Grundlagen Semantic Web

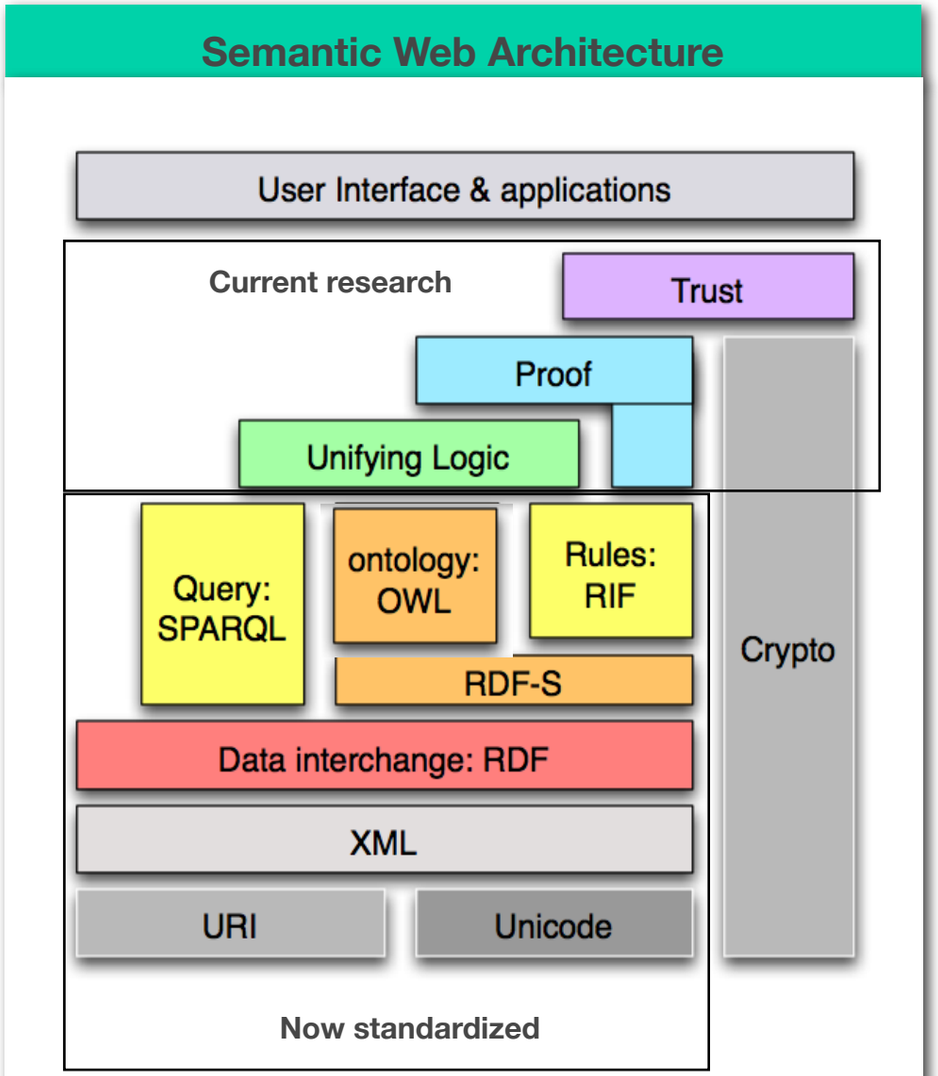
Lehrveranstaltung im WS09/10
Seminar für Computerlinguistik
Universität Heidelberg

Dr. Sebastian Rudolph
Institut AIFB
Universität Karlsruhe

AIFB

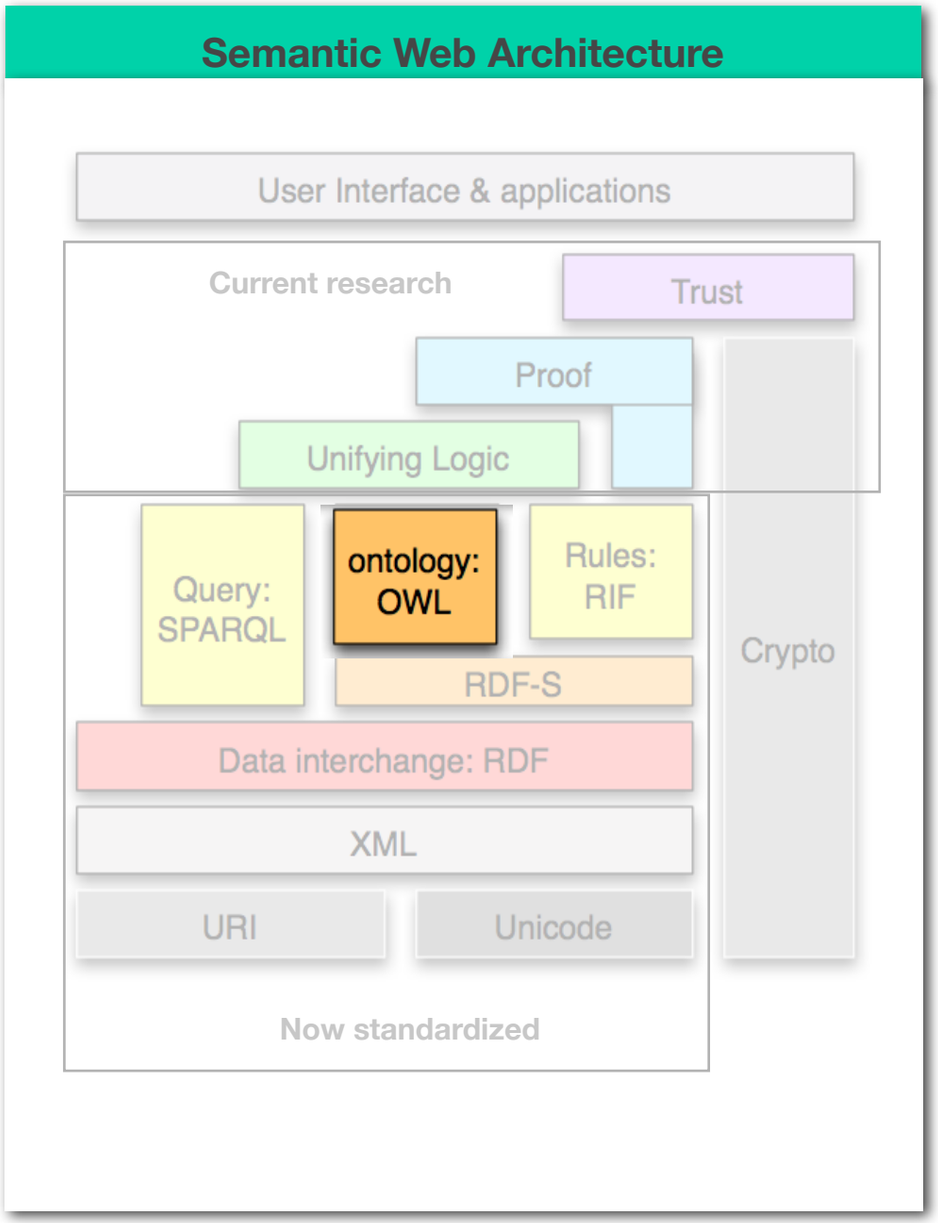
OWL - Syntax & Intuition

Dr. Sebastian Rudolph



OWL - Syntax & Intuition

Dr. Sebastian Rudolph



Agenda

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

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- 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

of a shared

conceptualization

of a domain of interest“

Ontologie – informatisch

AIFB

Gruber (1993):

„An Ontology is a

formal specification ⇒ maschinell interpretierbar

of a shared

conceptualization

of a domain of interest“

Ontologie – informatisch

AIFB

Gruber (1993):

„An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest“

⇒ maschinell interpretierbar

⇒ beruht auf Konsens

Ontologie – informatisch

AIFB

Gruber (1993):

„An Ontology is a

formal specification ⇒ maschinell interpretierbar

of a shared ⇒ beruht auf Konsens

conceptualization ⇒ beschreibt Begrifflichkeiten

of a domain of interest“

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)

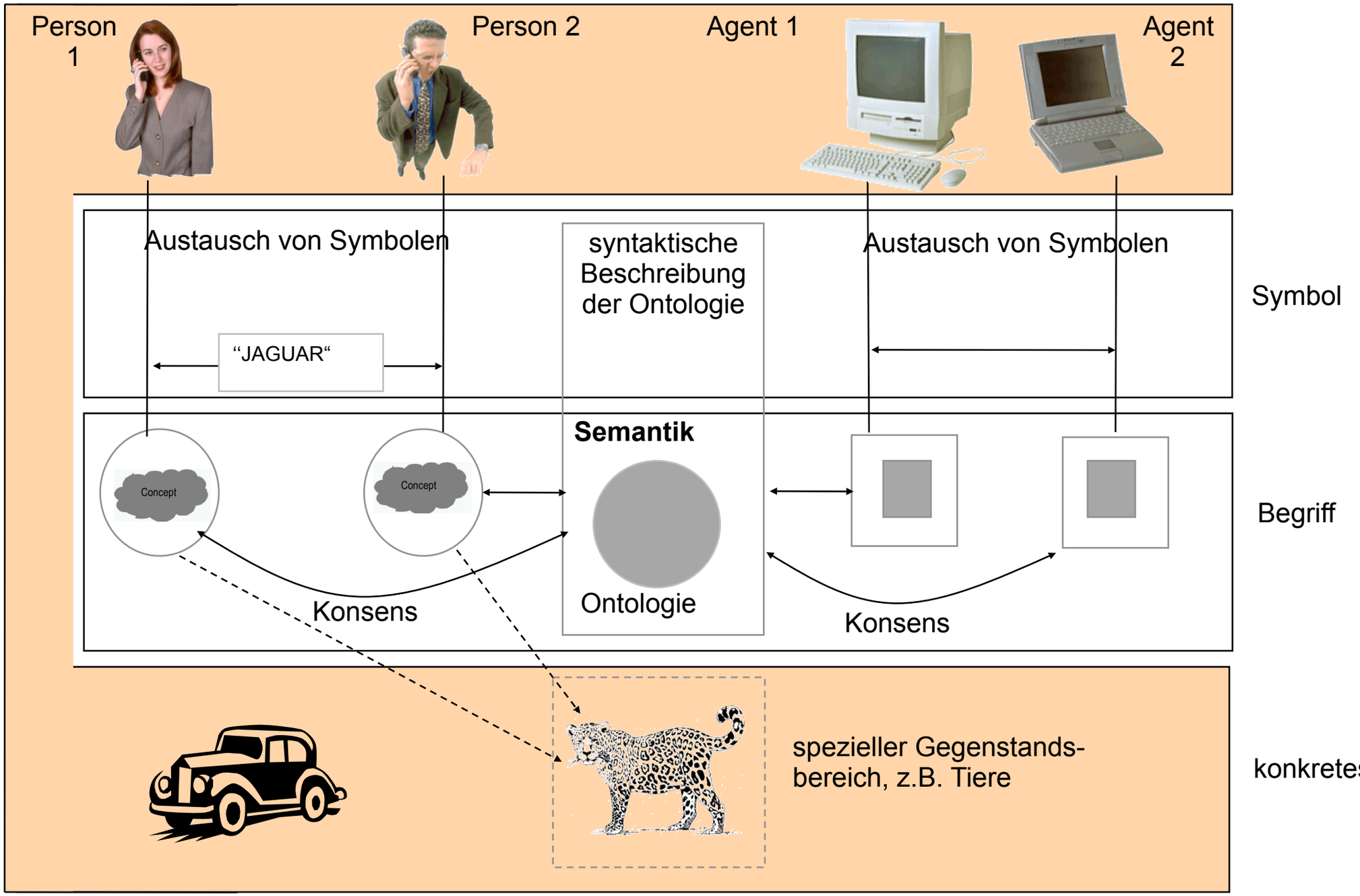
Ontologie – informatisch

Hitzler, Krötzsch, Rudolph (2009):

"In computer science, an ontology is a description of knowledge about a domain of interest, the core of which is a machine-processable specification with a formally defined meaning."

Ontologie & Kommunikation

AIFB



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
- OWL Varianten
- Anfragen an OWL-Ontologien

OWL – Allgemeines

- W3C Recommendation seit 2004
- Semantisches Fragment von FOL
- Drei Varianten:
  OWL Lite ⊆ OWL DL ⊆ OWL Full
- Keine Reifikation in OWL DL
  ➔ RDFS ist Fragment von OWL Full
- OWL DL ist entscheidbar
  entspricht der Beschreibungslogik $\mathcal{SHOIN}$(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

AIFB

- 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

AIFB

von RDFS geerbt

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

außerdem

- `owl:imports`

für Versionierung

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

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- 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 Propertys 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

AIFB

```
<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
- OWL Varianten
- Anfragen an OWL-Ontologien

Einfache Klassenbeziehungen

AIFB

```
<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

AIFB

```
<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

AIFB

```
<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

AIFB

```
<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="#SusanneWinter"/>
  </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
- OWL Varianten
- 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:intersectionOf
     rdf:parseType="Collection">
      <owl:Class rdf:about="#Sekretaerinnen"/>
      <owl:Class
     rdf:about="#AngehoerigeAGStuder"/>
    </owl:intersectionOf>
  </owl:equivalentClass>
</owl:Class>
```

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

Disjunktion


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

Negation


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

semantisch äquivalente Aussage:

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

Rolleneinschränkungen (allValuesFrom)

AIFB

- 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)

AIFB

```
<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)

AIFB

```
<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)

AIFB

```
<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)

AIFB

```
<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)

AIFB

```
<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)

AIFB

```
<owl:Class rdf:ID="PruefungBeiStuder">
  <rdfs: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>
  </rdfs:equivalentClass>
</owl:Class>
```

Agenda

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

Rollenbeziehungen

AIFB

```
<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

AIFB

```
<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!

AIFB

```
<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
- OWL Varianten
- Anfragen an OWL-Ontologien

OWL Varianten

- OWL Full
  - Enthält OWL DL und OWL Lite
  - Enthält als einzige OWL-Teilsprache ganz RDFS
  - Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind.
  - Unentscheidbar.
  - Wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt.
- OWL DL
  - Enthält OWL Lite und ist Teilsprache von OWL Full.
  - Entscheidbar.
  - Wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt.
  - Komplexität NExpTime (worst-case).
- OWL Lite
  - Ist Teilsprache von OWL DL und OWL Full.
  - Entscheidbar.
  - Wenig ausdrucksstark.
  - Komplexität ExpTime (worst-case).

OWL Full

- Uneingeschränkte Nutzung aller OWL und RDFS-Sprachelemente (muss gültiges RDFS sein).
- Schwierig z.B.: nicht vorhandene Typentrennung (Klassen, Rollen, Individuen), dadurch:
  - `owl:Thing` dasselbe wie `rdfs:resource`
  - `owl:Class` dasselbe wie `rdfs:Class`
  - `owl:DatatypeProperty` Subklasse von `owl:ObjectProperty`
  - `owl:ObjectProperty` dasselbe wie `rdf:Property`

Typendurchmischung in OWL Full


```
<owl:Class rdf:about="#Buch">
  <englischerName rdf:datatype="&xsd;string">
    book
  </englischerName>
  <franzoesischerName rdf:datatype="&xsd;string">
    livre
  </franzoesischerName>
</owl:Class>
```

Inferenzen über solche Konstrukte werden oft nicht wirklich benötigt.

OWL DL

- Nur Verwendung von explizit erlaubten RDFS Sprachelementen (z.B. die in unseren Beispielen). Nicht erlaubt: `rdfs:Class`, `rdfs:Property`
- Typentrennung. Klassen und Rollen müssen explizit deklariert werden.
- Konkrete Rollen dürfen nicht als transitiv, symmetrisch, invers oder invers funktional deklariert werden.
- Zahlenrestriktionen dürfen nicht mit transitiven Rollen, deren Subrollen, oder Inversen davon verwendet werden.

OWL Lite

- alle Einschränkungen für OWL DL
- außerdem:
  - nicht erlaubt: `oneOf`, `unionOf`, `complementOf`, `hasValue`, `disjointWith`
  - Zahlenrestriktionen nur mit 0 und 1 erlaubt.
  - Einige Einschränkungen zum Auftreten von anonymen (komplexen) Klassen, z.B. nur im Subjekt von `rdfs:subClassOf`.

Agenda

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- 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://www.mindswap.org/2003/pellet/index.shtml
  - KAON2, http://kaon2.semanticweb.org
  - FACT++, http://owl.man.ac.uk/factplusplus/
  - Racer, http://www.racer-systems.com/

OWL Sprachelemente

AIFB

Kopf

- rdfs:comment
- rdfs:label
- rdfs:seeAlso
- rdfs:isDefinedBy
- owl:versionInfo
- owl:priorVersion
- owl:backwardCompatibleWith
- owl:incompatibleWith
- owl:DeprecatedClass
- owl:DeprecatedProperty
- owl:imports

Beziehungen zwischen Individuen

- owl:sameAs
- owl:differentFrom
- owl:AllDifferent (zusammen mit owl:distinctMembers)

Vorgeschriebene Datentypen

- xsd:strong
- xsd:integer

OWL Sprachelemente

Klassenkonstruktoren und -beziehungen

- owl:Class
- owl:Thing
- owl:Nothing
- rdfs:subClassOf
- owl:disjointWith
- owl:equivalentClass
- owl:intersectionOf
- owl:unionOf
- owl:complementOf

Rollenrestriktionen

- owl:allValuesFrom
- owl:someValuesFrom
- owl:hasValue
- owl:cardinality
- owl:minCardinality
- owl:maxCardinality
- owl:oneOf

OWL Sprachelemente

Rollenkonstruktoren, -beziehungen und -eigenschaften

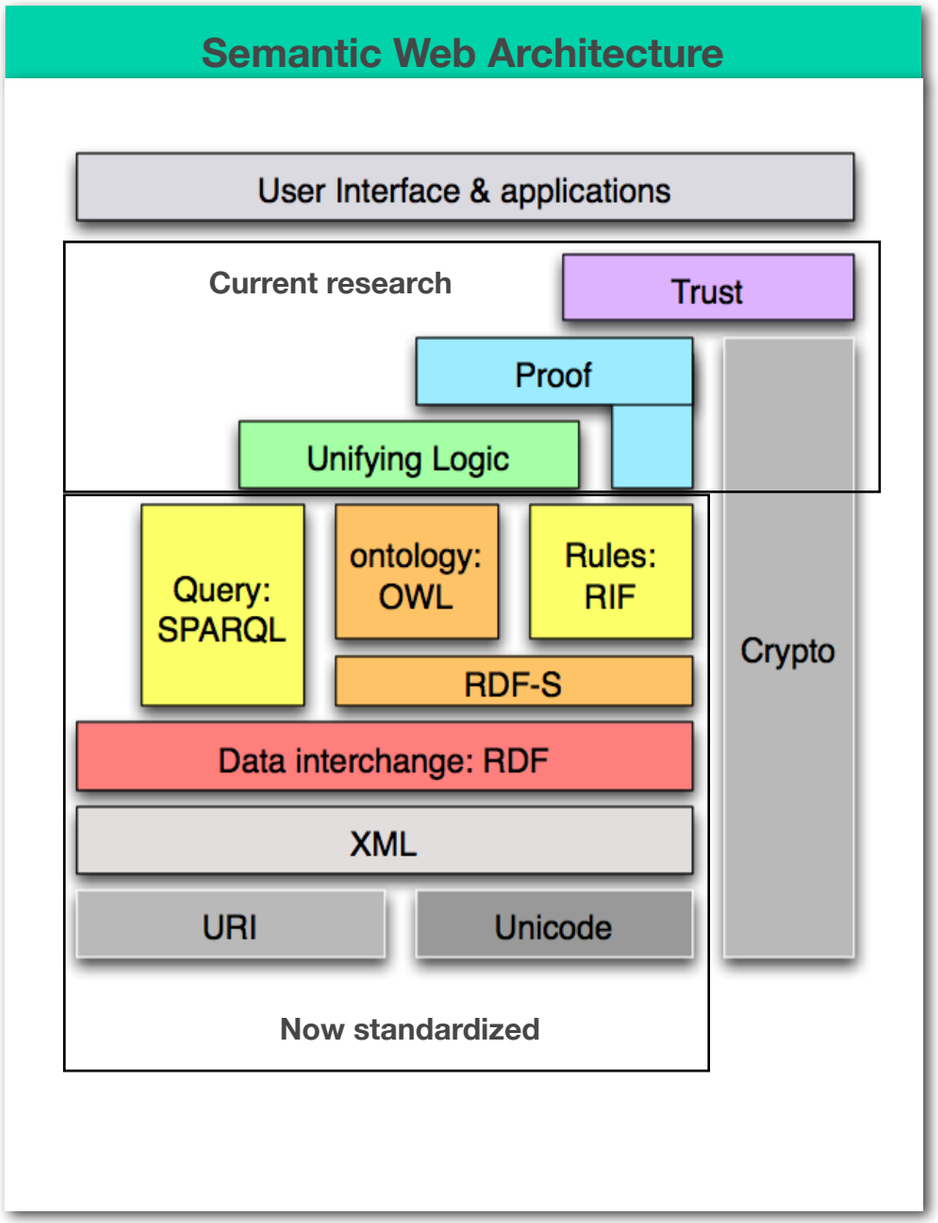
- owl:ObjectProperty
- owl:DatatypeProperty
- rdfs:subPropertyOf
- owl:equivalentProperty
- owl:inverseOf
- rdfs:domain
- rdfs:range
- rdf:resource="&owl;TransitiveProperty"
- rdf:resource="&owl;SymmetricProperty"
- rdf:resource="&owl;FunctionalProperty"
- rdf:resource="&owl;InverseFunctionalProperty"

Weiterführende Literatur

- `http://www.w3.org/2004/OWL/`
  zentrale W3C Webseite für OWL.
- `http://www.w3.org/TR/owl-features/`
  Überblick über OWL.
- `http://www.w3.org/TR/owl-ref/`
  vollständige Beschreibung der OWL-Sprachkomponenten.
- `http://www.w3.org/TR/owl-guide/`
  zeigt, wie OWL zur Wissensmodellierung verwendet werden kann.
- `http://www.w3.org/TR/owl-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.
  - Deutsche Übersetzungen mancher W3C Dokumente findet man unter `http://www.w3.org/2005/11/Translations/Lists/ListLang-de.html`

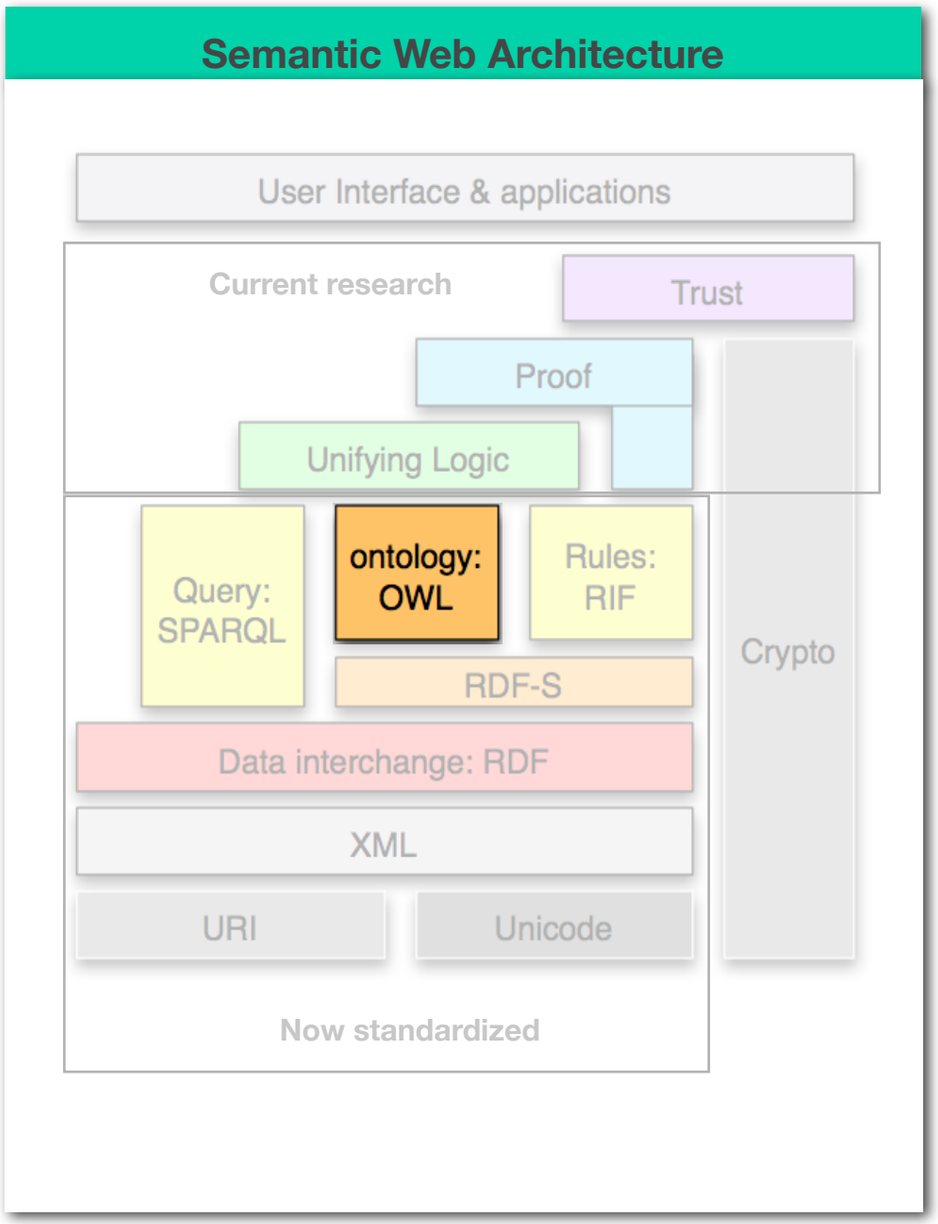
OWL - Semantik & Reasoning

Dr. Sebastian Rudolph



OWL - Semantik & Reasoning

Dr. Sebastian Rudolph



Agenda

- Beschreibungslogiken
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

Agenda

- Beschreibungslogiken
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

Beschreibungslogiken

AIFB

- 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 $\mathcal{SHOIN}$(D)
- wir besprechen zunächst $\mathcal{ALC}$ (Basis für komplexere DLs)

DLs – Aufbau

- DLs sind eine Familie logikbasierter Formalismen zur Wissensrepräsentation
- Spezielle Sprachen v.a. charakterisiert durch:
  - Konstruktoren für komplexe Klassen und Rollen aus einfacheren.
  - Menge von Axiomen um Fakten über Klassen, Rollen und Individuen auszudrücken.

- $\mathcal{ALC}$ ist die kleinste DL, die aussagenlogisch abgeschlossen ist
  - Konjunktion, Disjunktion, Negation sind Konstruktoren, geschrieben $\sqcap$, $\sqcup$, $\neg$.
  - Quantoren schränken Rollenbereiche ein, z.B.:

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

DLs – weitere Sprachmittel

- Andere Konstruktoren sind z.B.
  - number restrictions (Kardinalitätseinschränkungen) für Rollen:
    $\geq$3 hasChild, $\leq$1hasMother
  - qualified number restrictions (klassenspezifische Kardinalitäts-einschränkungen) für Rollen:
    $\geq$2 hasChild.Female, $\leq$1 hasParent.Male
  - Nominals (Definition durch Aufzählung):
    {Italy, France, Spain}
  - konkrete Bereiche (Datentypen):
    hasAge.($\geq$21)

  - inverse Rollen: hasChild^-
  - transitive Rollen: Trans(hasAncestor)
  - Rollenverknüpfung: Sibling $\circ$ Parent

Agenda

- Beschreibungslogiken
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

ALC - Grundbausteine

AIFB

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

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.

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

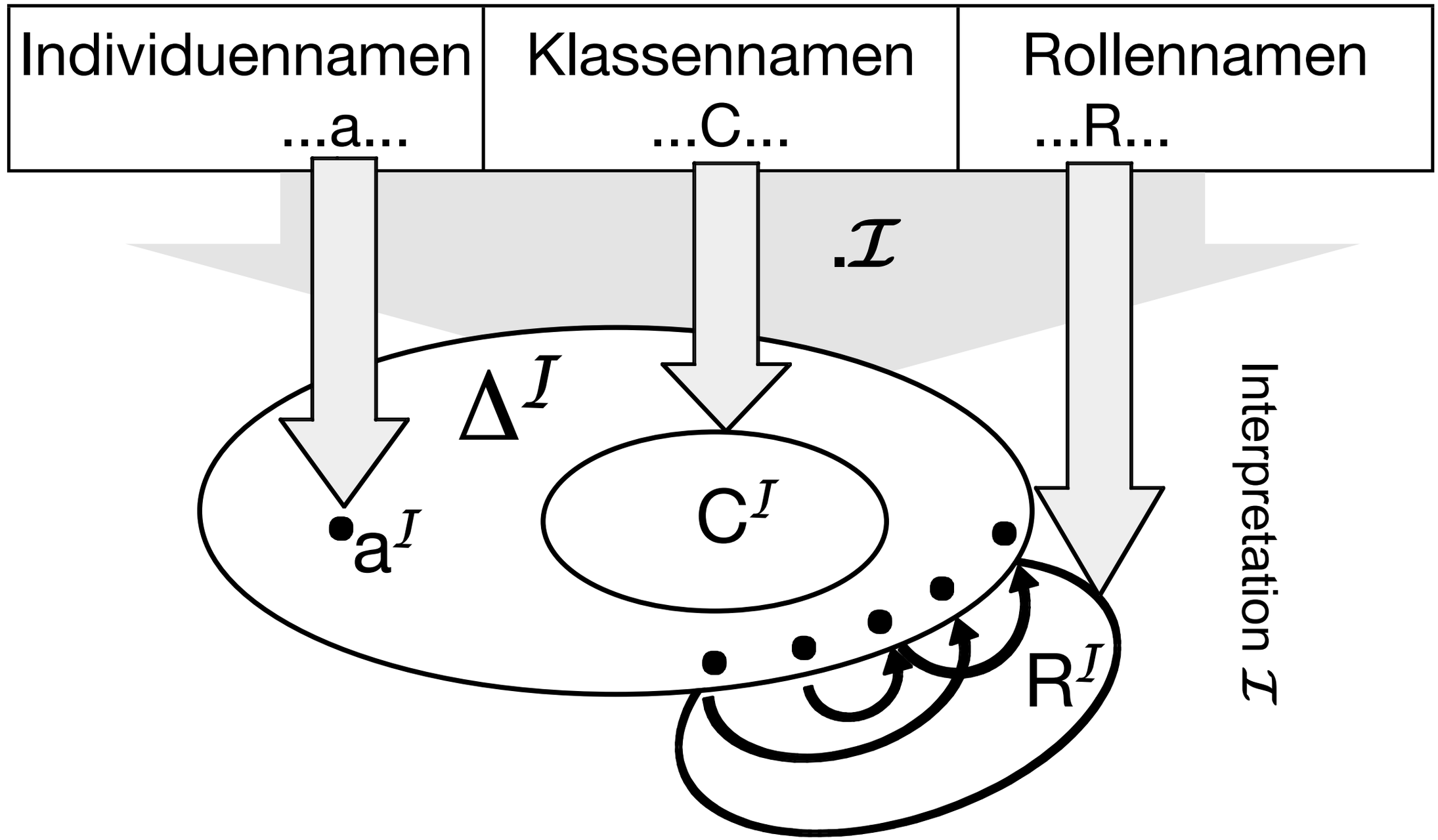
- Eine *$\mathcal{ALC}$-TBox* besteht aus Aussagen der Form $C \sqsubseteq D$ und $C \equiv D$, wobei C, D Klassen sind.
- Eine *$\mathcal{ALC}$-ABox* besteht aus Aussagen der Form C(a) und R(a,b), wobei C eine komplexe Klasse, R eine Rolle und a,b Individuen sind.
- Eine *$\mathcal{ALC}$-Wissensbasis*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

TBox vs. ABox

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 gegeben durch formale Semantik.

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
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

OWL als SHOIN(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 SHOIN(D) - NOMINALS

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 SHOIN(D) - Kardinalität

AIFB

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 SHOIN(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: $(\forall x)(\forall y)(R(x,y) \leftrightarrow S(y,x))$
- transitive Rollen: Trans(R)
  - gibt an, dass eine Rolle transitiv ist
  - 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 SHOIN(D) - Überblick

Erlaubt sind:

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

DL-Syntax - Übersicht

AIFB

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

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

Ontology (=Knowledge Base)

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

$\mathcal{S}$ $\mathcal{H}$	Role Axioms (RBox)	
$\mathcal{H}$	Subrole	R ⊑ S
$\mathcal{S}$	Transitivity	Trans(S)

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

$\mathcal{S}$ = $\mathcal{ALC}$ + Transitivity **OWL DL = $\mathcal{SHOIN}$(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

AIFB

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	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\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

AIFB

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.

Wissensmodellierung OWA vs. CWA

AIFB

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.

child(Bill,Bob)
Man(Bob)

Wissensmodellierung OWA vs. CWA

AIFB

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?

child(Bill,Bob)
Man(Bob)

? ⊨ ∀child.Man(Bill)

Wissensmodellierung OWA vs. CWA

AIFB

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.

child(Bill,Bob)
Man(Bob)

Are all children of Bill male?

? ⊨ ∀child.Man(Bill)

No idea, since we do not know all children of Bill.

DL answers don't know

Wissensmodellierung OWA vs. CWA

AIFB

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.

child(Bill,Bob)
Man(Bob)

? ⊨ ∀child.Man(Bill)

Are all children of Bill male?

No idea, since we do not know all children of Bill.

DL answers

don't know

If we assume that we know everything about Bill, then all of his children are male.

Prolog

yes

Wissensmodellierung OWA vs. CWA

AIFB

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)**		

Wissensmodellierung OWA vs. CWA

AIFB

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
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

Wichtige Inferenzprobleme

Wichtige Inferenzprobleme

- Globale Konsistenz der Wissensbasis — KB ⊨ **false**?
  - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz — $C \equiv \bot$?
  - Muss Klasse C leer sein?

Wichtige Inferenzprobleme

- Globale Konsistenz der Wissensbasis — KB $\models$ **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?

Wichtige Inferenzprobleme

- Globale Konsistenz der Wissensbasis — KB ⊨ **false**?
  - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz — C ≡ ⊥?
  - Muss Klasse C leer sein?
- Klasseninklusion (Subsumption) — C ⊑ D?
  - Strukturierung der Wissensbasis
- Klassenäquivalenz — C ≡ D?
  - Sind zwei Klassen eigentlich dieselbe?
- Klassendisjunktheit — C ⊓ D = ⊥?
  - Sind zwei Klassen disjunkt?
- Klassenzugehörigkeit — C(a)?
  - Ist Individuum a in der Klasse C?

Wichtige Inferenzprobleme

- Globale Konsistenz der Wissensbasis — KB ⊨ **false**?
  - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz — C ≡ ⊥?
  - Muss Klasse C leer sein?
- Klasseninklusion (Subsumption) — C ⊑ D?
  - Strukturierung der Wissensbasis
- Klassenäquivalenz — C ≡ D?
  - Sind zwei Klassen eigentlich dieselbe?
- Klassendisjunktheit — C ⊓ D = ⊥?
  - 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 von OWL DL

- Entscheidbarkeit: zu jedem Inferenzproblem gibt es einen immer terminierenden Algorithmus.

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.

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!

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!

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.
- Tableauverfahren zeigt 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
- ALC
- OWL als SHOIN(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

Teilen des Tableaus in zwei *Zweige*.

Tableau – weiteres Beispiel

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

Ableitung von Konsequenzen:

$C(a)$

Teilen des Tableaus in zwei *Zweige*.

Tableau – weiteres Beispiel

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

Ableitung von Konsequenzen:

$C(a)$

$\neg D(a)$

Teilen des Tableaus in zwei *Zweige*.

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

Teilen des Tableaus in zwei *Zweige*.

Tableau – weiteres Beispiel

C(a) ¬C ⊔ D ¬D(a)

Ableitung von Konsequenzen:

C(a)

¬D(a)

(¬C ⊔ D)(a)

Nun Fallunterscheidung

1. ¬C(a)

   Widerspruch

Teilen des Tableaus in zwei *Zweige*.

Tableau – weiteres Beispiel

C(a) ¬C ⊔ D ¬D(a)

Ableitung von Konsequenzen:

C(a)

¬D(a)

(¬C ⊔ D)(a)

Nun Fallunterscheidung

1. ¬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.

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)

$P(a)$

$\neg E(a)$

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

Tableau – Beispiel (2/2)

TBox: ¬P ⊔ (E ⊓ U) ⊔ (E ⊓ ¬D)

Tableau:

(P ⊓ ¬E)(a) (aus Wissensbasis)

P(a)

¬E(a)

(¬P ⊔ (E ⊓ U) ⊔ (E ⊓ ¬D))(a)

¬P(a) ((E ⊓ U) ⊔ (E ⊓ ¬D))(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)

$P(a)$

$\neg E(a)$

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

$\neg P(a)$

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

$(E \sqcap U)(a)$

$E(a)$

$U(a)$

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

$E(a)$

$\neg D(a)$

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)

Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

Person(Bill)
¬Person ⊔ ∃hasParent.Person(Bill)

Tableau – Terminierungsproblem

AIFB

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

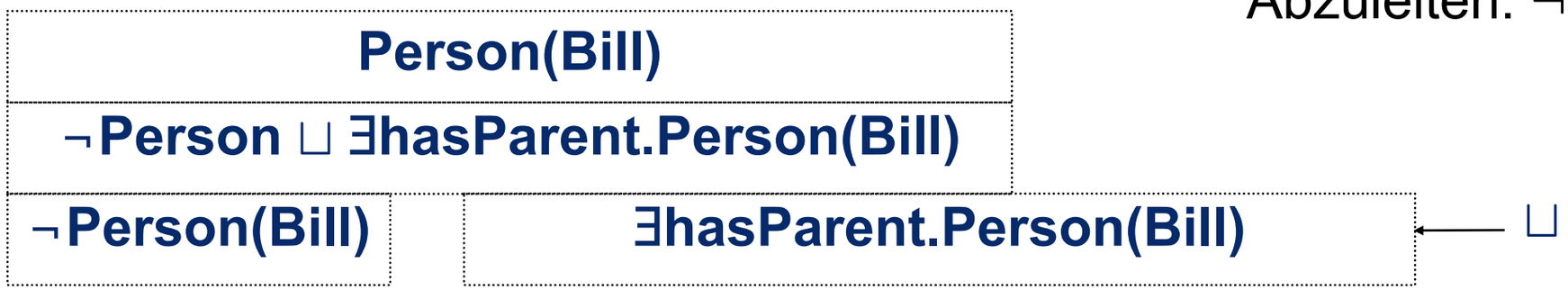


Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

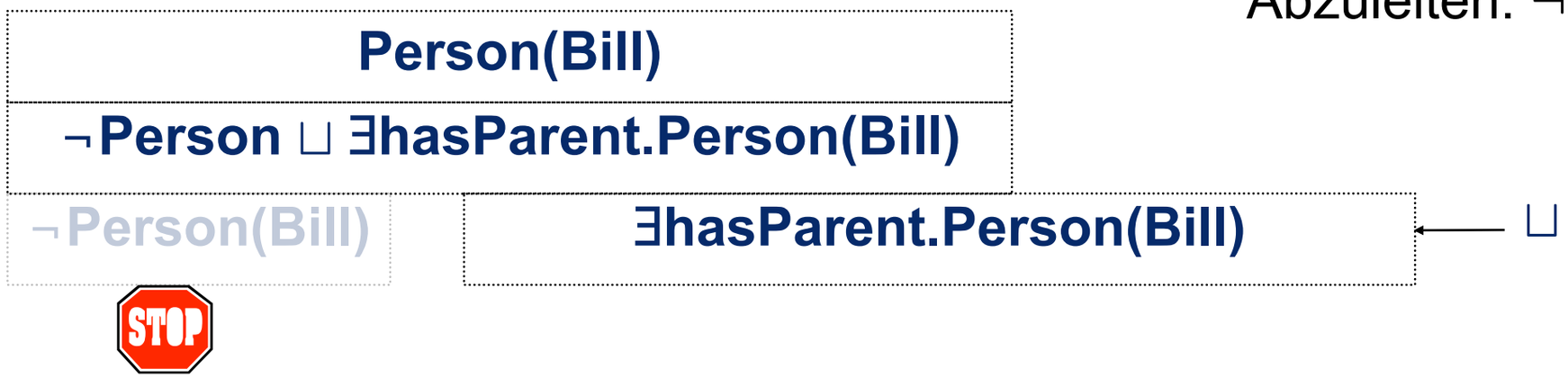


Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

Person(Bill)

¬Person ⊔ ∃hasParent.Person(Bill)

¬Person(Bill)

∃hasParent.Person(Bill) ← ⊔

hasParent(Bill,x_1)
Person(x_1) ← ∃

¬Person ⊔ ∃hasParent.Person(x_1)

Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

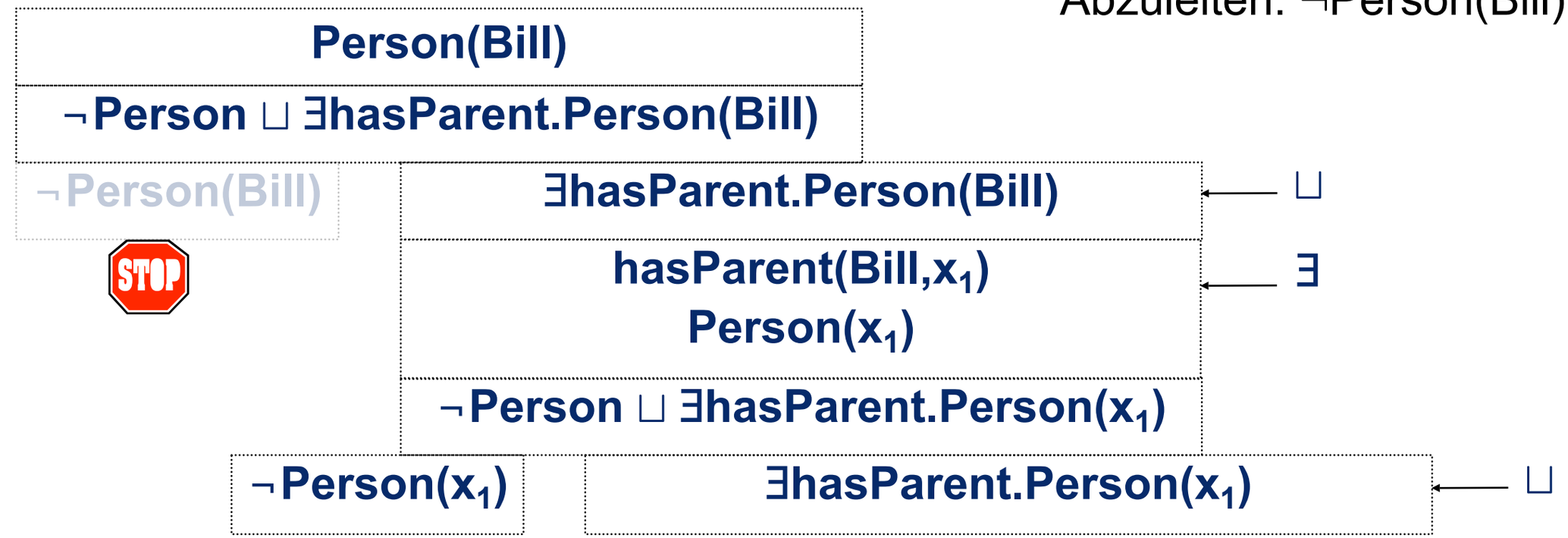


Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

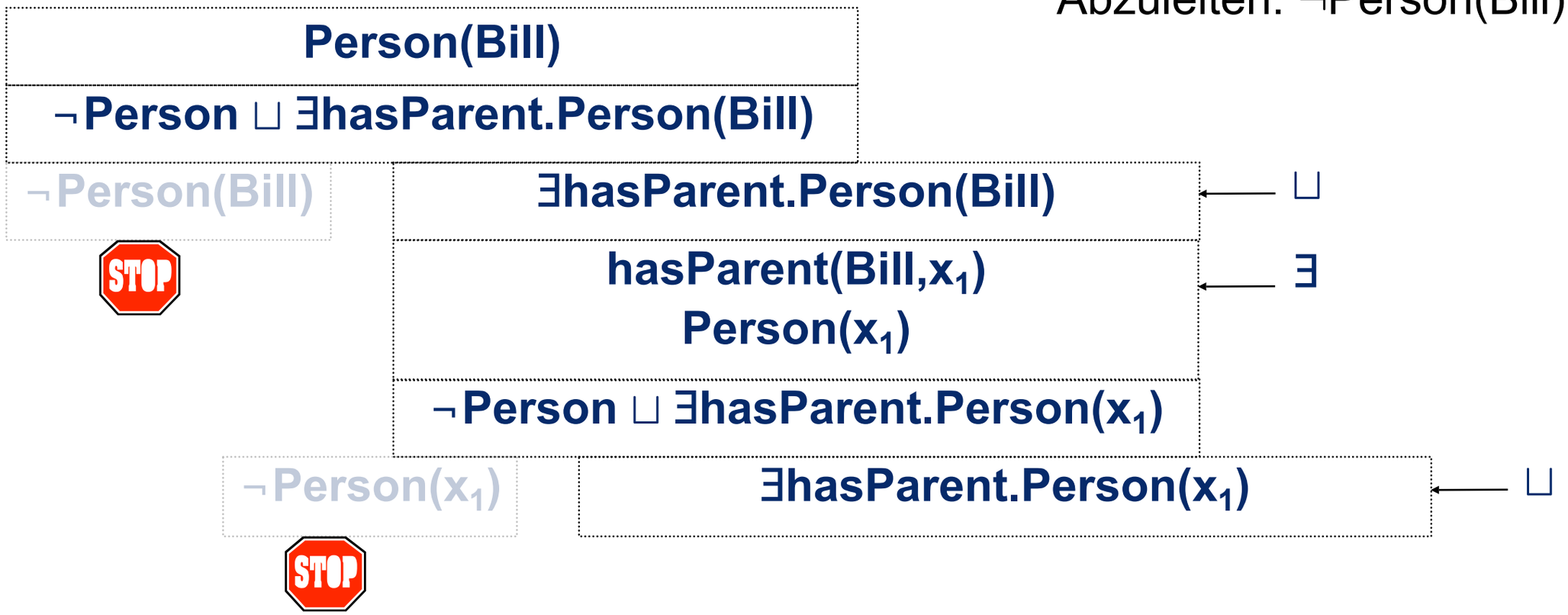


Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

Person(Bill)

¬Person ⊔ ∃hasParent.Person(Bill)

¬Person(Bill)

∃hasParent.Person(Bill) ← ⊔

hasParent(Bill,x_1)
Person(x_1) ← ∃

¬Person ⊔ ∃hasParent.Person(x_1)

¬Person(x_1)

∃hasParent.Person(x_1) ← ⊔

hasParent(x_1,x_2)
Person(x_2) ← ∃

¬Person ⊔ ∃hasParent.Person(x_2)

Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

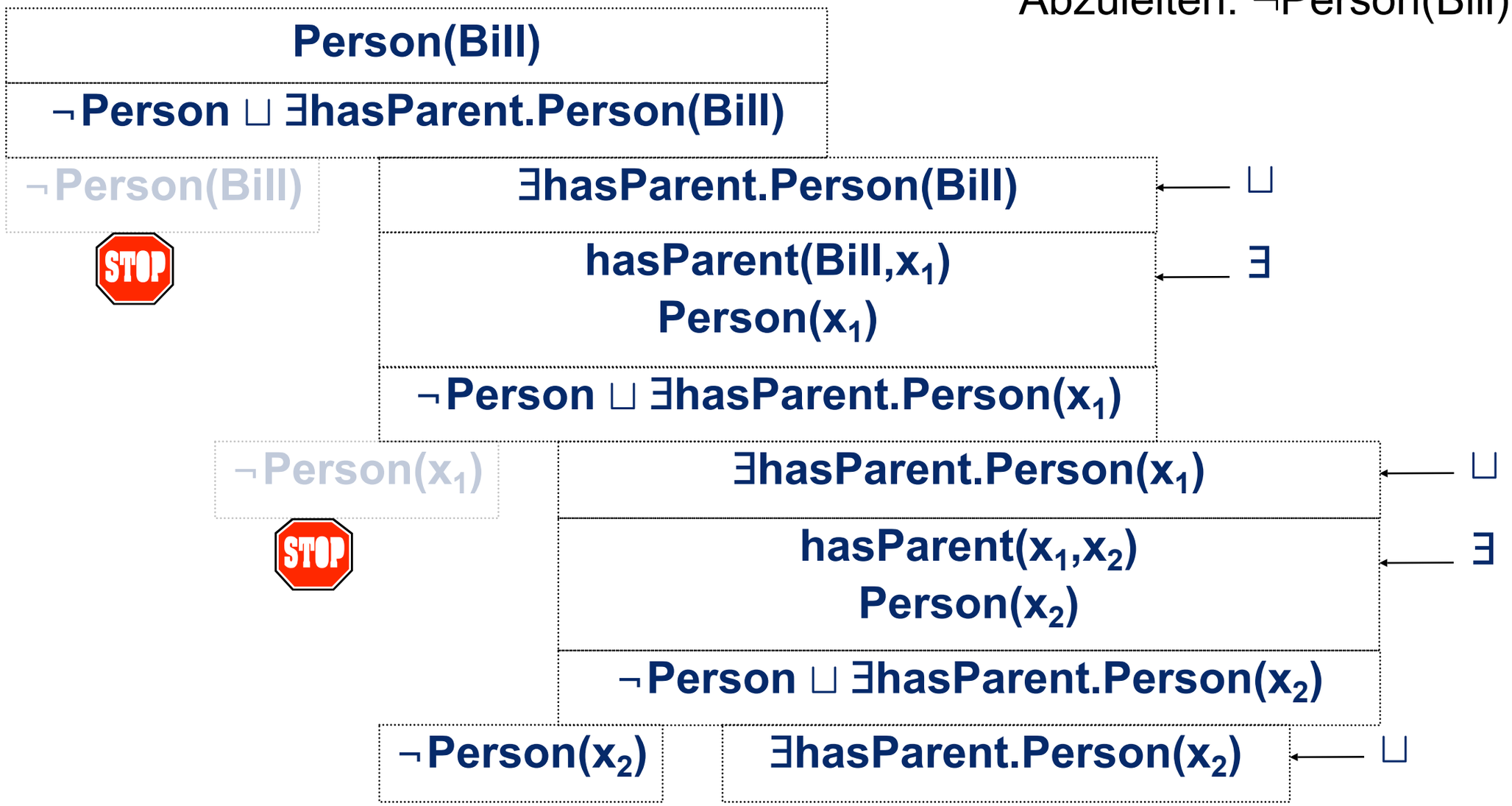


Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

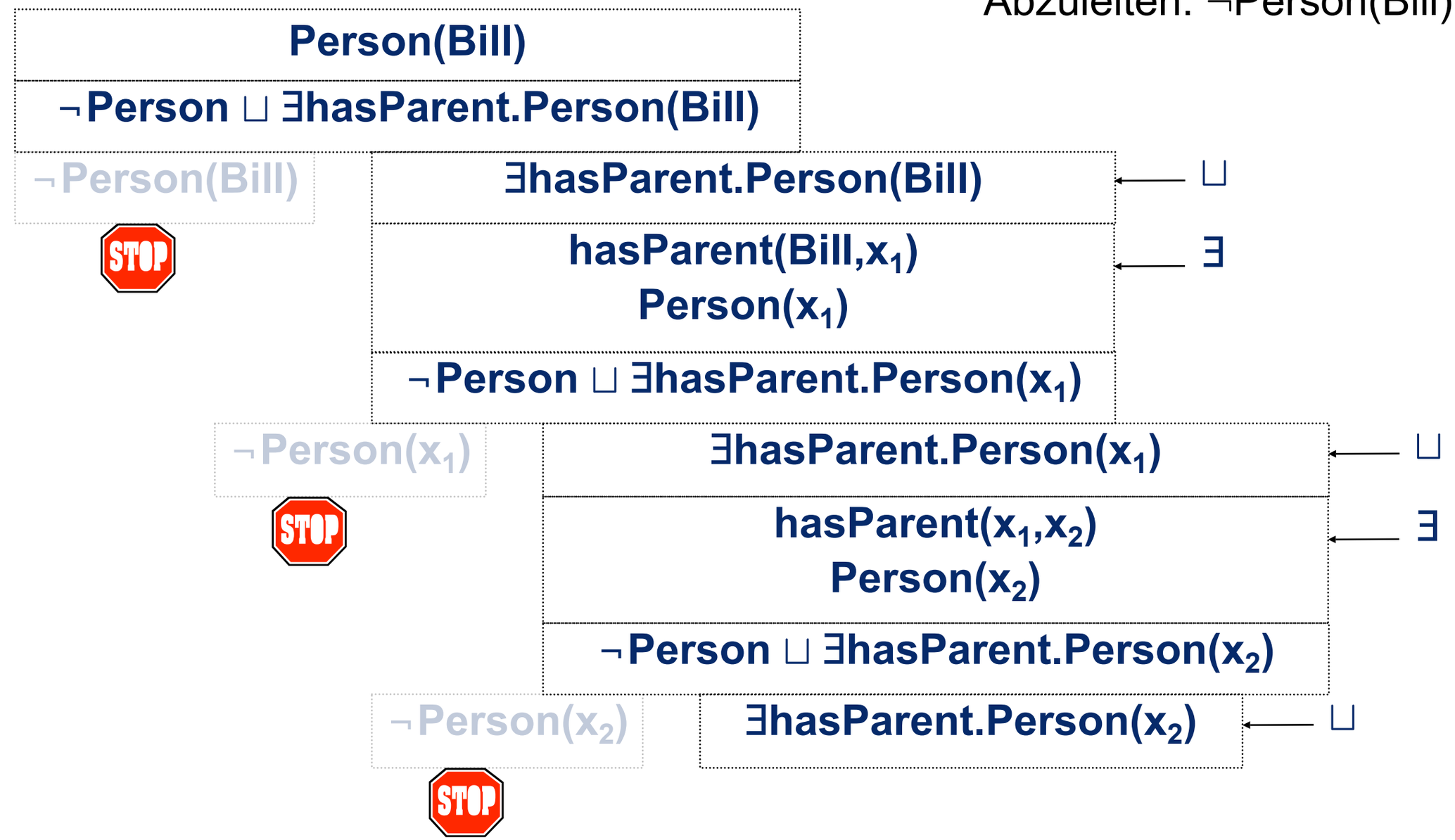


Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

Person(Bill)

¬Person ⊔ ∃hasParent.Person(Bill)

¬Person(Bill) — STOP

∃hasParent.Person(Bill) ← ⊔

hasParent(Bill,x_1)
Person(x_1) ← ∃

¬Person ⊔ ∃hasParent.Person(x_1)

¬Person(x_1) — STOP

∃hasParent.Person(x_1) ← ⊔

hasParent(x_1,x_2)
Person(x_2) ← ∃

¬Person ⊔ ∃hasParent.Person(x_2)

¬Person(x_2) — STOP

∃hasParent.Person(x_2) ← ⊔

etc.

Tableau – Terminierungsproblem

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

Abzuleiten: ¬Person(Bill)

Person(Bill)

¬Person ⊔ ∃hasParent.Person(Bill)

¬Person(Bill)

STOP

∃hasParent.Person(Bill) ← ⊔

hasParent(Bill,x_1)
Person(x_1) ← ∃

¬Person ⊔ ∃hasParent.Person(x_1)

¬Person(x_1)

STOP

∃hasParent.Person(x_1) ← ⊔

hasParent(x_1,x_2)
Person(x_2) ← ∃

¬Person ⊔ ∃hasParent.Person(x_2)

¬Person(x_2)

STOP

∃hasParent.Person(x_2) ← ⊔

etc.

Problem tritt auf durch Existenzquantoren (und minCardinality)

Tableau – Blocking – Idee

- Wir haben folgendes konstruiert:

Tableau – Blocking – Idee

AIFB

- Wir haben folgendes konstruiert:

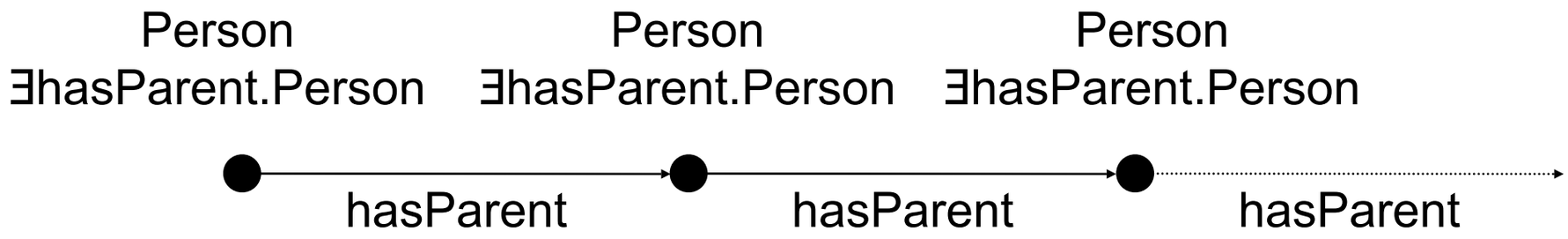
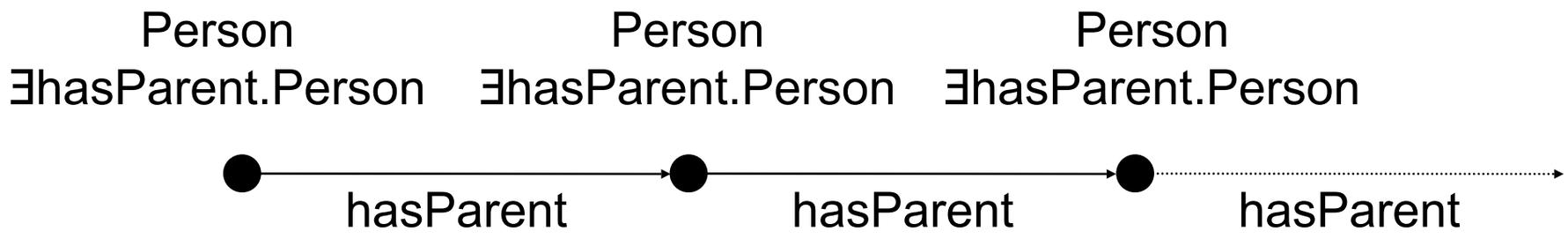


Tableau – Blocking – Idee

- Wir haben folgendes konstruiert:

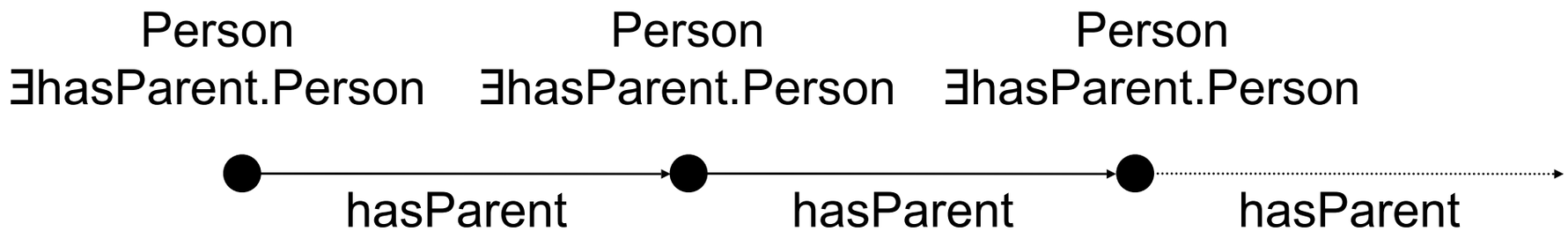


- Folgendes wäre aber auch denkbar:

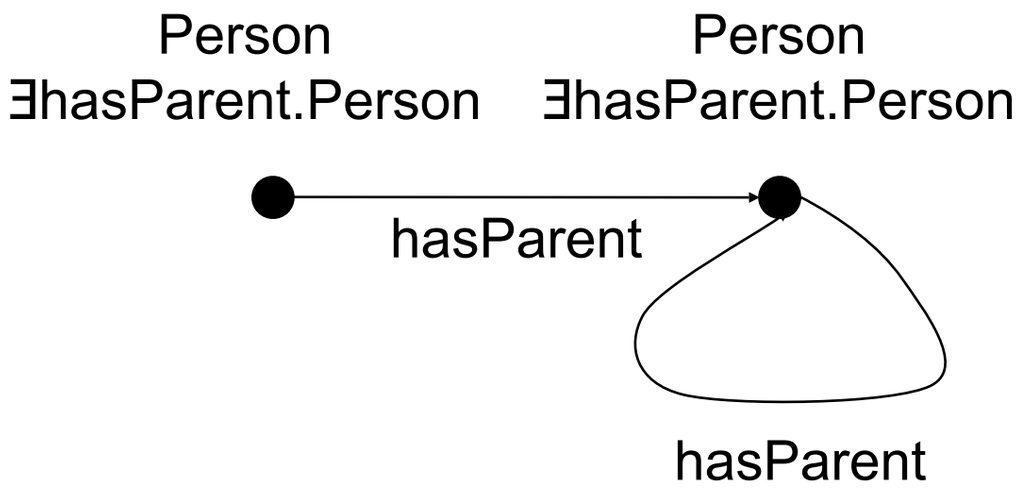
Tableau – Blocking – Idee

AIFB

- Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:

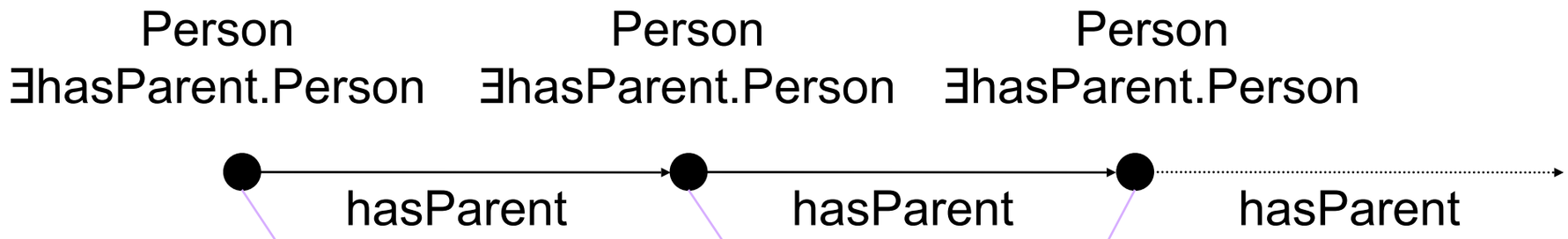


Blocking!

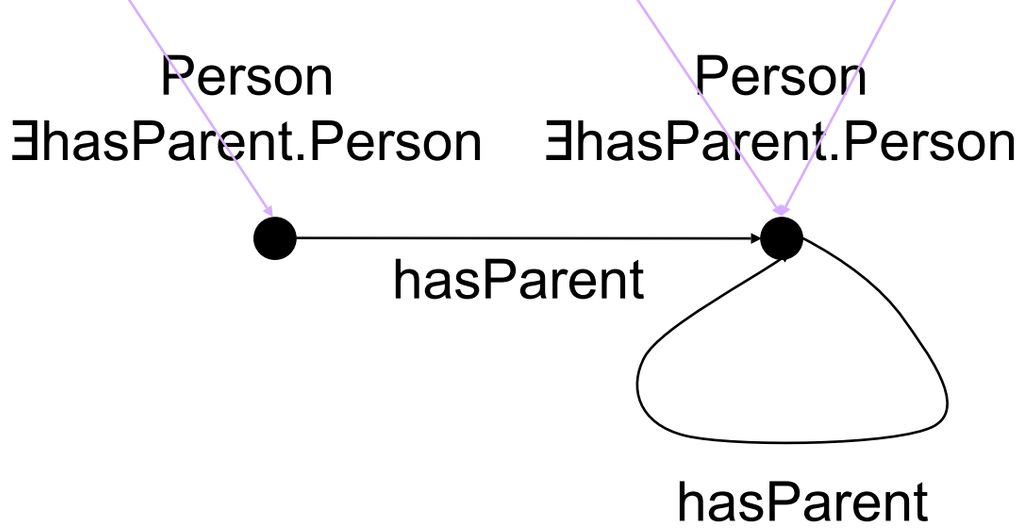
Tableau – Blocking – Idee

AIFB

- Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:

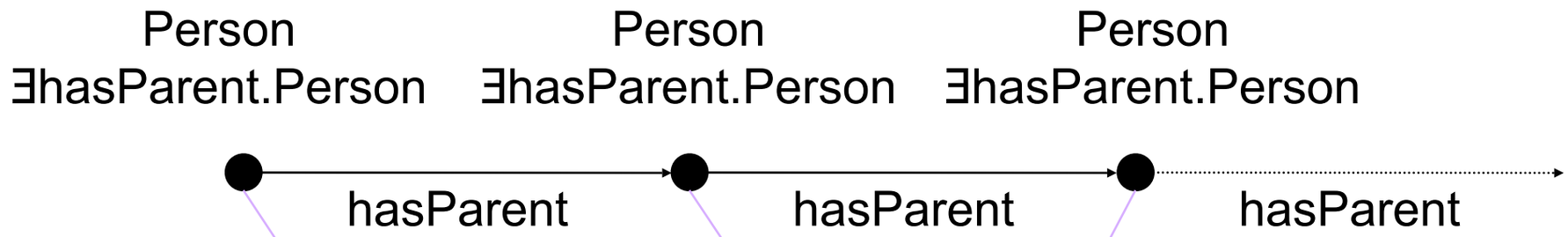


Blocking!

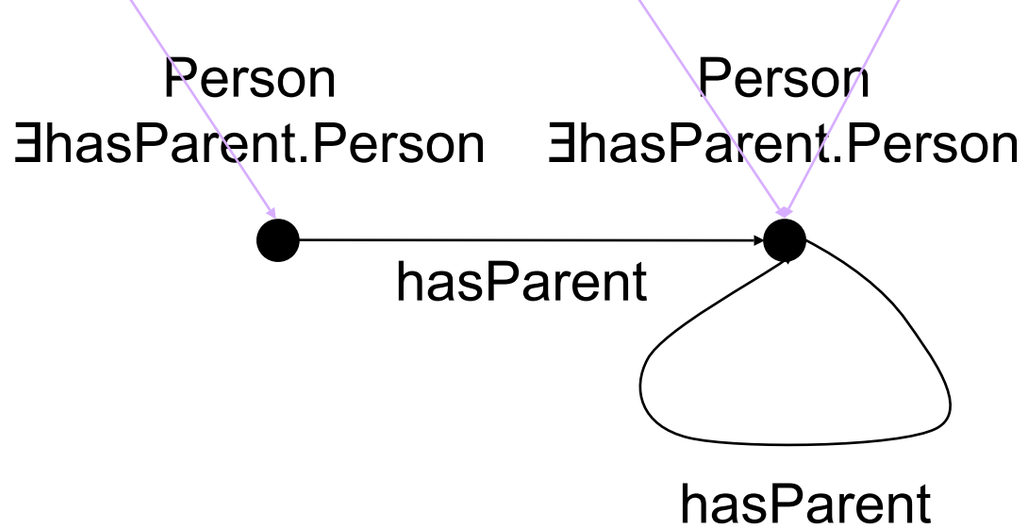
Tableau – Blocking – Idee

AIFB

- Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:



Blocking!

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)

Person(Bill)

Tableau mit Blocking

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

  Abzuleiten: ¬Person(Bill)

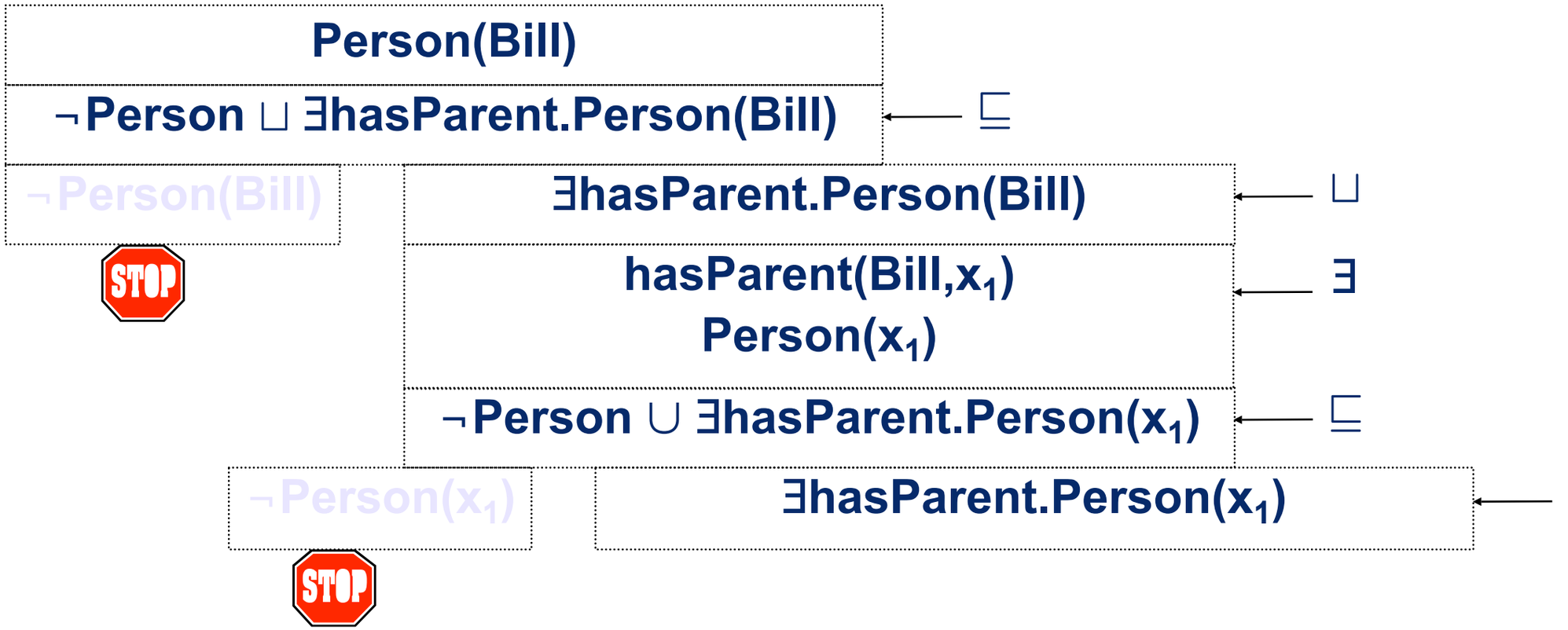


Tableau mit Blocking

- Einziges Axiom: ¬Person ⊔ ∃hasParent.Person

  Abzuleiten: ¬Person(Bill)

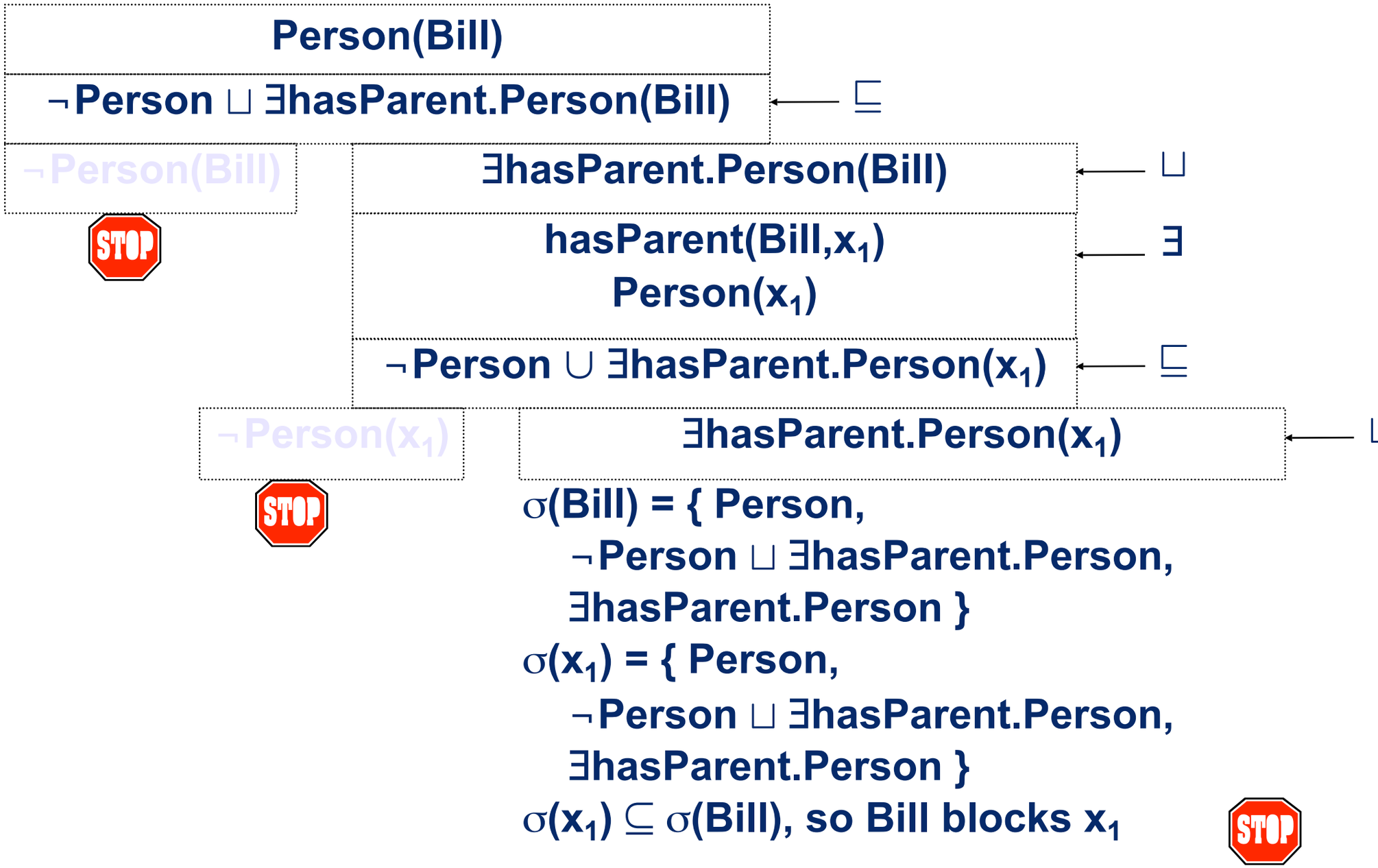
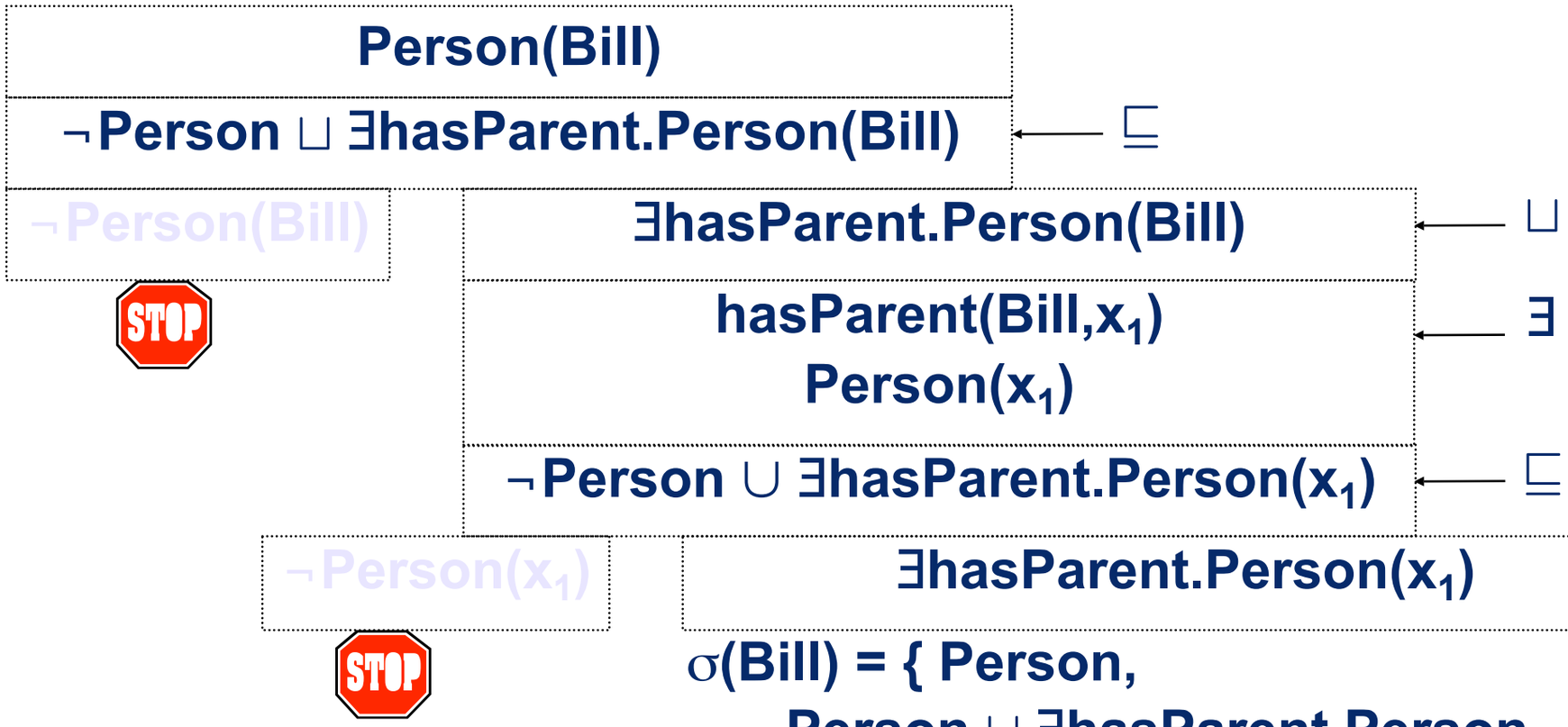


Tableau mit Blocking

- 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

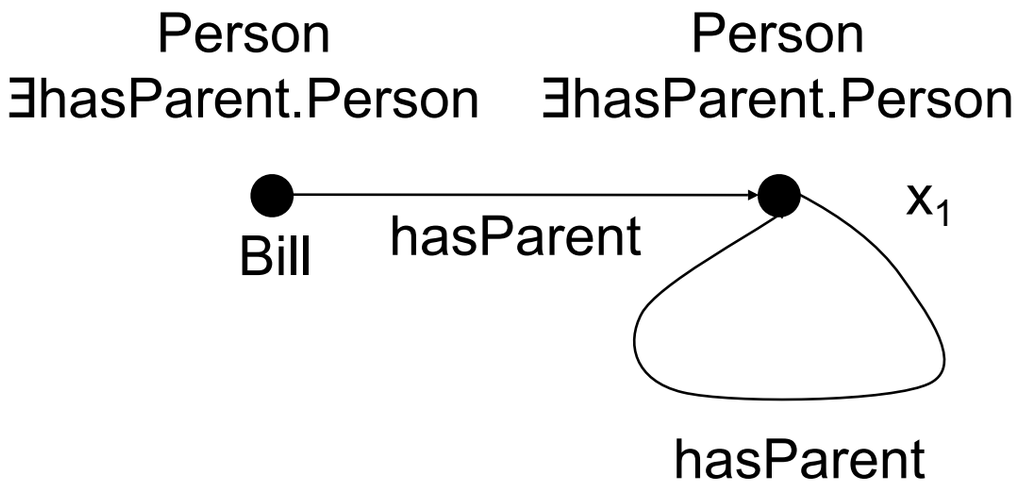


Tableau – Blocking – Definition

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 mit Blocking ist 2NExptime!
  $\rightarrow$ schlechter als nötig!

Tableau-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)

OWL 2 – Syntax und Semantik

Sebastian Rudolph

Institut AIFB · Universität Karlsruhe

Grundlagen Semantic Web (WS09/10)
Seminar für Computerlinguistik, Universität Heidelberg
http://semantic-web-grundlagen.de

Grundlagen Semantic Web

1. Einleitung und Ausblick
2. XML und URIs
3. Einführung in RDF
4. RDF Schema
5. Logik – Grundlagen
6. Semantik von RDF(S)
7. OWL – Syntax und Intuition
8. OWL – Semantik und Reasoning
9. OWL 2 – Syntax und Semantik
10. SPARQL – Syntax und Intuition
11. Semantik von SPARQL
12. Konjunktive Anfragen und Regelsprachen

Entwicklung von OWL 2

OWL 2 als „nächste Version“ von OWL

Erweiterungen aufgrund von Praxiserfahrung mit OWL 1.0:

- zusätzliche Ausdrucksstärke durch neue ontologische Axiome
- nicht-logische Erweiterunen (Syntax, Metadaten, . . .)
- Überarbeitung der OWL-Varianten (Lite/DL/Full)

Zielstellungen:

- weitestgehende Kompatibilität zum existierenden OWL-Standard
- Erhaltung der Entscheidbarkeit von OWL DL
- Behebung von Problemen im OWL-1.0-Standard

Von $\mathcal{SHOIN}$ zu $\mathcal{SROIQ}$

OWL DL basiert auf Beschreibungslogik $\mathcal{SHOIN}(D)$:

- Axiome:
  - Tbox: Subklassenbeziehungen $C \sqsubseteq D$
  - Rbox: Subrollenbeziehungen $R \sqsubseteq S$ ($\mathcal{H}$), Inverse Rollen R^- ($\mathcal{I}$), Transitivität
  - Abox: Fakten zu Klassen $C(a)$, Rollen $R(a, b)$, und Gleichheit $a \approx b$ bzw. $a \not\approx b$
- Klassenkonstruktoren:
  - Konjunktion $C \sqcap D$, Disjunktion $C \sqcup D$, Negation $\neg C$ von Klassen
  - Rollenrestriktionen: universell $\forall R.C$ und existenziell $\exists R.C$
  - Zahlenrestriktionen ($\mathcal{N}$): $\leq n\, R$ und $\geq n\, R$ (n nicht-negative Zahl)
  - Nominale ($\mathcal{O}$): $\{a\}$
- Datentypen (D)

Erweiterung in OWL 2 zu $\mathcal{SROIQ}(D)$

Abox

$\mathcal{SHOIN}$ unterstützt verschiedene Abox-Fakten:

- Klassenzugehörigkeit $C(a)$ (C komplexe Klasse),
- Sonderfall: negierte Klassenzugehörigkeit $\neg C(a)$ (C komplexe Klasse),
- Gleichheit $a \approx b$,
- Ungleichheit $a \not\approx b$
- Rollenbeziehungen $R(a, b)$
- *negierte Rollenbeziehungen?*

$\rightsquigarrow$ $\mathcal{SROIQ}$ erlaubt auch negierte Rollen in der Abox: $\neg R(a, b)$

Zahlenrestriktionen

$\mathcal{SHOIN}$ unterstützt nur einfache Zahlenrestriktionen ($\mathcal{N}$):

$$\text{Person} \sqcap \geq 3\, \text{hatKind}$$

„Klasse aller Personen mit 3 oder mehr Kindern.“

$\rightsquigarrow \mathcal{SROIQ}$ erlaubt auch qualifizierte Zahlenrestriktionen ($\mathcal{Q}$):

$$\text{Person} \sqcap \geq 3\, \text{hatKind}.(\text{Frau} \sqcap \text{Professor})$$

„Klasse aller Personen mit 3 oder mehr Töchtern, die Professoren sind.“

Das Konzept Self

Modellierungsaufgabe: „Jeder Mensch kennt sich selbst.“

- $\mathcal{SHOIN}$:

  $$\text{kennt}(\text{tom}, \text{tom}) \quad \text{kennt}(\text{tina}, \text{tina}) \quad \text{kennt}(\text{udo}, \text{udo}) \quad \ldots$$

  $\rightsquigarrow$ nicht allgemein anwendbar

- $\mathcal{SROIQ}$: spezieller Ausdruck Self

  $$\text{Mensch} \sqsubseteq \exists \text{kennt}.\text{Self}$$

Rollenaxiome und die universelle Rolle

$\mathcal{SROIQ}$ bietet zusätzliche Aussagen über Rollen:

- Tra(R): R ist transitiv (definiert wie in $\mathcal{SHOIN}$)
  Beispiel: Tra(liegtIn)
- Sym(R): R ist symmetrisch (definiert wie in $\mathcal{SHOIN}$)
  Beispiel: Sym(verwandtMit)
- Ref(R): R ist reflexiv, $(x, x) \in R^{\mathcal{I}}$ für alle Domänenindividuen x
  Beispiel: Ref(kennt)
- Irr(R): R ist irreflexiv, $(x, x) \notin R^{\mathcal{I}}$ für alle Domänenindividuen x
  Beispiel: Irr(hatKind)
- Dis(R, S): R und S sind disjunkt, $(x, y) \notin R^{\mathcal{I}} \cap S^{\mathcal{I}}$ für alle x, y
  Beispiel: Dis(hatVater, hatSohn)
- Universelle Rolle U: $(x, y) \in U^{\mathcal{I}}$ für alle x, y
  Beispiel: $\top \sqsubseteq \leq 7000000000\, U.\text{Menschen}$ (nicht empfohlen!)
  $\rightsquigarrow$ U ist vor allem als Gegenstück zu $\top$ sinnvoll, z.B. als Wurzel der Rollenhierarchie in grafischen Editoren

Allgemeine Rolleninklusion

„Die Freunde meiner Freunde sind auch meine Freunde.“

$\rightsquigarrow$ Kann in $\mathcal{SHOIN}$ ausgedrückt werden: hatFreund ist transitiv.

„Die Feinde meiner Freunde sind auch meine Feinde.“

$\rightsquigarrow$ Kann nicht in $\mathcal{SHOIN}$ ausgedrückt werden!

Rolleninklusion

- Rbox-Ausdrücke der Form $R_1 \circ R_2 \circ \ldots \circ R_n \sqsubseteq S$,
  Beispiel: hatFreund $\circ$ hatFeind $\sqsubseteq$ hatFeind
- Semantik: wenn $(x_0, x_1) \in R_1^{\mathcal{I}}, (x_1, x_2) \in R_2^{\mathcal{I}}, \ldots, (x_{n-1}, x_n) \in R_n^{\mathcal{I}}$,
  dann gilt auch $(x_0, x_n) \in S^{\mathcal{I}}$
  Beispiel: wenn $(x, y) \in \text{hatFreund}^{\mathcal{I}}$ und $(y, z) \in \text{hatFeind}^{\mathcal{I}}$,
  dann gilt auch $(x, z) \in \text{hatFeind}^{\mathcal{I}}$

Weitere Beispiele:

teilVon $\circ$ gehört $\sqsubseteq$ gehört hatBruder $\circ$ hatKind $\sqsubseteq$ istOnkelVon

Ausdrucksstärke der Rolleninklusion

Wie kompliziert ist Rolleninklusion?

Mit Rboxen kann man formale Sprachen kodieren: (Skizze!)

Grammatik für Sprache der Wörter `ab`, `aabb`, `aaabbb`, ...:

`L ::= ab`
`L ::= aLb`

wird zu Rbox

$$R_a \circ R_b \sqsubseteq L$$
$$R_a \circ L \circ R_b \sqsubseteq L$$

$\rightsquigarrow \exists L.\top \not\equiv \bot$ ("$\exists L.\top$ notwendig nicht-leer") bedeutet*:
„Es gibt eine Kette aus R_a und R_b, die zur Sprache gehört."

$\rightsquigarrow \exists L_1.\exists L_2^- \not\equiv \bot$ für zwei kodierte Sprachen L_1 und L_2 bedeutet:
„Es gibt ein Wort, das zu L_1 und zu L_2 gehört."

*) bei entsprechender Tbox!

Leider gilt: Leerheit der Überschneidung kontextfreier Sprachen ist unentscheidbar.

$\rightsquigarrow$ OWL mit Rolleninklusionen ist unentscheidbar

Reguläre Rboxen

Kann man Rolleninklusion zwecks Entscheidbarkeit einschränken?

- Rboxen sind wie Grammatiken für kontextfreie formale Sprachen
- Überschneidungen von kontextfreien Sprachen problematisch

$\rightsquigarrow$ Einschränkung auf reguläre Sprachen!

Reguläre Rboxen

Rollennamen werden mit $\prec$ geordnet (strenge totale Ordnung).
Jede Rbox-Inklusion muss eine der folgenden Formen haben:

- $R \circ R \sqsubseteq R$
- $R^- \sqsubseteq R$
- $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$
- $R \circ S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$
- $S_1 \circ S_2 \circ \ldots \circ S_n \circ R \sqsubseteq R$

Dabei gilt: $S_i \prec R$ für alle $i = 1, 2, \ldots, n$.
Rbox ist regulär, wenn es so eine Ordnung $\prec$ gibt.

Reguläre Rboxen – Beispiel

Beispiel:

$R \circ S \sqsubseteq R \qquad S \circ S \sqsubseteq S \qquad R \circ S \circ R \sqsubseteq T$

$\rightsquigarrow$ ist regulär mit Ordnung $S \prec R \prec T$

Beispiel:

$R \circ T \circ S \sqsubseteq T$

$\rightsquigarrow$ ist nicht regulär (unzulässige Inklusions-Form)

Beispiel:

$R \circ S \sqsubseteq S \qquad S \circ R \sqsubseteq R$

$\rightsquigarrow$ ist nicht regulär (keine gültige Ordnung möglich)

Beschränkung einfacher Rollen

- Einfache Rollen in $\mathcal{SHOIN}$ = Rollen ohne transitive Unterrollen
- In $\mathcal{SROIQ}$: Beachtung der Rolleninklusionen nötig!

Einfache Rollen sind alle Rollen ...

- die nicht auf der rechten Seite einer Rolleninklusion vorkommen,
- die Inverse von anderen einfachen Rollen sind,
- die nur auf der rechten Seite von Rolleninklusionen vorkommen, bei denen links ausschließlich einfache Rollen stehen.

(Achtung: induktive Definition)

$\rightsquigarrow$ nicht-einfach sind Rollen, die direkt oder indirekt von sich selbst abhängen (und deren Überrollen)

Warum ist das wichtig?
Ausdrücke $\leq n\, R.C$, $\geq n\, R.C$, $\mathrm{Irr}(R)$, $\mathrm{Dis}(R, S)$, $\exists R.\mathrm{Self}$ $\neg R(a, b)$
nur für einfache Rollen R und S erlaubt!
(Grund: Sicherstellung von Entscheidbarkeit)

Überblick $\mathcal{SROIQ}$

Klassenausdrücke

Klassennamen	A, B
Konjunktion	$C \sqcap D$
Disjunktion	$C \sqcup D$
Negation	$\neg C$
Exist. Rollenrestr.	$\exists R.C$
Univ. Rollenrestr.	$\forall R.C$
Self	$\exists S.\mathrm{Self}$
Größer-als	$\geq n\ S.C$
Kleiner-als	$\leq n\ S.C$
Nominale	$\{a\}$

Rollen

Rollennamen	R, S, T
einfache Rollen	S, T
Inverse Rollen	R^-
Universelle Rolle	U

Tbox (Klassenaxiome)

Inklusion	$C \sqsubseteq D$
Äquivalenz	$C \equiv D$

Rbox (Rollenaxiome)

Inklusion	$R_1 \sqsubseteq R_2$
Allgemeine Inkl.	$R_1^{(-)} \circ \ldots \circ R_n^{(-)} \sqsubseteq R$
Transitivität	$\mathrm{Tra}(R)$
Symmetrie	$\mathrm{Sym}(R)$
Reflexivität	$\mathrm{Ref}(R)$
Irreflexivität	$\mathrm{Irr}(S)$
Disjunktheit	$\mathrm{Dis}(S, T)$

Abox (Fakten)

Klassenzugehörigkeit	$C(a)$
Rollenbeziehung	$R(a, b)$
Neg. Rollenbeziehung	$\neg S(a, b)$
Gleichheit	$a \approx b$
Ungleichheit	$a \not\approx b$

Wie kompliziert ist $\mathcal{SROIQ}$?

Rückblick: $\mathcal{SHOIN}$ (OWL DL) ist sehr komplex (NExpTime)

Wie komplex ist $\mathcal{SROIQ}$?

Beobachtung: einige Ausdrucksmittel sind nicht wirklich nötig!

- Tra(R) durch $R \circ R \sqsubseteq R$ ausdrückbar
- Sym(R) durch $R^- \sqsubseteq R$ ausdrückbar
- Irr(S) durch $\top \sqsubseteq \neg\exists S.\text{Self}$ ausdrückbar
- Universelle Rolle durch transitive, reflexive Überrolle aller Rollen ersetzbar (hier nicht vertieft)
- Abox durch Nominale darstellbar, z.B. $R(a, b)$ durch $\{a\} \sqsubseteq \exists R.\{b\}$

Qualifizierte Zahlenrestriktionen kaum problematisch (bekannt und implementiert, siehe Vorlesung zu OWL)

$\rightsquigarrow$ Hauptproblem Rollenaxiome (Rbox)

Rolleninklusion, Sprachen, Automaten

Wie geht man mit Rboxen um?

- Rbox-Regeln ähneln formalen Grammatiken
- jede Rolle R definiert eine reguläre Sprache:
  die Sprache der Rollen-Ketten, aus denen R folgt
- reguläre Sprachen $\equiv$ reguläre Ausdrücke $\equiv$ endliche Automaten

$\rightsquigarrow$ Ansatz: Tableauverfahren werden mit „Rbox-Automaten“ erweitert

Details siehe Literaturangaben zu SROIQ

Entscheibarkeit von $\mathcal{SROIQ}$

Tableauverfahren von $\mathcal{SROIQ}$ zeigt:

$\mathcal{SROIQ}$ ist entscheidbar.

- Algorithmus hat gute Anpassungseigenschaften: ungenutze Merkmale belasten die Abarbeitung kaum („pay as you go“)
- aber: Tableau-Verfahren ungeeignet für enge Komplexitätsabschätzungen
- besser: Rückführung auf andere Logiken mit bekannter Komplexität $\rightsquigarrow$ Komplexität von $\mathcal{SROIQ}$: N2ExpTime

OWL 2 DL: Weitere Aspekte

$\mathcal{SROIQ}$ ist „nur“ logische Grundlage von OWL 2 DL

Weitere nicht-logische Aspekte:

- Syntax (Erweiterung nötig)
- Datentypdeklaration und Datentypfunktionen, neue Datentypen?
- Metamodellierung: „Punning“
- Kommentarfunktionen und ontologische Metadaten
- Invers-funktionale konkrete Rollen (DatatypeProperties): Keys?
- Mechanismen zu Ontologieimport?
- …

Metamodellierung

Spezifikation ontologischen Wissens *über* einzelne Elemente der Ontologie (einschließlich Klassen, Rollen, Axiome).

Beispiele:

- „Die Klasse *Person* wurde am 30.1.2008 von *MarkusK* angelegt.“
- „Für die Klasse *Stadt* wird die Property *Einwohnerzahl* empfohlen.“
- „Die Aussage ‚Dresden wurde 1206 gegründet‘ wurde maschinell ermittelt mit einer Sicherheit von 85%.“

Wortspiele in OWL: Punning

Metamodellierung in ausdrucksstarken Logiken ist gefährlich und teuer!

OWL 2. unterstützt zurzeit einfachste Form von Metamodellierung:

Punning

- Bezeichner für Klassen, Rollen, Individuen müssen nicht disjunkt sein
- keine *logische* Beziehung zwischen Klasse, Individuum und Rolle gleichen Namens
- Beziehung nur relevant für pragmatische Interpretation

Beispiel:

$$\mathrm{Person}(\mathrm{Sebastian}) \quad \mathrm{klasseErstelltVon}(\mathrm{Person}, \mathrm{Markus})$$

Kommentare und Metadaten

Punning unterstützt einfache Metadaten mit (schwacher) semantischer Bedeutung

Wie kann man rein syntaktische Kommentare zu einer Ontologie machen?

- Kommentare in XML-Dateien: `<!-- Kommentar -->`
  $\rightsquigarrow$ kein Bezug auf OWL-Axiome dieser Datei
- nicht-logische Annotationen in OWL:
  `owl:AnnotationProperty`
  $\rightsquigarrow$ fest verknüpft mit (semantischem) ontologischem Element, kein syntaktischer Bezug

OWL 2 soll „echte“ syntaktische Kommentare unterstützen

Syntax für OWL 2

zwei grundlegende Varianten:

- Funktionale Syntax: ersetzt „Abstrakte Syntax“ von OWL 1.0
- RDF-Syntax: Erweiterung von OWL/RDF 1.0

$\rightsquigarrow$ funktionale Syntax einfacher zu definieren (keine RDF-Beschränkungen), kompakter

$\rightsquigarrow$ RDF-Syntax für Abwärtskompatibilität wichtig

Quo vadis, OWL Lite?

OWL Lite als Fehlschlag:

- beinahe so komplex wie OWL DL
- komplizierte Syntax gibt keinen direkten Zugang zu wahrer Ausdrucksstärke
- Verwendung in Ontologien heute praktisch nur „zufällig“, nicht bewusst

Ursprüngliches Ziel:
einfach und effizient implementierbarer Teil von OWL

$\rightsquigarrow$ 3 *tractable profiles* für OWL 2: EL, RL und QL

OWL 2 EL

basiert auf Beschreibungslogik $\mathcal{EL}^{++}$

- Konzpete nur mit Konjunktion $C \sqcap D$, Existenz $\exists R.C$, $\top$ und $\bot$
- Nominale
- allgemeine Rolleninklusionen (Rbox), Transitivität

Vorteile:

- polynomielle Komplexität
- schnelle Implementierungen verfügbar
- einfache Definition
- unterstützt praktisch relevante Ontologien (SNOMED)

OWL 2 RL

basiert auf Beschreibungslogik DLP

- in Subklassen-Axiomen sind auf rechter und linker Seite unterschiedliche Arten Klassen erlaubt
- Inverse Rollen, RIAs, einfache Rollenhierarchien
- Abox wie in $\mathcal{SROIQ}$

Vorteile:

- polynomielle Komplexität (verwandt mit Logikprorammierung)
- schnelle Implementierungen verfügbar (Oracle)
- unterstützt RDFS

OWL 2 QL

basiert auf Beschreibungslogik DL Lite$_R$

- Konzpete nur mit Konjunktion $C \sqcap D$, unqualifizierter Existenz $\exists R.\top$, und $\bot$
- Inverse Rollen, einfache Rollenhierarchien
- Abox wie in $\mathcal{SROIQ}$

Vorteile:

- sub-polynomielle Komplexität (verwandt mit relationalen Datenbanken)
- schnelle Implementierungen verfügbar
- relativ einfache Definition

Zusammenfassung

OWL 2 als erste Weiterentwicklung des OWL-Standards

- logische Erweiterung: Beschreibungslogik $\mathcal{SROIQ}$ als Grundlage
- neue Ausdrucksmittel vor allem Rollenaxiome, qualifizierte Zahlenrestriktionen
- nicht-logische Erweiterungen: Punning, Kommentare, Datentypen, u.a.
- Einführung von 3 neuen OWL profiles mit polynomieller Komplexität
- Verabschiedung Oktober 2009