Semantic Web Technologies I

Lehrveranstaltung im WS10/11
Seminar für Computerlinguistik
Universität Heidelberg

PD Dr. Sebastian Rudolph
Institut AIFB
Karlsruher Institut für Technologie

AIFB

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

Ontologie – philosophisch

- Begriff existiert nur in der Einzahl (es gibt also keine „Ontologien“)
- bezeichnet die *„Lehre vom Sein“*
- zu finden bei Aristoteles (Sokrates), Thomas von Aquin, Descartes, Kant, Hegel, Wittgenstein, Heidegger, Quine, ...

Ontologie – informatisch

Gruber (1993):

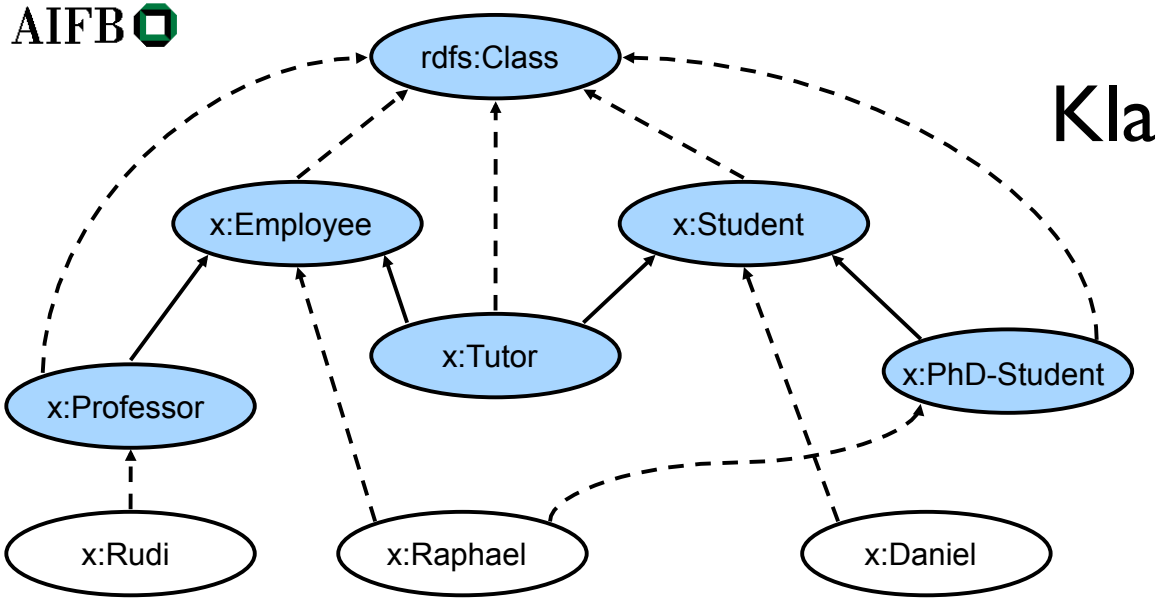
„An Ontology is a

formal specification	⇒ maschinell interpretierbar
of a shared	⇒ beruht auf Konsens
conceptualization	⇒ beschreibt Begrifflichkeiten
of a domain of interest“	⇒ bezogen auf ein „Thema“ (Gegenstandsbereich)

Ontologie – praktisch
Einige Anforderungen

- Instanziierung von Klassen durch Individuen
- Begriffshierarchien (Taxonomien, „Vererbung“): Klassen, Begriffe
- binäre Relationen zwischen Individuen: Properties, Roles
- Eigenschaften von Relationen (z.B. range, transitive)
- Datentypen (z.B. Zahlen): concrete domains
- logische Ausdrucksmittel
- klare Semantik!

RDFS – einfache Ontologien

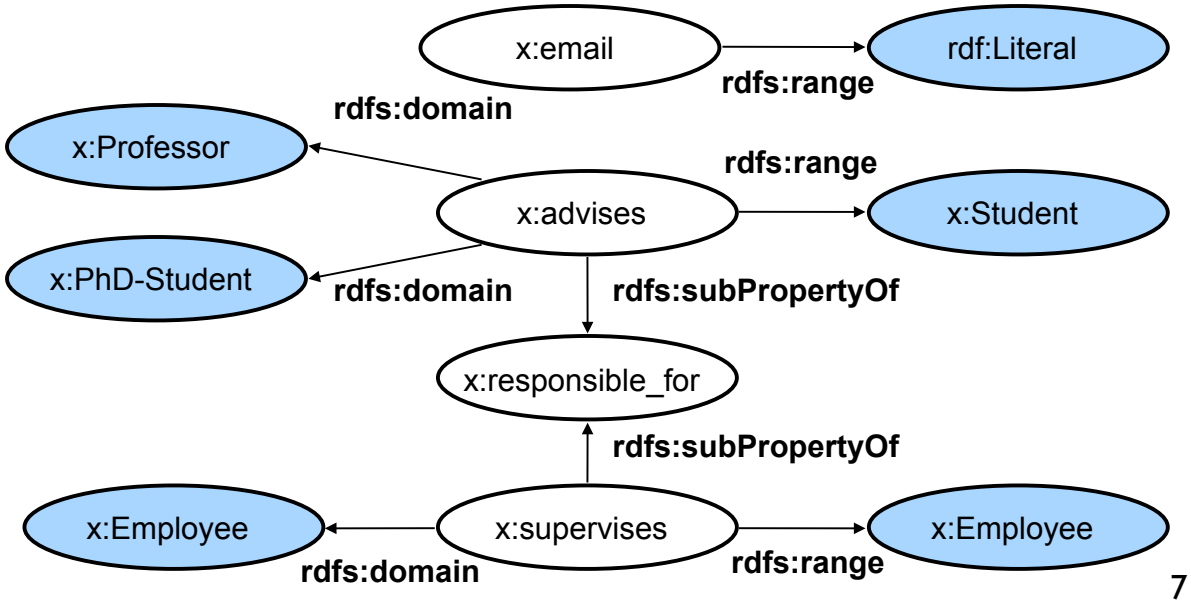


Klassen

subClass

Instanziierung

Relationen



RDF Schema als Ontologiesprache?

- geeignet für einfache Ontologien
- Vorteil: automatisches Schlussfolgern ist relativ effizient
- aber: für komplexere Modellierungen ungeeignet
- Rückgriff auf mächtigere Sprachen, wie
  - OWL
  - F-Logik

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

OWL – Allgemeines

- W3C Recommendation seit 2009 (2. Version)
- Semantisches Fragment von FOL
- Fünf Varianten:
  OWL EL, OWL RL, OWL QL $\subseteq$ OWL DL $\subseteq$ OWL Full
- Keine Reifikation in OWL DL
  ➔ RDFS ist Fragment von OWL Full
- OWL DL ist entscheidbar
  entspricht der Beschreibungslogik $\mathcal{SROIQ}$(D)
- W3C-Dokumente (Vorlesungswebseite) enthalten Details, die hier nicht alle angesprochen werden können.

OWL Dokumente

- sind RDF Dokumente
  (zumindest in der Standard-Syntax; es gibt auch andere)
- bestehen aus
  - Kopf mit allgemeinen Angaben
  - Rest mit der eigentlichen Ontologie

Der Kopf eines OWL Dokumentes

Definition von Namespaces in der Wurzel

```
<rdf:RDF
    xmlns ="http://www.semanticweb-grundlagen.de/beispielontologie#"
    xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd  ="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl  ="http://www.w3.org/2002/07/owl#">
...
</rdf:RDF>
```

Der Kopf eines OWL Dokumentes

Allgemeine Informationen

```
<owl:Ontology rdf:about="">

    <rdfs:comment    rdf:datatype="http://www.w3.org/2001/
    XMLSchema#string“>

    SWRC Ontologie in der Version vom Dezember 2005

    </rdfs:comment>

   <owl:versionInfo>v0.5</owl:versionInfo>

   <owl:imports rdf:resource="http://www.semanticweb-
    grundlagen.de/foo"/>

   <owl:priorVersion      rdf:resource="http://ontoware.org/
    projects/swrc"/>

  </owl:Ontology>
```

Der Kopf eines OWL Dokumentes

von RDFS geerbt

```
rdfs:comment
rdfs:label
rdfs:seeAlso
rdfs:isDefinedBy
```

für Versionierung

```
owl:versionInfo
owl:priorVersion
owl:backwardCompatibleWith
owl:incompatibleWith
owl:DeprecatedClass
owl:DeprecatedProperty
```

außerdem

```
owl:imports
```

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

Klassen, Rollen und Individuen

Die drei Bausteine von Ontologieaxiomen.

Klassen

- Vergleichbar mit Klassen in RDFS

Individuen

- Vergleichbar mit Objekten in RDFS

Rollen

- Vergleichbar mit Properties in RDFS

Klassen

Definition

```
<owl:Class rdf:ID="Professor"/>
```

vordefiniert:

```
owl:Thing
owl:Nothing
```

Individuen

Definition durch Klassenzugehörigkeit

```
<rdf:Description rdf:ID="RudiStuder">
<rdf:type rdf:resource="#Professor"/>
</rdf:Description>
```

gleichbedeutend:

```
<Professor rdf:ID="RudiStuder"/>
```

Abstrakte Rollen

abstrakte Rollen werden definiert wie Klassen

```
<owl:ObjectProperty
    rdf:ID="Zugehoerigkeit"/>
```

Domain und Range abstrakter Rollen

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>
```

KONKRETE ROLLEN

konkrete Rollen haben Datentypen im Range

```
<owl:DatatypeProperty rdf:ID="Vorname"/>
```

Domain und Range konkreter Rollen

```
<owl:DatatypeProperty rdf:ID="Vorname">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range  rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Viele XML Datentypen können verwendet werden.
Im Standard vorgeschrieben sind `integer` und `string`.

Individuen und Rollen


```
<Person rdf:ID="RudiStuder">
  <Zugehoerigkeit rdf:resource="#AIFB"/>
  <Zugehoerigkeit rdf:resource="#ontoprise"/>
  <Vorname rdf:datatype="&xsd;string">Rudi</Vorname>
</Person>
```

Rollen sind im allgemeinen nicht funktional.

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Professor">

  <rdfs:subClassOf
    rdf:resource="#Fakultaetsmitglied"/>

</owl:Class>

<owl:Class rdf:ID="Fakultaetsmitglied">

  <rdfs:subClassOf rdf:resource="#Person"/>

</owl:Class>
```

Es folgt durch Inferenz, dass `Professor` eine Subklasse von `Person` ist.

Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Professor">
  <rdfs:subClassOf
   rdf:resource="#Fakultaetsmitglied"/>
</owl:Class>
<owl:Class rdf:ID="Buch">
  <rdfs:subClassOf rdf:resource="#Publikation"/>
</owl:Class>
<owl:Class rdf:about="#Fakultaetsmitglied">
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Es folgt durch Inferenz, dass `Professor` und `Buch` ebenfalls disjunkte Klassen sind.

Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Buch">

  <rdfs:subClassOf rdf:resource="#Publikation"/>

</owl:Class>



<owl:Class rdf:about="#Publikation">

  <owl:equivalentClass
    rdf:resource="#Publication"/>

</owl:Class>
```

Es folgt durch Inferenz, dass `Buch` eine Subklasse von `Publication` ist.

Individuen und Klassenbeziehungen


```
<Buch rdf:ID="SemanticWebGrundlagen">
  <Autor rdf:resource="#PascalHitzler"/>
  <Autor rdf:resource="#MarkusKrötzsch"/>
  <Autor rdf:resource="#SebastianRudolph"/>
  <Autor rdf:resource="#YorkSure"/>
</Buch>
<owl:Class rdf:about="#Buch">
  <rdfs:subClassOf rdf:resource="#Publikation"/>
</owl:Class>
```

Es folgt durch Inferenz, dass `SemanticWebGrundlagen` eine `Publikation` ist.

Beziehungen zwischen Individuen

```
<Professor rdf:ID="RudiStuder"/>

<rdf:Description rdf:about="#RudiStuder">

   <owl:sameAs
     rdf:resource="#ProfessorStuder"/>

</rdf:Description>
```

Es folgt durch Inferenz, dass `ProfessorStuder` ein `Professor` ist.

Verschiedenheit von Individuen mittels

`owl:differentFrom`.

Beziehungen zwischen Individuen

```
<owl:AllDifferent>

  <owl:distinctMembers
     rdf:parseType="Collection">

    <Person rdf:about="#RudiStuder"/>

    <Person rdf:about="#YorkSure"/>

    <Person rdf:about="#PascalHitzler"/>

  </owl:distinctMembers>

</owl:AllDifferent>
```

Abgekürzte Schreibweise anstelle der Verwendung von mehreren `owl:differentFrom`.

Der Einsatz von `owl:AllDifferent` und `owl:distinctMembers` ist nur dafür vorgesehen.

Abgeschlossene Klassen

```
<owl:Class rdf:about="#SekretaerinnenVonStuder">
  <owl:oneOf rdf:parseType="Collection">
    <Person rdf:about="#GiselaSchillinger"/>
    <Person rdf:about="#BeateKuehner"/>
  </owl:oneOf>
</owl:Class>
```

Dies besagt, dass es nur **genau diese beiden** `SekretaerinnenVonStuder` gibt.

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

Logische Klassenkonstruktoren

- logisches Und (Konjunktion):
  `owl:intersectionOf`
- logisches Oder (Disjunktion):
  `owl:unionOf`
- logisches Nicht (Negation):
  `owl:complementOf`
- Werden verwendet, um komplexe Klassen aus einfachen Klassen zu konstruieren.

Konjunktion


```
<owl:Class rdf:about="#SekretaerinnenVonStuder">
  <owl:equivalentClass>
    <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Sekretaerinnen"/>
      <owl:Class rdf:about="#AngehoerigeAGStuder"/>
    </owl:intersectionOf>
    <owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Es folgt z.B. durch Inferenz, dass alle `SekretaerinnenVonStuder` auch `Sekretaerinnen` sind.

Disjunktion

```
<owl:Class rdf:about="#Professor">
  <rdfs:subClassOf>
    <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#aktivLehrend"/>
      <owl:Class rdf:about="#imRuhestand"/>
    </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

Negation


```
<owl:Class rdf:about="#Fakultaetsmitglied">
  <rdfs:subClassOf>
    <owl:Class>
    <owl:complementOf rdf:resource="#Publikation"/>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

semantisch äquivalente Aussage:

```
<owl:Class rdf:about="#Fakultaetsmitglied">
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Rolleneinschränkungen (allValuesFrom)

dienen der Definition komplexer Klassen durch Rollen

```
<owl:Class rdf:ID="Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:allValuesFrom rdf:resource="#Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

D.h. *alle* Prüfer einer Prüfung müssen Professoren sein.

Rolleneinschränkungen (someValuesFrom)

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom rdf:resource="#Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

D.h. jede Prüfung muss *mindestens einen* Prüfer haben.

Rolleneinschränkungen (Kardinalitäten)


```
<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.

Rolleneinschränkungen (Kardinalitäten)


```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatThema"/>
      <owl:minCardinality
     rdf:datatype="&xsd;nonNegativeInteger">
     3
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung muss sich über *mindestens drei* Themengebiete erstrecken.

Rolleneinschränkungen (Kardinalitäten)


```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatThema"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
       3
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung muss sich über *genau drei* Themengebiete erstrecken.

Rolleneinschränkungen (hasValue)

```
<owl:Class rdf:ID="PruefungBeiStuder">
  <rdfs:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:hasValue rdf:resource="#RudiStuder"/>
    </owl:Restriction>
  </rdfs:equivalentClass>
</owl:Class>
```

`owl:hasValue` verweist immer auf eine konkrete Instanz. Dies ist äquivalent zum Beispiel auf der nächsten Folie.

Rolleneinschränkungen (hasValue)


```
<owl:Class rdf:ID="PruefungBeiStuder">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom>
        <owl:oneOf rdf:parseType="Collection">
          <owl:Thing rdf:about="#RudiStuder"/>
        </owl:oneOf>
      </owl:someValuesFrom
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

Rollenbeziehungen

```
<owl:ObjectProperty rdf:ID="hatPruefer">

  <rdfs:subPropertyOf
    rdf:resource="#hatAnwesenden"/>

</owl:ObjectProperty>
```

Ebenso: `owl:equivalentProperty`

Rollen können auch invers zueinander sein:

```
<owl:ObjectProperty rdf:ID="hatPruefer">

  <owl:inverseOf rdf:resource="#prueferVon"/>

</owl:ObjectProperty>
```

Rolleneigenschaften

- Domain
- Range
- Transitivität, d.h.
  r(a,b) und r(b,c) impliziert r(a,c)
- Symmetrie, d.h.
  r(a,b) impliziert r(b,a)
- Funktionalität
  r(a,b) und r(a,c) impliziert b=c
- Inverse Funktionalität
  r(a,b) und r(c,b) impliziert a=c

Domain und Range

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>
```

ist gleichbedeutend mit dem Folgenden:

```
<owl:Class rdf:about="\&owl;Thing">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Zugehoerigkeit"/>
      <owl:allValuesFrom rdf:resource="#Organisation"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Domain und Range: Vorsicht!

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>
<Zahl rdf:ID="Fuenf">
  <Zugehoerigkeit rdf:resource="#Primzahlen"/>
</Zahl>
```

Es folgt nun, dass `Primzahlen` eine Organisation ist!

Rolleneigenschaften


```
<owl:ObjectProperty rdf:ID="hatKollegen">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hatProjektleiter">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="istProjektleiterFuer">
  <rdf:type
  rdf:resource="&owl;InverseFunctionalProperty"/>
</owl:ObjectProperty>
<Person rdf:ID="YorkSure">
  <hatKollegen rdf:resource="#PascalHitzler"/>
  <hatKollegen rdf:resource="#AnupriyaAnkolekar"/>
  <istProjektleiterFuer rdf:resource="#SEKT"/>
</Person>
<Projekt rdf:ID="SmartWeb">
  <hatProjektleiter rdf:resource="#PascalHitzler"/>
  <hatProjektleiter rdf:resource="#HitzlerPascal"/>
</Projekt>
```

Folgerungen aus dem Beispiel

- `AnupriyaAnkolekar hatKollegen YorkSure`
- `AnupriyaAnkolekar hatKollegen PascalHitzler`
- `PascalHitzler owl:sameAs HitzlerPascal`

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- Anfragen an OWL-Ontologien

Terminologische Anfragen an OWL (nur Klassen und Rollen)

- Klassenäquivalenz
- Subklassenbeziehung
- Disjunktheit von Klassen
- globale Konsistenz (Erfüllbarkeit, Widerspruchsfreiheit)
- Klassenkonsistenz: Eine Klasse ist *inkonsistent*, wenn sie äquivalent zu `owl:Nothing` ist - dies deutet oft auf einen Modellierungsfehler hin:

```
<owl:Class rdf:about="#Buch">
  <owl:subClassOf rdf:resource="#Publikation"/>
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Assertionale Anfragen an OWL (mit Individuen)

- Instanzüberprüfung: Gehört gegebenes Individuum zu gegebener Klasse?
- Suche nach allen Individuen, die in einer Klasse enthalten sind.
- Werden zwei gegebene Individuen durch Rolle verknüpft?
- Suche nach allen Individuenpaaren, die durch eine Rolle verknüpft sind.
- ...Vorsicht: es wird nur nach „beweisbaren“ Antworten gesucht!

OWL Werkzeuge

- Editoren
  - Protegé, http://protege.stanford.edu
  - SWOOP, http://www.mindswap.org/2004/SWOOP/
  - OWL Tools, http://owltools.ontoware.org/
- Inferenzmaschinen
  - Pellet, http://clarkparsia.com/pellet/
  - HermiT http://hermit-reasoner.com/
  - KAON2, http://kaon2.semanticweb.org
  - FACT++, http://owl.man.ac.uk/factplusplus/
  - Racer, http://www.racer-systems.com/

Es gibt noch mehr…

- weitere Features von OWL
  - mehr Klassenkonstruktoren
  - mehr für Modellierungsmittel für Properties
  - Datentypen
  - OWL Profiles

Weiterführende Literatur

- `http://www.w3.org/2007/OWL/`
  zentrale W3C Webseite für OWL.
- `http://www.w3.org/TR/owl2-overview/`
  Überblick über OWL.
- `http://www.w3.org/TR/owl2-syntax/`
  vollständige Beschreibung der OWL-Sprachkomponenten.
- `http://www.w3.org/TR/owl2-primer/`
  zeigt, wie OWL zur Wissensmodellierung verwendet werden kann.
- `http://www.w3.org/TR/owl2-direct-semantics/`
  beschreibt die Semantik von OWL, die wir auf andere Weise später behandeln werden. Es beschreibt außerdem die abstrakte Syntax für OWL DL, die wir hier später noch ansprechen.

Semantic Web Technologies I

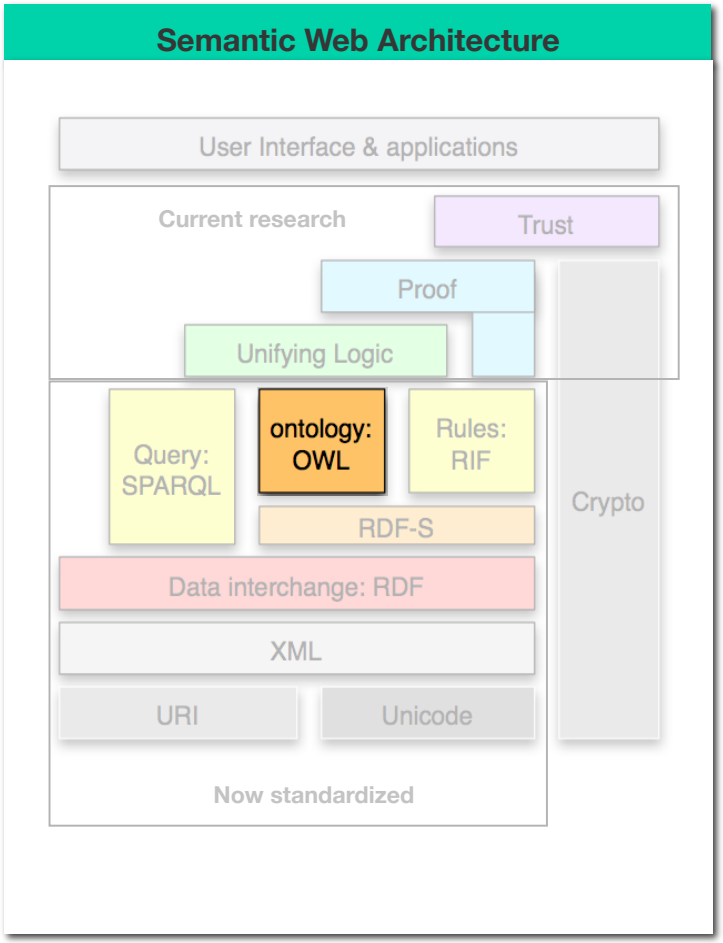
Lehrveranstaltung im WS11/12
Seminar für Computerlinguistik
Universität Heidelberg

PD Dr. Sebastian Rudolph
Institut AIFB
Karlsruher Institut für Technologie

AIFB

OWL – Semantik & Reasoning

Dr. Sebastian Rudolph



Agenda

- Beschreibungslogiken
- $\mathcal{ALC}$
- OWL als $\mathcal{SROIQ}$(D)
- Inferenzprobleme
- Tableau-Beweiser

Agenda

- Beschreibungslogiken
- $\mathcal{ALC}$
- OWL als $\mathcal{SROIQ}$(D)
- Inferenzprobleme
- Tableau-Beweiser

Beschreibungslogiken

- engl.: description logics (DLs)
- Fragmente von FOL
- meist entscheidbar
- vergleichsweise ausdrucksstark
- entwickelt aus semantischen Netzwerken
- enge Verwandtschaft mit Modallogiken

- W3C Standard OWL DL basiert auf der Beschreibungslogik $\mathcal{SROIQ}$ (D)
- wir besprechen zunächst $\mathcal{ALC}$ (Basis für komplexere DLs)

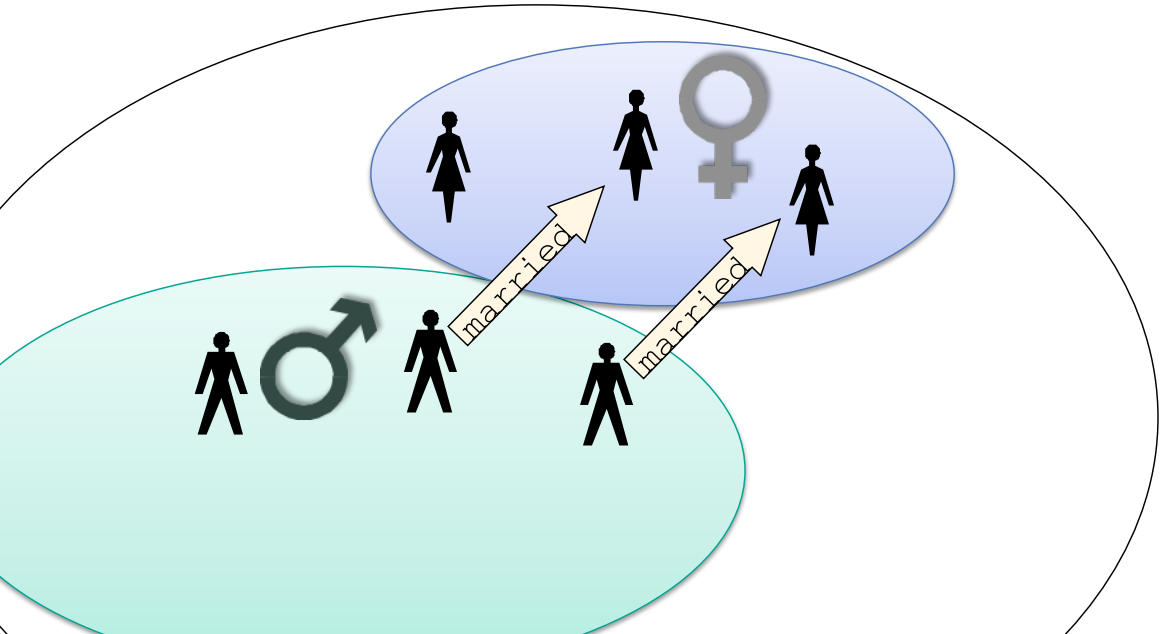
Agenda

- Beschreibungslogiken
- $\mathcal{ALC}$
- OWL als $\mathcal{SROIQ}$(D)
- Inferenzprobleme
- Tableau-Beweiser

ALC - Grundbausteine

Grundbausteine:

- Individuennamen
- Klassennamen (auch als Konzepte bezeichnet)
- Rollennamen



ALC - Grundbausteine

Grundbausteine:

- Individuennamen
- Klassennamen (auch als Konzepte bezeichnet)
- Rollennamen

Angabe von Klassen- und Rolleninstanzen:

Professor(RudiStuder)

- Individuum RudiStuder ist in Klasse Professor

Zugehoerigkeit(RudiStuder,AIFB)

- RudiStuder ist dem AIFB zugehörig

ALC – Subklassenbeziehungen

Professor $\sqsubseteq$ Fakultaetsmitglied

- „Jeder Professor ist ein Fakultätsmitglied.“
- entspricht $(\forall x)(\text{Professor}(x) \rightarrow \text{Fakultaetsmitglied}(x))$
- entspricht owl:subClassOf

Professor $\equiv$ Fakultaetsmitglied

- „Die Fakultätsmitglieder sind genau die Professoren.“
- entspricht $(\forall x)(\text{Professor}(x) \leftrightarrow \text{Fakultaetsmitglied}(x))$
- entspricht owl:equivalentClass

ALC – komplexe Klassen

Konjunktion $\sqcap$ entspricht owl:intersectionOf

Disjunktion $\sqcup$ entspricht owl:unionOf

Negation $\neg$ entspricht owl:complementOf

Beispiel:

$$\text{Professor} \sqsubseteq (\text{Person} \sqcap \text{Unversitaetsangehoeriger}) \sqcup (\text{Person} \sqcap \neg\text{Doktorand})$$

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

ALC – Quantoren auf Rollen

Pruefung $\sqsubseteq \forall$hatPruefer.Professor

- „Jede Prüfung hat nur Professoren als Prüfer.“
- $(\forall x)(\text{Pruefung}(x) \rightarrow (\forall y)(\text{hatPruefer}(x,y) \rightarrow \text{Professor}(y)))$
- entspricht owl:allValuesFrom

Pruefung $\sqsubseteq \exists$hatPruefer.Person

- „Jede Prüfung hat mindestens einen Prüfer.“
- $(\forall x)(\text{Pruefung}(x) \rightarrow (\exists y)(\text{hatPruefer}(x,y) \wedge \text{Person}(y)))$
- entspricht owl:someValuesFrom

Weitere OWL-Konstrukte in ALC

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

ALC – Formale Syntax

- Folgende Syntaxregeln erzeugen Klassen in $\mathcal{ALC}$. Dabei ist A eine atomare Klasse und R eine Rolle.

  $C,D \rightarrow A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$

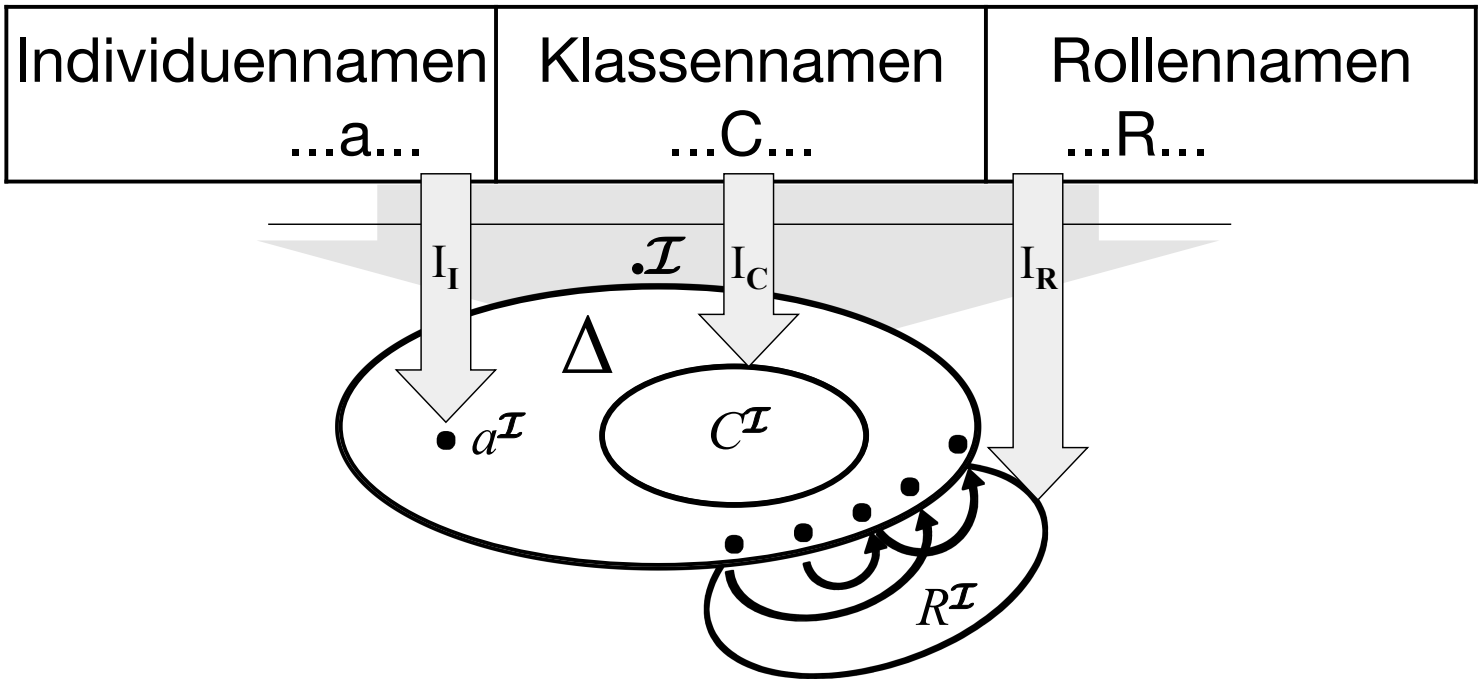
- Eine *$\mathcal{ALC}$-TBox* besteht aus Aussagen der Form $C \sqsubseteq D$ und $C \equiv D$, wobei C, D Klassen sind.
- Eine *$\mathcal{ALC}$-ABox* besteht aus Aussagen der Form C(a) und R(a,b), wobei C eine komplexe Klasse, R eine Rolle und a,b Individuen sind.
- Eine *$\mathcal{ALC}$-Wissensbasis* besteht aus einer ABox und einer TBox.

ALC – Semantik (Interpretationen)

- wir definieren modelltheoretische Semantik für $\mathcal{ALC}$ (d.h. Folgerung wird über Interpretationen definiert)
- eine Interpretation $\mathcal{I}=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ besteht aus
  - einer Menge $\Delta^{\mathcal{I}}$, genannt Domäne und
  - einer Funktion $\cdot^{\mathcal{I}}$, die abbildet von
    - Individuennamen a auf Domänenelemente $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
    - Klassennamen C auf Mengen von Domänenelementen $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
    - Rollennamen R auf Mengen von Paaren von Domänenelementen $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

ALC – Semantik (Interpretationen)

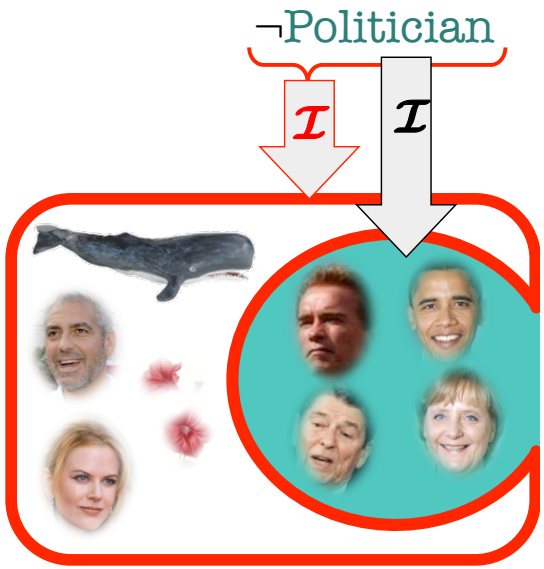
- schematisch:

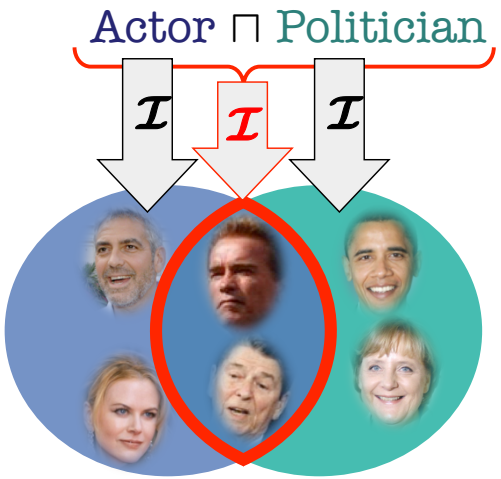


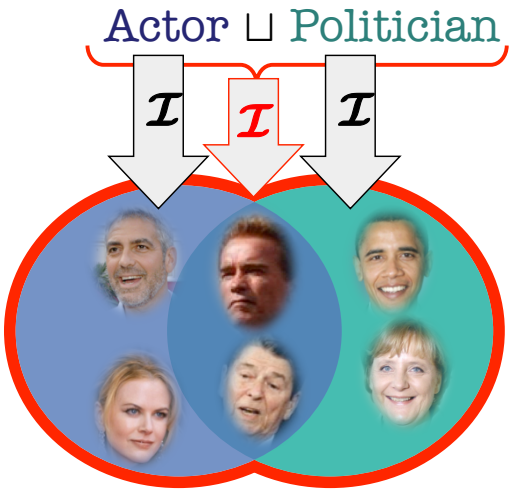
ALC-Semantik (komplexe Klassen)

- wird auf komplexe Klassen erweitert:
  - $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ $\quad\quad$ $\bot^{\mathcal{I}} = \varnothing$
  - $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ $\quad\quad$ $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
  - $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
  - $\forall R.C = \{ x \mid \forall (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}} \}$
  - $\exists R.C = \{ x \mid \exists (x,y) \in R^{\mathcal{I}}$ mit $y \in C^{\mathcal{I}} \}$
- ...und schließlich auf Axiome:
  - $C(a)$ gilt, wenn $a^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $R(a,b)$ gilt, wenn $(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}$
  - $C \sqsubseteq D$ gilt, wenn $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
  - $C \equiv D$ gilt, wenn $C^{\mathcal{I}} = D^{\mathcal{I}}$

Boolesche Konstruktoren

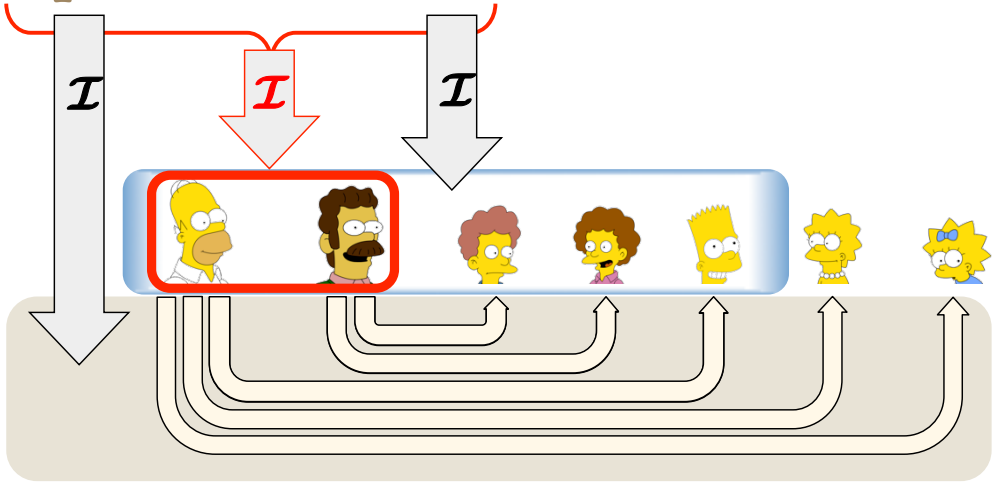




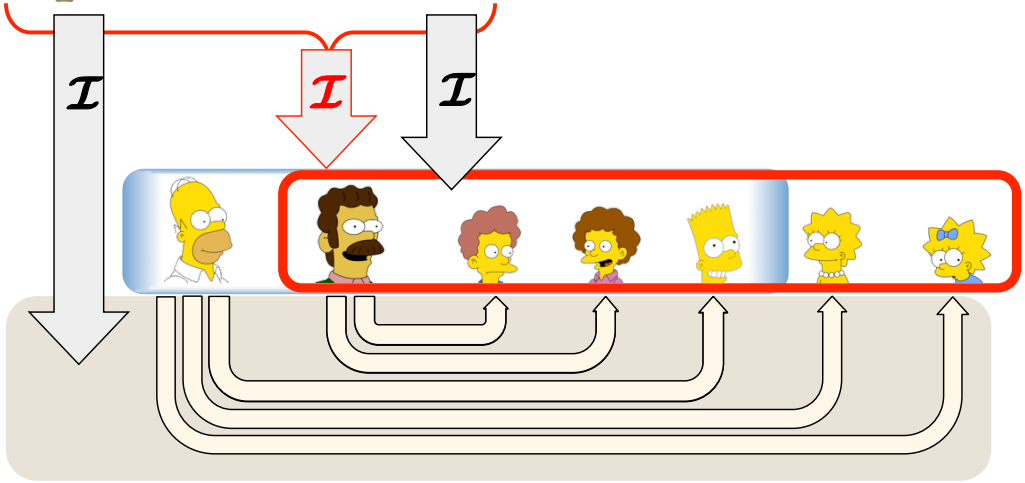


Rollenrestriktionen

$\exists parentOf.Male$

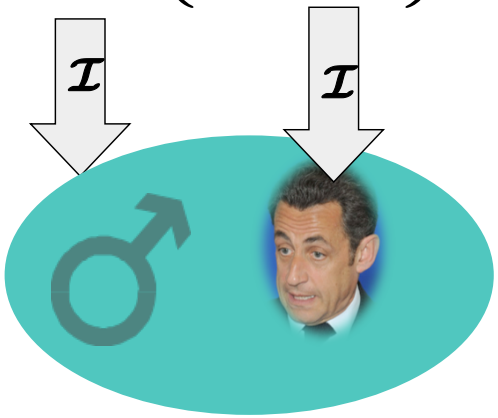


$\forall parentOf.Male$

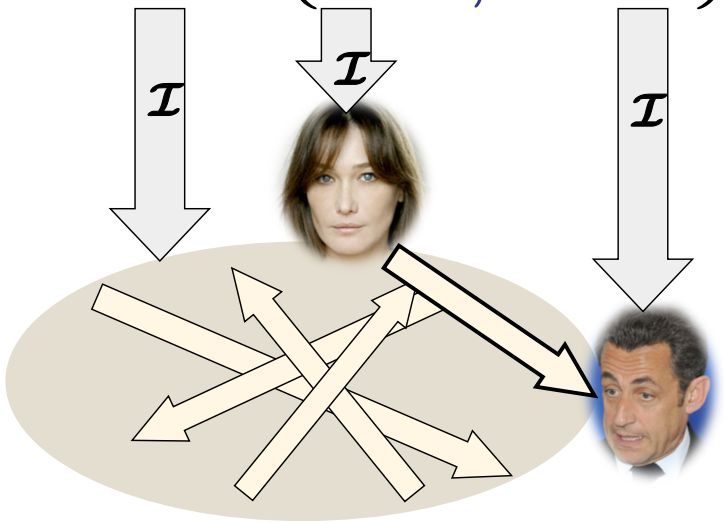


Axiome

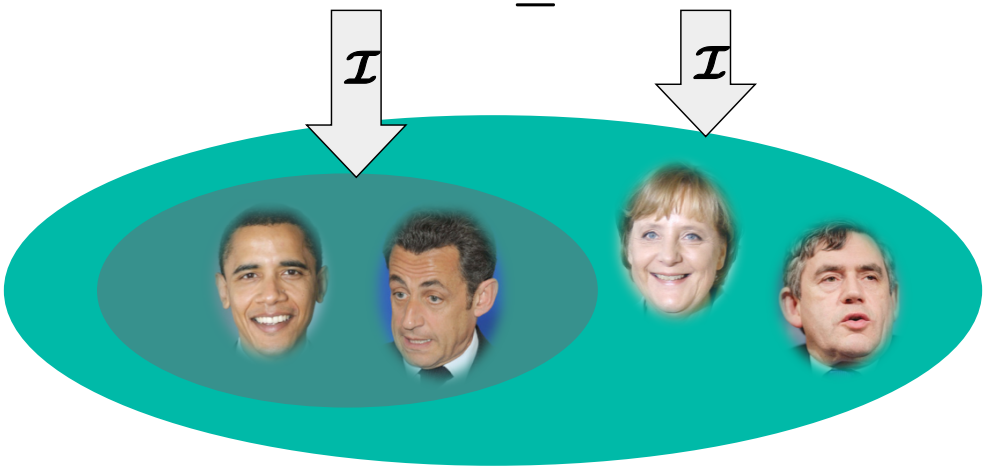
Male(nicolas)



married(carla,nicolas)



President $\sqsubseteq$ Politician



ALC – alternative 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.

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

DL-Wissensbasen

- DL Wissensbasen bestehen aus 2 Teilen:
  - TBox: Axiome, die die Struktur der zu modellierenden Domäne beschreiben (konzeptionelles Schema):
    - **HappyFather** ≡ **Man** ⊓ ∃**hasChild.Female** ⊓ **…**
    - **Elephant** ⊑ **Animal** ⊓ **Large** ⊓ **Grey**
    - **transitive(hasAncestor)**
  - Abox: Axiome, die konkrete Situationen (Daten) beschreiben:
    - **HappyFather(John)**
    - **hasChild(John, Mary)**

Einfaches Beispiel

Terminologisches Wissen (*TBox*):

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

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

Wissen um Individuen (*ABox*):

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)

Semantik und logische Konsequenzen klar, da übersetzbar nach FOL.

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
- owl:allValuesFrom, owl:someValuesFrom
- rdfs:range und rdfs:domain

Agenda

- Beschreibungslogiken
- $\mathcal{ALC}$
- OWL als $\mathcal{SROIQ}$(D)
- Inferenzprobleme
- Tableau-Beweiser

OWL als SROIQ(D) - Individuen

owl:sameAs

- gibt an dass zwei Individuennamen dasselbe Domänenelement bezeichnen
- DL: a=b
- FOL: Erweiterung durch Gleichheitsprädikat

owl:differentFrom

- gibt an dass zwei Individuennamen unterschiedliche Domänenelemente bezeichnen
- DL: a≠b
- FOL: ¬(a=b)

OWL als SHROIQ(D) – abgeschlossene Klassen

Abgeschlossene Klassen

- owl:oneOf
  - definiert eine Klasse durch vollständige Aufzählung ihrer Instanzen
  - DL: $C \equiv \{a,b,c\}$
  - FOL: $(\forall x)\ (C(x) \leftrightarrow (x=a \vee x=b \vee x=c))$

- owl:hasValue
  - „erzwingt“ Rolle zu einem bestimmten Individuum
  - darstellbar mittels owl:someValuesFrom und owl:oneOf

OWL als SROIQ(D) – Kardinalität

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: Pruefung $\sqsubseteq \leq 2$ hatPruefer

In FOL: (P… Prüfung, h…hatPruefer)

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

Entsprechend für die anderen Zahlenrestriktionen

OWL ALS SROIQ(D) – SELF-LOOPS

Self-Restriktionen

- owl:hasSelf
  - definiert eine Klasse der mit sich selbst verknüpften Instanzen
  - DL: $C \equiv \exists R.Self$
  - FOL: $(\forall x)\ (C\ (x) \leftrightarrow R(x,x))$

OWL ALS SROIQ(D) – ROLLEN

rdfs:subPropertyOf
- spezifiziert Unterrolle-Oberrolle-Beziehung
- DL: $R \sqsubseteq S$
- FOL: $(\forall x)(\forall y)(R(x,y) \rightarrow S(x,y))$

Entsprechend Rollenäquivalenz

inverse Rollen: $R \equiv S^-$
- Konstruktor für Rollen zur Bildung der Inversen
- FOL: $(\forall x)(\forall y)(R(x,y) \leftrightarrow S(y,x))$

Rollenketten: $R_1 \circ \ldots \circ R_n \sqsubseteq S$
- FOL: $(\forall x_1 \ldots x_{n+1})\ (R_1(x_1,x_2) \wedge \ldots \wedge R_n(x_n,x_{n+1}) \rightarrow S(x_1,x_{n+1}))$

Rollendisjunktheit: Disj(R,S)
- FOL: $(\forall x)(\forall y)(R(x,y) \rightarrow \neg S(y,x))$

OWL als SROIQ(D) – Rollen

transitive Rollen: Trans(R)
- gibt an, dass eine Rolle transitiv ist
- Ausdrückbar durch Rollenketten: $R \circ R \sqsubseteq R$
- FOL: $(\forall x)(\forall y)(\forall z)(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$

Symmetrie: $R \equiv R^{-}$
- ausdrückbar als Rollenäquivalenz mit der Inversen

Funktionalität: $\top \sqsubseteq \leq 1\ R$
- ausdrückbar durch Klasseninklusion und Kardinalitätsrestriktion

Inverse Funktionalität: $\top \sqsubseteq \leq 1\ R^{-}$
- ausdrückbar wie Funktionalität zuzüglich inverser Rolle

OWL als SROIQ(D) – Überblick

Erlaubt sind:

- $\mathcal{ALC}$
- Gleichheit und Ungleichheit zwischen Individuen
- Abgeschlossene Klassen
- Zahlenrestriktionen
- Subrollen und Rollenäquivalenz
- Inverse und transitive Rollen
- Datentypen

DLs – Nomenklatur

- $\mathcal{ALC}$: Attribute Language with Complement
- $\mathcal{S}$: $\mathcal{ALC}$ + Rollentransitivität
- $\mathcal{H}$: Subrollenbeziehung
- $\mathcal{O}$: abgeschlossene Klassen
- $\mathcal{I}$: inverse Rollen
- $\mathcal{N}$: Zahlenrestriktionen ≤n R etc.
- $\mathcal{Q}$: Qualifizierende Zahlenrestriktionen ≤n R.C etc.
- (D): Datentypen
- $\mathcal{F}$: Funktionale Rollen
- $\mathcal{R}$: Rollenverkettung, Self-Loops, Rollendisjunktheit

- OWL DL ist $\mathcal{SROIQ}$ (D)

Semantic Web TECHNOLOGIES

AIFB

DL-Syntax - Übersicht

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

	Roles	
	Atomic	R
$\mathcal{I}$	Inverse	R^-

Ontology (=Knowledge Base)

Concept Axioms (TBox)	
Subclass	C $\sqsubseteq$ D
Equivalent	C $\equiv$ D

	Role Axioms (RBox)	
$\mathcal{H}$	Subrole	R $\sqsubseteq$ S
$\mathcal{S}$	Transitivity	Trans(S)
$\mathcal{R}$	Role Chain	R $\circ$ R′ $\sqsubseteq$ S
	R. Disjointness	Disj(S)

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

$\mathcal{S}$ = $\mathcal{ALC}$ + Transitivity **OWL DL = $\mathcal{SROIQ}$(D)** (D: concrete domain)

DL-Syntax - Klassenkonstruktoren

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	$\exists^{\geqslant n} y.P(x,y)$

Beliebig komplexes Schachteln von Konstruktoren erlaubt:

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

DL-Syntax - 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
sameAs	$\{x_1\} \equiv \{x_2\}$	{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 ($\sqsubseteq$) 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))

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.

		DL answers	Prolog
child(Bill,Bob) Man(Bob)	? ⊨ ∀child.Man(Bill)	don't know	yes
≤ 1 child.T(Bill)	? ⊨ ∀child.Man(Bill)	yes	

Now we know everything about Bill's children.

Agenda

- Beschreibungslogiken
- $\mathcal{ALC}$
- OWL als $\mathcal{SROIQ}$(D)
- Inferenzprobleme
- Tableau-Beweiser

Wichtige Inferenzprobleme

- Globale Konsistenz der Wissensbasis — $KB \models \mathbf{false}$?
  - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz — $C \equiv \bot$?
  - Muss Klasse C leer sein?
- Klasseninklusion (Subsumption) — $C \sqsubseteq D$?
  - Strukturierung der Wissensbasis
- Klassenäquivalenz — $C \equiv D$?
  - Sind zwei Klassen eigentlich dieselbe?
- Klassendisjunktheit — $C \sqcap D = \bot$?
  - Sind zwei Klassen disjunkt?
- Klassenzugehörigkeit — $C(a)$?
  - Ist Individuum a in der Klasse C?
- Instanzgenerierung (Retrieval) „alle X mit C(X) finden“
  - Finde alle (bekannten!) 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!

Rückführung auf Unerfüllbarkeit

- Wir werden das Tableauverfahren für OWL DL abwandeln.
  - Genauer: Wir werden nur $\mathcal{ALC}$ behandeln.
- Tableau- und Resolutionsverfahren zeigen Unerfüllbarkeit einer Theorie.

- $\rightarrow$ Rückführung der Inferenzprobleme auf das Finden von Inkonsistenten in der Wissensbasis, d.h. zeigen der Unerfüllbarkeit der Wissensbasis!

Rückführung auf Unerfüllbarkeit

- Klassenkonsistenz — $C \equiv \bot$ gdw
  - KB $\cup$ {C(a)} unerfüllbar (a neu)
- Klasseninklusion (Subsumption) — $C \sqsubseteq D$ gdw
  - KB $\cup$ {C $\sqcap$ ¬D(a)} unerfüllbar (a neu)
- Klassenäquivalenz — $C \equiv D$ gdw
  - $C \sqsubseteq D$ und $D \sqsubseteq C$
- Klassendisjunktheit — $C \sqcap D = \bot$ gdw
  - KB $\cup$ {(C $\sqcap$ D)(a)} unerfüllbar (a neu)
- Klassenzugehörigkeit — C(a) gdw
  - KB $\cup$ {¬C(a)} unerfüllbar (a neu)
- Instanzgenerierung (Retrieval) — alle C(X) finden
  - Prüfe Klassenzugehörigkeit für alle Individuen.
  - Schwerer, dies gut zu implementieren!

Agenda

- Beschreibungslogiken
- $\mathcal{ALC}$
- OWL als $\mathcal{SROIQ}$(D)
- Inferenzprobleme
- Tableau-Beweiser

Tableau – Transformation in NNF

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.

Tableau – Transformation in NNF

NNF(C) = C, falls C atomar ist

NNF(¬C) = ¬C, falls C atomar ist

NNF(¬¬C) = NNF(C)

NNF(C ⊔ D) = NNF(C) ⊔ NNF(D)

NNF(C ⊓ D) = NNF(C) ⊓ NNF(D)

NNF(¬(C ⊔ D)) = NNF(¬C) ⊓ NNF(¬D)

NNF(¬(C ⊓ D)) = NNF(¬C) ⊔ NNF(¬D)

NNF(∀R.C) = ∀ R.NNF(C)

NNF(∃R.C) = ∃ R.NNF(C)

NNF(¬∀R.C) = ∃R.NNF(¬C)

NNF(¬∃R.C) = ∀R.NNF(¬C)

W und NNF(W) sind logisch äquivalent.

Tableau – NNF – Beispiel

- $P \sqsubseteq (E \sqcap U) \sqcup \neg(\neg E \sqcup D)$.
- In Negationsnormalform:
- $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$.

Naives Tableauverfahren

Rückführung auf Unerfüllbarkeit/Widerspruch

Idee:

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

Tableau – einfaches Beispiel

C(a)

(¬ C ⊓ D)(a)

¬C(a) ist logische Konsequenz:

2. Formel in FOL: ¬C(a)∧ D(a)

daraus folgt u.a. ¬C(a)

Widerspruch ist gefunden.

Tableau – 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), ¬C(a), R(a,b).
- *Tableau*: Endliche Menge von Tableauzweigen.
- Tableauzweig ist *abgeschlossen* wenn er ein Paar widersprüchlicher Aussgen C(a) und ¬C(a) enthält.
- Tableau ist *abgeschlossen*, wenn jeder Zweig von ihm abgeschlossen ist.

Tableau – Erzeugung

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 $C(a)$ fr 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.

Tableau – Beispiel (1/2)

- 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:
  $\{\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D), (P \sqcap \neg E)(a)\}$

Tableau – Beispiel (2/2)

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

Tableau:

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

$P(a)$

$\neg E(a)$

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

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

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

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

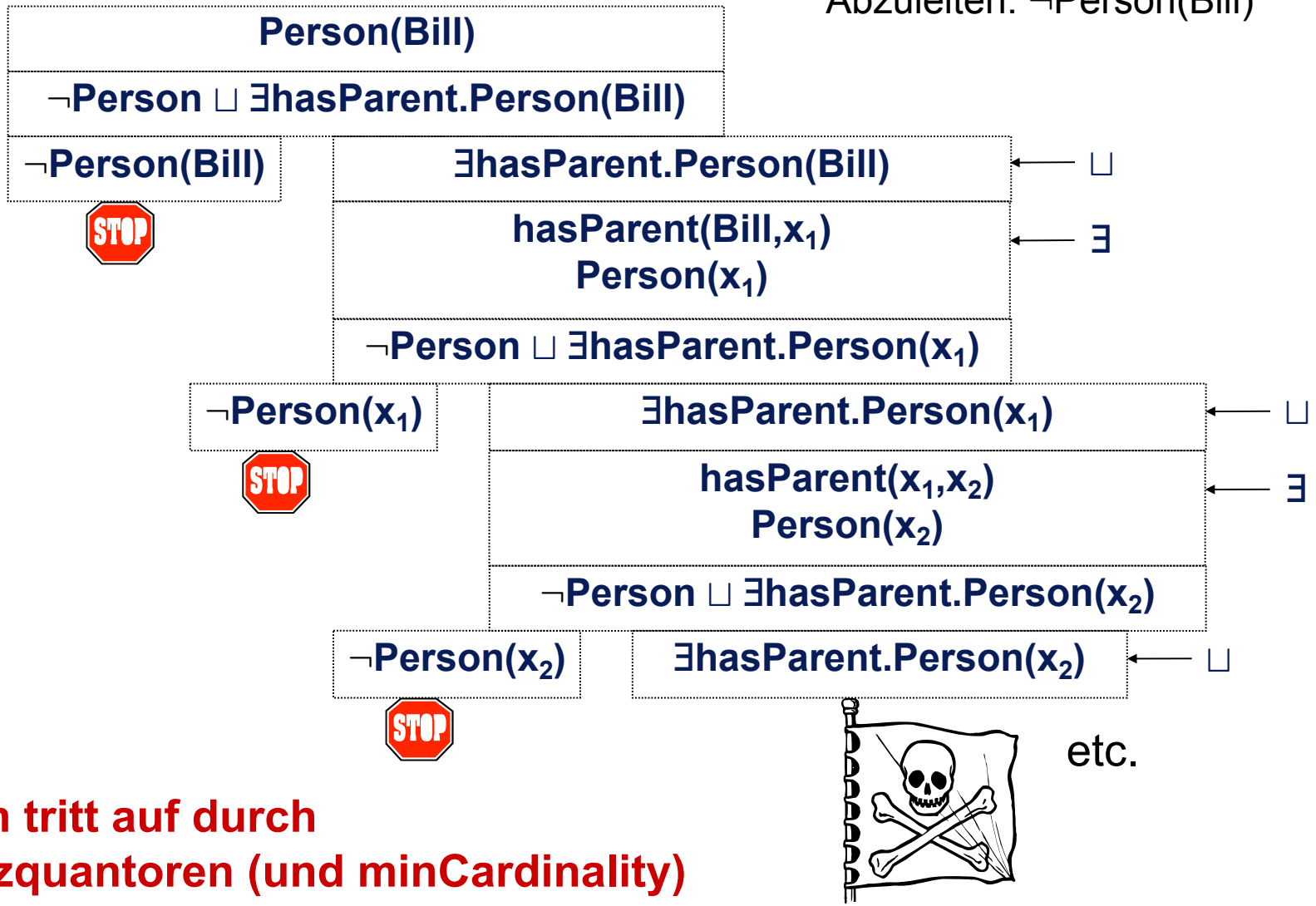
$U(a)$ $\neg D(a)$

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

Tableau – Terminierungsproblem

Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

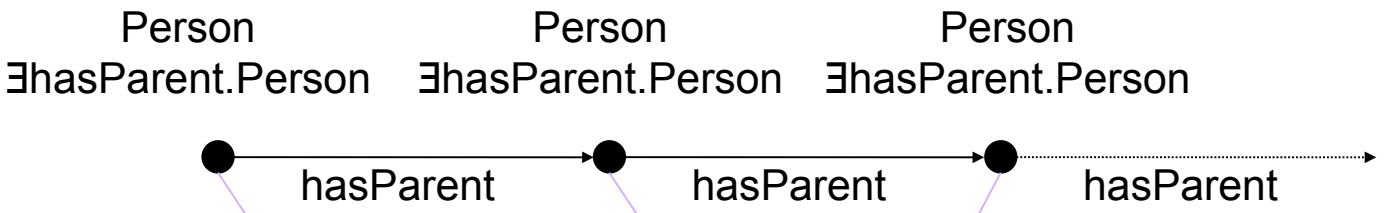


Problem tritt auf durch Existenzquantoren (und minCardinality)

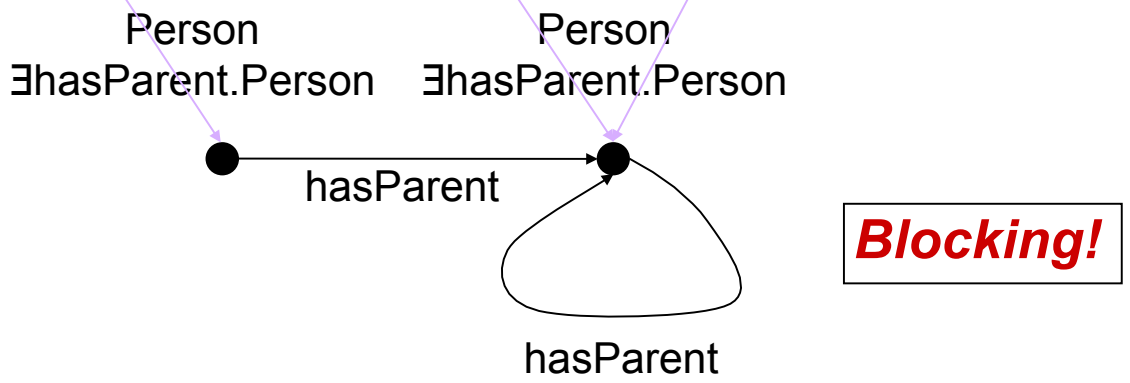
Tableau – Blocking – Idee

AIFB

Wir haben folgendes konstruiert:



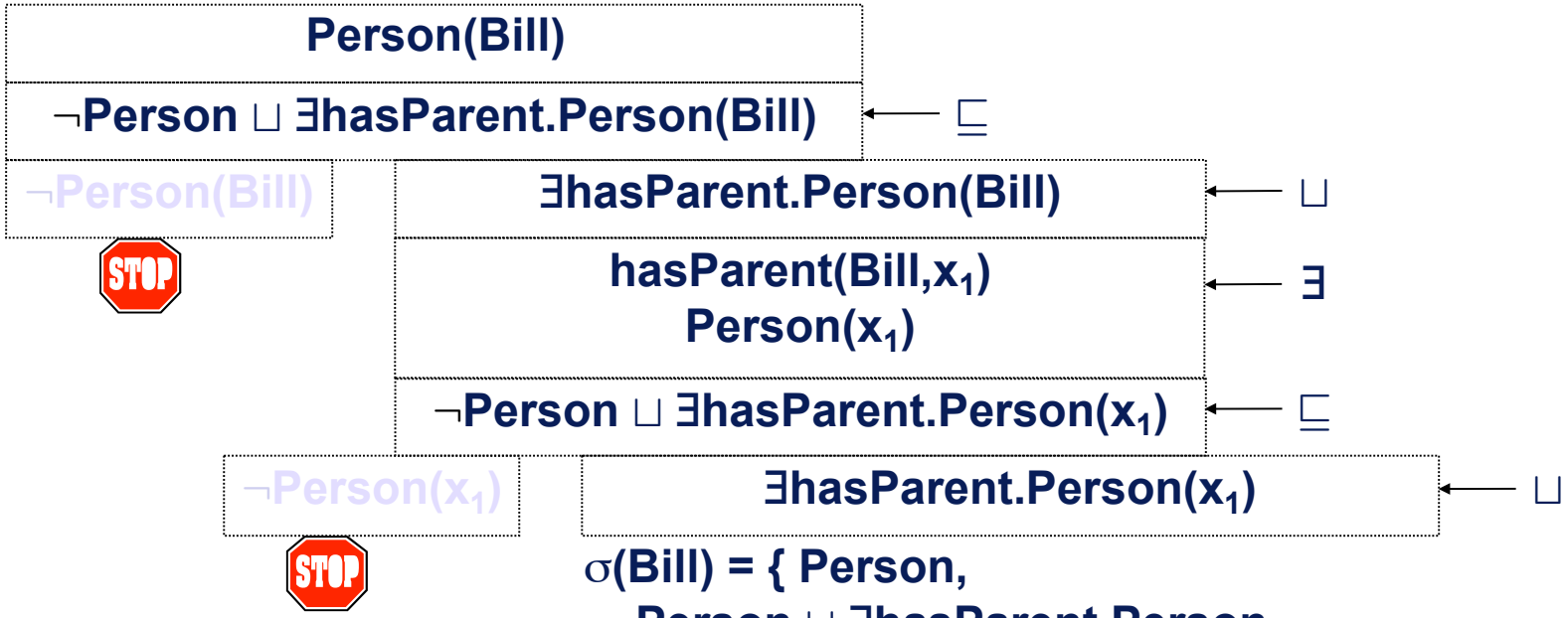
Folgendes wäre aber auch denkbar:

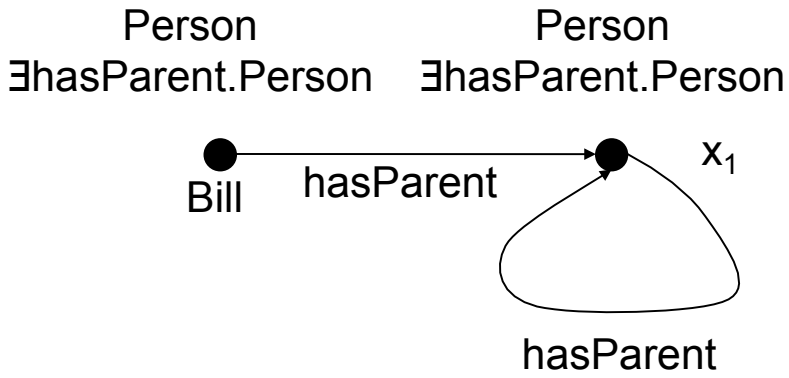


D.h. Wiederverwendung alter Knoten!
Es muss natürlich formal nachgewiesen werden, dass das ausreicht!

Tableau mit Blocking

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person
  Abzuleiten: ¬Person(Bill)





σ(Bill) = { Person,
Person ⊔ ∃hasParent.Person,
∃hasParent.Person }
σ(x1) = { Person,
Person ⊔ ∃hasParent.Person,
∃hasParent.Person }
σ(x1) ⊆ σ(Bill), so Bill blocks x1

Tableau – Blocking – Definition

Die Auswahl von (∃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.

Tableau für OWL DL

- Die Grundidee ist dieselbe!
- Kompliziertere Blockingregeln müssen verwendet werden.
- Schlechte Unterstützung von Instanzgenerierung.

Tableau-Beweiser

- Fact
  - http://www.cs.man.ac.uk/~horrocks/FaCT/
  - SHIQ
- Fact++
  - http://owl.man.ac.uk/factplusplus/
  - SROIQ(D)
- HermiT
  - http://hermit-reasoner.com
  - SROIQ(D)
- Pellet
  - http://www.mindswap.org/2003/pellet/index.shtml
  - SROIQ(D)
- RacerPro
  - http://www.sts.tu-harburg.de/~r.f.moeller/racer/
  - SHIQ(D)