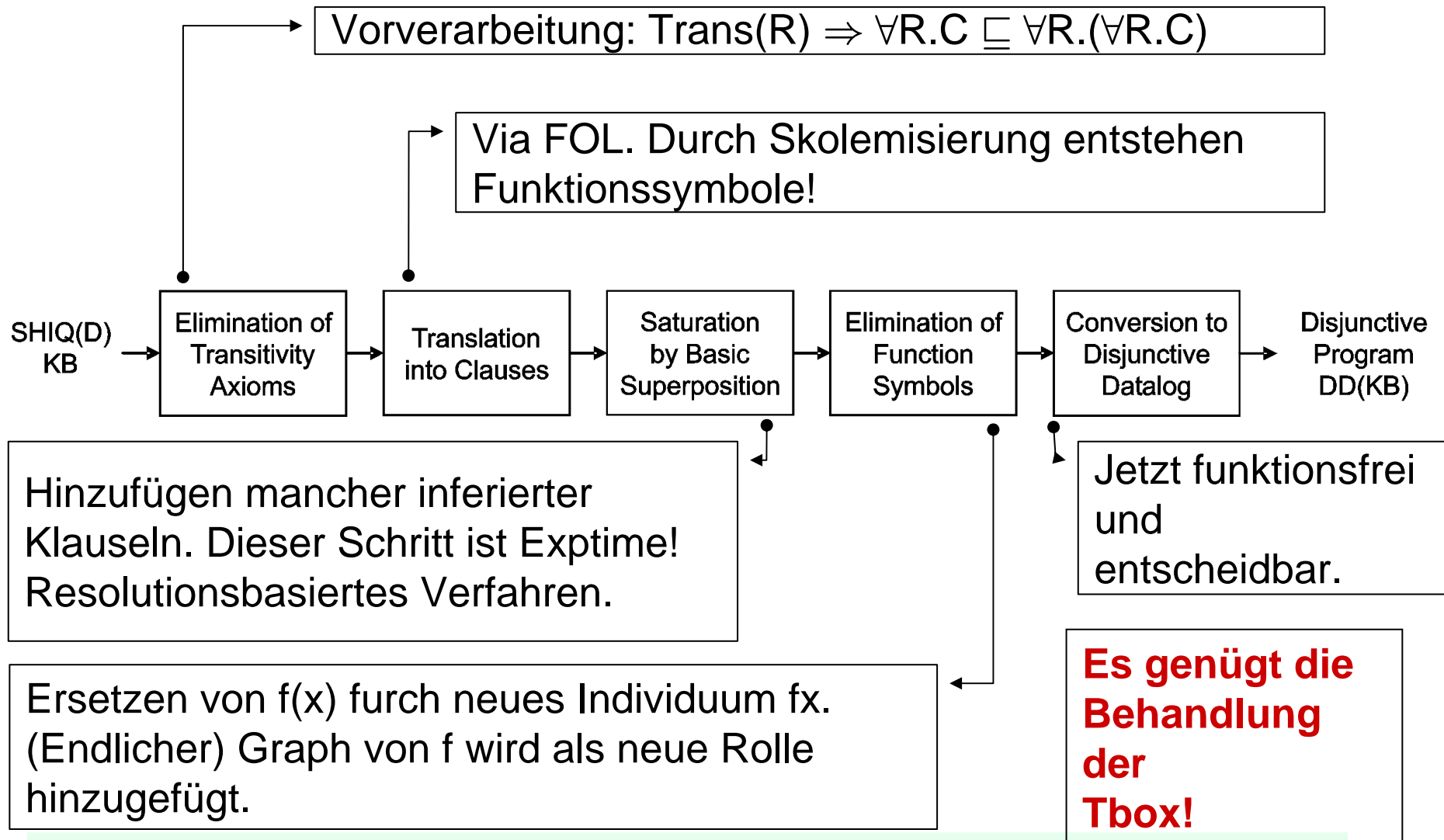
KAON2: Vorgehensweise für Terminierung

- 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!

KAON2 Reasoner Kernarchitektur

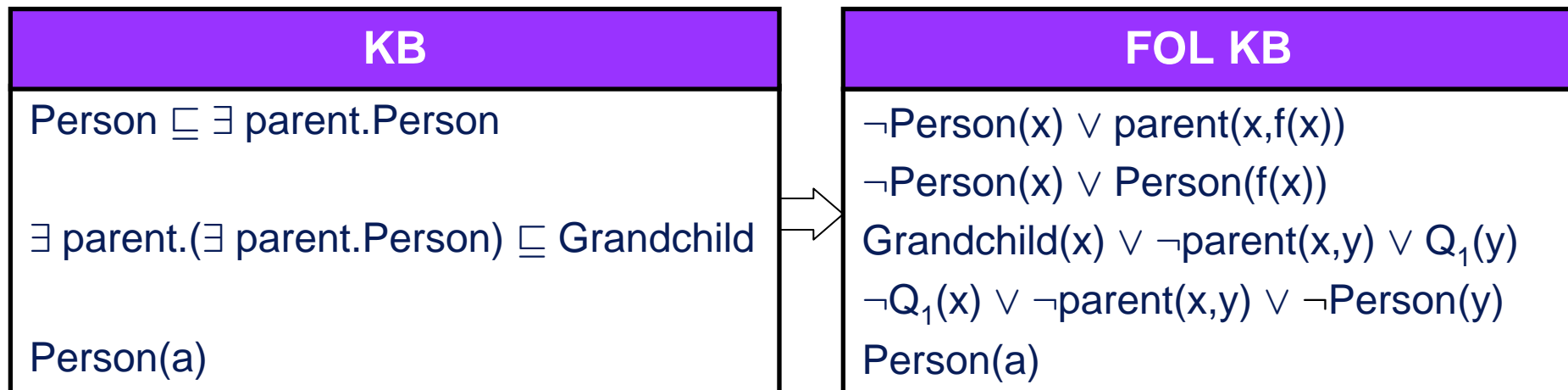


Transformationsalgorithmus von KAON2

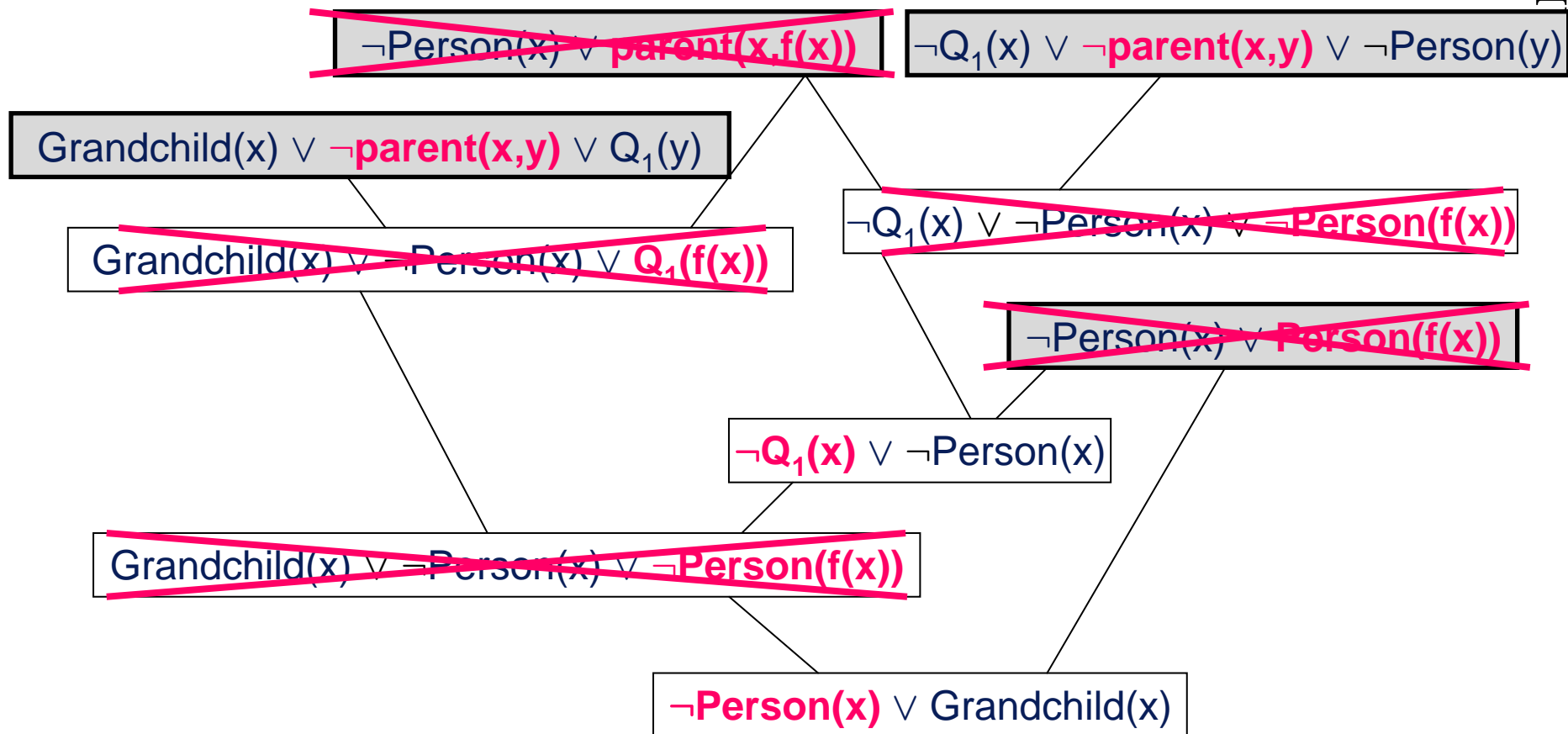


Einfaches Beispiel für Transformation

structural transformation & clausification



Saturierung



Translate to Datalog

Ergebnis: Datalog Programm

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

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!
- Effiziente Unterstützung durch OWL-Beweiser zur Zeit noch nicht erreicht.
- Es wird daran gearbeitet ...

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, 2007. (Manche Kapitel)
- S. Staab, R. Studer (eds.): Handbook on Ontologies, Springer, 2004. (Manche Kapitel) Neue Auflage in Vorbereitung!
- W3C Dokumente (siehe Vorlesungswebseite)
- Literatur auf <http://kaon2.semanticweb.org>.

Inhalt der nächsten Sitzung

- I. OWL – Syntax und allgemeines Verständnis
- II. Logik (Wiederholung)
- III. OWL – Semantische Grundlagen
 - a. Beschreibungslogiken
 - b. Beweistheorie
- IV. Ontologiesprache F-Logik

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 Vrandečić, AIFB Karlsruhe
- and possibly some other people for the cases where I couldn't trace the origin of my files ...