Vorlesung Semantic Web

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

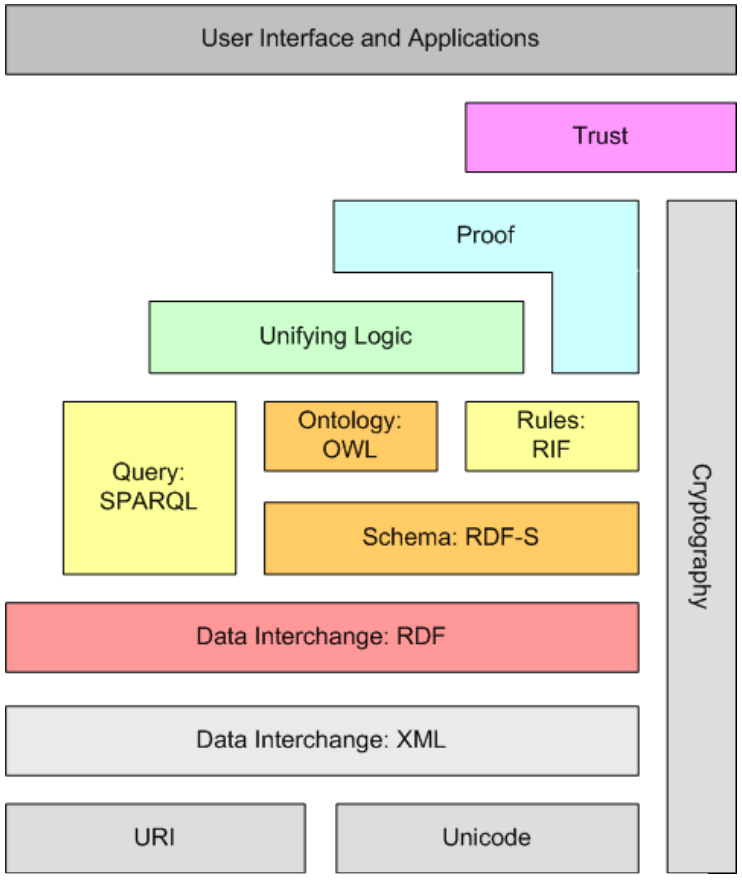
Fachgebiet Knowledge Engineering

Semantic Web – Aufbau

here be dragons...

Semantic-Web-Technologie (Fokus der Vorlesung)

Technische Grundlagen



Berners-Lee (2009): *Semantic Web and Linked Data*
http://www.w3.org/2009/Talks/0120-campus-party-tbl/

Was bisher geschah

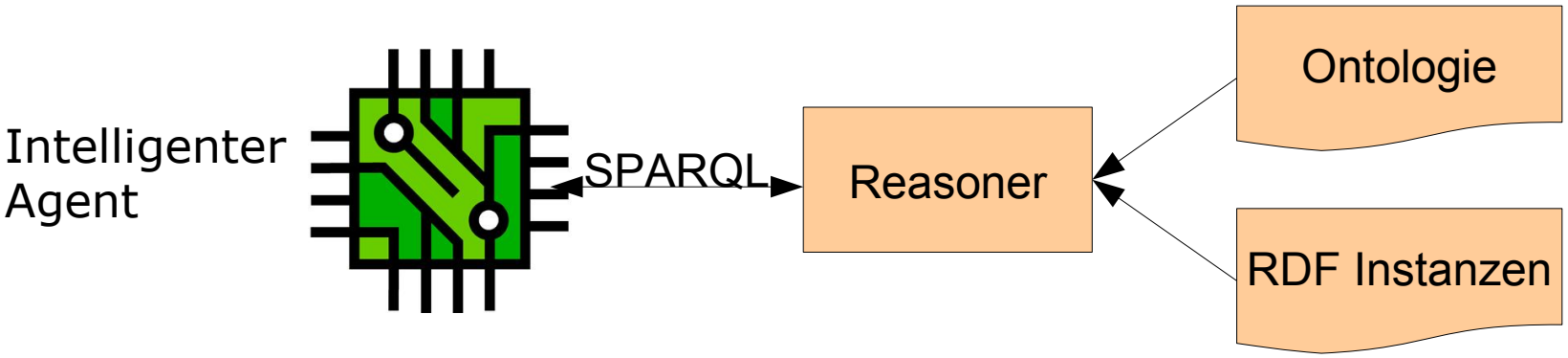
TECHNISCHE UNIVERSITÄT DARMSTADT

- Komplexe Ontologien
  - RDF Schema ist leichtgewichtig
  - OWL kann mehr
- Reasoning
  - für RDF Schema können wir das schon
  - wie funktioniert Reasoning mit OWL?

Recap: Kombination von SPARQL & Reasoning

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reasoning mit RDF Schema haben wir schon kennen gelernt
- Viele Reasoner haben auch eine SPARQL-Schnittstelle



Aufgaben für einen Reasoner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was würden wir gern von einem Reasoner wissen?
  - Subklassenbeziehung
    - z.B.: Sind alle Säugetiere Landbewohner?
  - Klassenäquivalenz
    - z.B.: Sind alle Wasserbewohner mit Flossen Fische und umgekehrt?
  - Klassendisjunktheit
    - z.B.: Gibt es Tiere, die zu den Säugetieren und zu den Vögeln gehören?
  - Klassenkonsistenz
    - z.B.: Kann es eierlegende Säugetiere geben?
  - Instanzbeziehung
    - z.B.: Ist Flipper ein Delfin?
  - Klassenaufzählung
    - z.B.: Liste alle Elefanten auf

Aufgaben für einen Reasoner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das sind ziemlich viele mögliche Aufgaben
- Macht einen Reasoner ziemlich kompliziert

- Naiver Forward-Chaining-Algorithmus
  - ist nicht besonders gut skalierbar
  - kann nicht mit Konjunktion umgehen
    - die gibt es aber in OWL, z.B. unionOf

Aufgaben für einen Reasoner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ansatz: alle Aufgaben auf ein gemeinsames Grundproblem zurückführen
- Zum Beispiel: Widerspruchsfreiheit
  - d.h.: es lässt sich nicht gleichzeitig eine Aussage und ihr Gegenteil aus T-Box und A-Box ableiten

Beispiel: Widerspruchsfreiheit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel:

```
:Man a owl:Class .
:Woman a owl:Class .
:Man owl:disjointWith :Woman .

:Alex a :Man .
:Alex a :Woman .
```

Beispiel: Widerspruchsfreiheit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Daraus folgt:
  - :Man ∩ :Woman = ∅

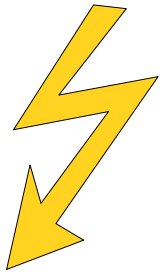
```
owl:Nothing owl:intersectionOf (:Man :Woman) .
```

  - :Alex ∈ (:Man ∩ :Woman)

```
:Alex a [ a owl:Class; owl:intersectionOf (:Man :Woman)] .
```

- also:
  - :Alex ∈ ∅

```
:Alex a owl:Nothing .
```



Neudefinition der Reasoner-Tasks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Subklassenbeziehung

  Student $\subseteq$ Person $\Leftrightarrow$ Student(x) $\rightarrow$ Person(x)
- Prüfungsmethode: indirekter Beweis
  - "Erfinde" eine Instanz i
  - Setze Student(i) und $\neg$Person(i)
  - Prüfe auf Widerspruchsfreiheit
    - bei Widerspruch: Student $\subseteq$ Person muss gelten
    - kein Widerspruch: Student $\subseteq$ Person kann nicht abgeleitet werden

Beispiel: Subklassenbeziehung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ontologie:

```
:Student owl:subClassOf :UniversityMember .
:UniversityMember owl:subClassOf :Person .
```

- Neu hinzugefügte Instanz:

```
:i a :Student .
:i a [ owl:complementOf :Person ] .
```

Beispiel: Subklassenbeziehung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aus

  ```
  :i a :Student .
  ```

  und

  ```
  :Student owl:subClassOf :UniversityMember .
  ```

  folgt

  ```
  :i a :UniversityMember .
  ```

- analog folgt mit

  ```
  :UniversityMember owl:subClassOf :Person .
  ```

- auch

  ```
  :i a Person .
  ```

Beispiel: Subklassenbeziehung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jetzt haben wir

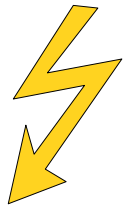
```
:i a :Person .
:i a [ owl:complementOf :Person ] .
```

damit gilt

```
:i a [ owl:intersectionOf (:Person
                           [ owl:complementOf :Person ])] .
```

- und folglich

```
:i a owl:Nothing .
```



Neudefinition der Reasoner-Tasks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Klassenäquivalenz
  - Person ≡ Mensch
- Das kann man auflösen in
  - Person ⊆ Mensch und
  - Mensch ⊆ Person
- Also: zweimal Subklassenbeziehung zeigen
  - können wir schon

- Klassendisjunktheit
  - Sind C und D disjunkt?
  - "Erfinde" eine Instanz i
  - setze C(i) und D(i)
  - das haben wir schon gesehen (Beispiel: Alex)

Klassenkonsistenz

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kann eine Klasse Instanzen haben?
  - zum Beispiel: verheiratete Junggesellen

```
:Junggeselle owl:subClassOf :Mann .
:Junggeselle owl:subClassOf
  [ a owl:Restriction;
    owl:onProperty :verheiratetMit;
    owl:cardinality 0 ] .
:VerheiratetePerson owl:subClassOf [
    a owl:Restriction;
    owl:onProperty :verheiratetMit;
    owl:cardinality 1 ] .

:VerheirateterJunggeselle owl:intersectionOf
  (:Junggeselle :VerheiratetePerson) .
```

Klassenkonsistenz

TECHNISCHE UNIVERSITÄT DARMSTADT

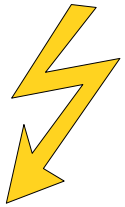
- Prüfen auf Klassenkonsistenz:
  - "Erfinde" eine Instanz i
  - Füge sie als Instanz der Klasse hinzu
  - Prüfe, ob ein Widerspruch entsteht

Beispiel: Klassenkonsistenz

TECHNISCHE UNIVERSITÄT DARMSTADT

- In unserem Beispiel entsteht:

```
:i a [
    a owl:Restriction;
    owl:onProperty :verheiratetMit;
    owl:cardinality 1 ] .
```



- und

```
:i a [
    a owl:Restriction;
    owl:onProperty :verheiratetMit;
    owl:cardinality 0 ] .
```

Aufgaben für einen Reasoner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Instanzbeziehung
  - Ist Flipper ein Delfin?
- Prüfung:
  - setze ¬Delfin(Flipper)
  - prüfe auf Widerspruch

- Instanzen einer Klasse aufzählen
  - Prüfe die Instanzbeziehung für alle bekannten Individuen durch

Aufgaben für einen Reasoner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was würden wir gern von einem Reasoner wissen?
  - Subklassenbeziehung
    - z.B.: Sind alle Säugetiere Landbewohner?
  - Klassenäquivalenz
    - z.B.: Sind alle Wasserbewohner mit Flossen Fische und umgekehrt?
  - Klassendisjunktheit
    - z.B.: Gibt es Tiere, die zu den Säugetieren und zu den Vögeln gehören?
  - Klassenkonsistenz
    - z.B.: Kann es eierlegende Säugetiere geben?
  - Instanzbeziehung
    - z.B.: Ist Flipper ein Delfin?
  - Klassenaufzählung
    - z.B.: Liste alle Elefanten auf

Aufgaben für einen Reasoner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wir haben jetzt gesehen:
  - alle möglichen Aufgaben lassen sich auf dieselbe Aufgabe herunterbrechen
  - nämlich Prüfung auf Konsistenz

- Jetzt muss ein Reasoner also nur noch einen Task implementieren

Tableau-Verfahren

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das heute am häufigsten eingesetzte Reasoning-Verfahren
- kommt aus dem Bereich der Prädikatenlogik

DL-Schreibweisen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Klassen und Instanzen
  - C(x) ↔ `x a C .`
  - R(x,y) ↔ `x R y .`
  - $C \sqsubseteq D$ ↔ `C rdfs:subClassOf D`
  - $C \equiv D$ ↔ `C owl:equivalentClass D`
  - $C \sqsubseteq \neg D$ ↔ `C owl:disjointWith D`
  - $C \equiv \neg D$ ↔ `C owl:complementOf D`
  - $C \equiv D \sqcap E$ ↔ `C owl:intersectionOf (D E) .`
  - $C \equiv D \sqcup E$ ↔ `C owl:unionOf (D E) .`
  - $\top$ ↔ `owl:Thing`
  - $\bot$ ↔ `owl:Nothing`

DL-Schreibweisen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Restriktionen
  - $C \sqsubseteq \forall R.D$ ↔
```
C owl:subClassOf
  [ a owl:Restriction;
    owl:onProperty R;
    owl:allValuesFrom D ] .
```
  - $C \sqsubseteq \exists R.D$ ↔
```
C owl:subClassOf
  [ a owl:Restriction;
    owl:onProperty R;
    owl:someValuesFrom D ] .
```
  - $\exists R.T \sqsubseteq C$ ↔ `R rdfs:domain C .`
  - $T \sqsubseteq \forall R.C$ ↔ `R rdfs:range C .`

DL-Schreibweisen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Abkürzung für universell gültige Aussagen
  - $D \sqcap E \quad \leftrightarrow \forall x: x \in D \sqcap E$
  - $D \sqcup E \quad \leftrightarrow \forall x: x \in D \sqcup E$
  - $\exists R.C \quad \leftrightarrow \forall x: x \in \exists R.C$
    $\leftrightarrow \forall x: \exists y: y \in C \wedge R(x,y)$
  - $\forall R.C \quad \leftrightarrow \forall x: x \in \forall R.C$
    $\leftrightarrow \forall x: R(x,y) \rightarrow y \in C$

Vorbereitung: Ontologien in Negationsnormalform (NNF)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Negationsnormalform:
  - $\sqsubseteq$ und $\equiv$ kommen nicht vor
  - Negation nur vor atomaren Klassen oder Aussagen

- Eine vereinfachte Schreibweise für Ontologien
- Von Tableau-Reasonern als Eingabe verlangt

- Wie kommt man da hin?

Ontologien in Negationsnormalform (NNF)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Elimination von $\sqsubseteq$:
  - Ersetze $C \sqsubseteq D$ durch $\neg C \sqcup D$
  - Kurzschreibweise für: $\forall x: \neg C(x) \vee D(x)$

- Warum gilt diese Äquivalenz?
  - $C \sqsubseteq D$ ist äquivalent zu $C(x) \rightarrow D(x)$

C(x)	D(x)	C(x) → D(x)	¬C(x) v D(x)
true	true	true	true
true	false	false	false
false	true	true	true
false	false	true	true

Ontologien in Negationsnormalform

TECHNISCHE UNIVERSITÄT DARMSTADT

- Elimination von ≡:
  - Ersetze C ≡ D durch C ⊑ D und D ⊑ C
  - Verfahre wie gehabt

Ontologien in Negationsnormalform (NNF)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Transformationsregeln
  - NNF(C) = C (für atomare C)
  - NNF(¬C) = ¬C (für atomare C)
  - NNF(¬ ¬C) = 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)

Ontologien in Negationsnormalform (NNF)

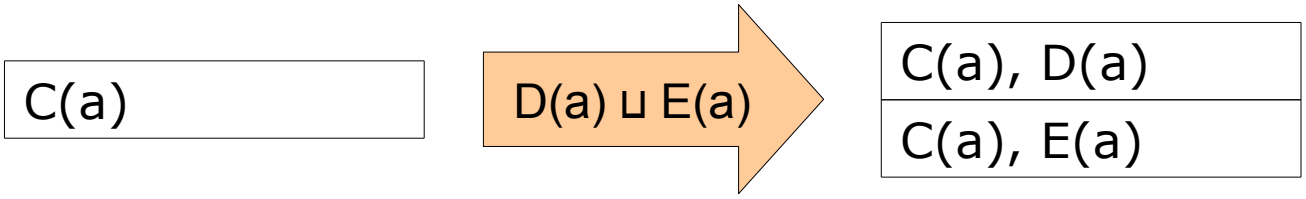
TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel: ein Verein kann natürliche und juristische Personen als Mitglieder haben
  - :hasMember rdfs:range
    [owl:unionOf (:NaturalPerson :LegalPerson)]
  - $T \sqsubseteq \forall hasMember.(NaturalPerson \sqcup LegalPerson)$
  - $\neg T \sqcup \forall hasMember.(NaturalPerson \sqcup LegalPerson)$
  - Damit sind alle Aussagen in NNF
    (keine Negation vor nicht-Elementaren Ausdrücken)

Der einfache Tableau-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Tableau: Sammlung von abgeleiteten Aussagen
  - wird nach und nach erweitert
  - wie beim Forward Chaining
- Bei Konjunktion
  - Teilen in zwei Tableaus



Wann ist eine Wissensbasis widerspruchsfrei?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Tableau wird kontinuierlich erweitert und geteilt
- Widerspruchsfreiheit, wenn...
  - keine weiteren Axiome erzeugt werden können
  - mindestens ein Teiltableau ohne Widerspruch bleibt
  - Ein Teiltableau enthält einen Widerspruch, wenn es ein Axiom und sein direktes Gegenteil enthält
    - z.B. Mensch(Hans) und ¬Mensch(Hans)
    - man sagt dann, dass das Teiltableau *abgeschlossen* ist

Der einfache Tableau-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: eine Wissensbasis W in NNF

  Solange nicht alle Teiltableaus abgeschlossen sind

  * Wähle ein nicht abgeschlossenes Teiltableau T und ein A ∈ W ∪ T

  Wenn A noch nicht in T enthalten

  Wenn A eine A-Box-Aussage ist

  füge A zu T hinzu

  zurück zu *

  Wenn A eine T-Box-Aussage ist

  Wähle ein Individuum a ∈ W ∪ T

  Füge A(a) zu T hinzu

  zurück zu *

  sonst

  Erweitere das Tableau mit Konsequenzen aus A

  zurück zu *

Der einfache Tableau-Algorithmus

- Erweiterungsregeln für eine Aussage A

Nr	Aussage	Aktion
1	C(a)	Füge C(a) hinzu
2	R(a,b)	Füge R(a,b) hinzu
3	C	Wähle ein Individuum a, füge C(a) hinzu
4	(C ⊓ D)(a)	Füge C(a) und D(a) hinzu
5	(C ⊔ D)(a)	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu
6	(∃R.C)(a)	Füge R(a,b) und C(b) für ein *neues* Individuum b hinzu
7	(∀R.C)(a)	Für alle b mit R(a,b) ∈ T: füge C(b) hinzu

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben folgende Ontologie:

  :Animal owl:unionOf (:Mammal :Bird :Fish :Insect) .
  :Animal owl:disjointWith :Human .
  :Seth a :Human .
  :Seth a :Insect .

- Ist diese Wissensbasis konsistent?

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben folgende Ontologie:

  :Animal owl:disjointWith :Human .
  :Animal owl:unionOf (:Mammal :Bird :Fish :Insect) .
  :Seth a :Human .
  :Seth a :Insect .

  - Dieselbe Ontologie in DL-NNF:

    $\neg$Animal $\sqcup$ $\neg$Human
    Animal $\sqcup$ ($\neg$Mammal $\sqcap$ $\neg$Bird $\sqcap$ $\neg$Fish $\sqcap$ $\neg$Insect)
    $\neg$Animal $\sqcup$ (Mammal $\sqcup$ Bird $\sqcup$ Fish $\sqcup$ Insect)
    Human(Seth)
    Insect(Seth)

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Human(Seth), Insect(Seth)

Nr	Aussage	Aktion
1	C(a)	Füge C(a) hinzu

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Human(Seth), Insect(Seth),
(¬Animal ⊔ ¬Human)(Seth)

Nr	Aussage	Aktion
3	C	Wähle ein Individuum a, füge C(a) hinzu

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Human(Seth), Insect(Seth), ¬Animal(Seth)
Human(Seth), Insect(Seth), ¬Human(Seth)

Nr	Aussage	Aktion
5	(C ⊔ D)(a)	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Human(Seth), Insect(Seth), ¬Animal(Seth) Animal ⊔ (¬Mammal ⊓ ¬Bird ⊓ ¬Fish ⊓ ¬Insect)(Seth)
Human(Seth), Insect(Seth), ¬Human(Seth)

Nr	Aussage	Aktion
3	C	Wähle ein Individuum a, füge C(a) hinzu

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Human(Seth), Insect(Seth), ¬Animal(Seth) Animal(Seth)
Human(Seth), Insect(Seth), ¬Animal(Seth) (¬Mammal ⊓ ¬Bird ⊓ ¬Fish ⊓ ¬Insect)(Seth)
Human(Seth), Insect(Seth), ¬Human(Seth)

Nr	Aussage	Aktion
5	(C ⊔ D)(a)	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu

Ein einfaches Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Human(Seth), Insect(Seth), ¬Animal(Seth) Animal(Seth)
Human(Seth), Insect(Seth), ¬Animal(Seth) (¬Mammal ⊓ ¬Bird ⊓ ¬Fish ⊓ ¬Insect)(Seth) ¬Mammal(Seth) ⊓ ¬Bird(Seth) ⊓ ¬Fish(Seth) ⊓ ¬Insect(Seth)
Human(Seth), Insect(Seth), ¬Human(Seth)

Nr	Aussage	Aktion
4	(C ⊓ D)(a)	Füge C(a) und D(a) hinzu

Der einfache Tableau-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: eine Wissensbasis W in NNF

  Solange nicht alle Teiltableaus abgeschlossen sind

  * Wähle ein nicht abgeschlossenes Teiltableau T und ein A ∈ W ∪ T

  Wenn A noch nicht in T enthalten

  Wenn A eine A-Box-Aussage ist

  füge A zu T hinzu

  zurück zu *

  Wenn A eine T-Box-Aussage ist

  Wähle ein Individuum a ∈ W ∪ T

  Füge A(a) zu T hinzu

  zurück zu *

  sonst

  Erweitere das Tableau mit Konsequenzen aus A

  zurück zu *

Der einfache Tableau-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Funktioniert prima, wenn wir einen Widerspruch finden
  - aber wo ist das Abbruchkriterium für widerspruchsfreie Wissensbasen?

Nr	Aussage	Aktion
6	(∃R.C)(a)	Füge R(a,b) und C(b) für ein *neues* Individuum b hinzu

  - Diese Regel ist potentiell gefährlich!

Noch ein Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben folgende Ontologie:

```
:ParentsOfSons owl:subClassOf [
  a owl:Restriction;
  owl:onProperty :hasChild;
  owl:someValuesFrom :Man ] .
:Peter :hasChild :Julia .
:Julia a :Woman .
```

- in DL-NNF:

  $\neg$ParentsOfSons $\sqcup$ $\exists$hasChild.Man
  hasChild(Peter,Julia)
  Woman(Julia)

Noch ein Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

hasChild(Peter,Julia)

Nr	Aussage	Aktion
2	R(a,b)	Füge R(a,b) hinzu

Noch ein Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

hasChild(Peter,Julia), Woman(Julia)

Nr	Aussage	Aktion
1	C(a)	Füge C(a) hinzu

Noch ein Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

hasChild(Peter,Julia), Woman(Julia),
(¬ParentsOfSons ⊔ ∃hasChild.Man)(Peter)

Nr	Aussage	Aktion
3	C	Wähle ein Individuum a, füge C(a) hinzu

Noch ein Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

hasChild(Peter,Julia), Woman(Julia), (¬ParentsOfSons ⊔ ∃hasChild.Man)(Peter), ¬ParentsOfSons(Peter)
hasChild(Peter,Julia), Woman(Julia), (¬ParentsOfSons ⊔ ∃hasChild.Man)(Peter), ∃hasChild.Man(Peter)

Nr	Aussage	Aktion
5	(C ⊔ D)(a)	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu

Noch ein Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

hasChild(Peter,Julia), Woman(Julia), (¬ParentsOfSons ⊔ ∃hasChild.Man)(Peter), ¬ParentsOfSons(Peter)
hasChild(Peter,Julia), Woman(Julia), (¬ParentsOfSons ⊔ ∃hasChild.Man)(Peter), ∃hasChild.Man(Peter), hasChild(Peter,b0),Man(b0)

Nr	Aussage	Aktion
6	(∃R.C)(a)	Füge R(a,b) und C(b) für ein *neues* Individuum b hinzu

Noch ein Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

hasChild(Peter,Julia), Woman(Julia), (¬ParentsOfSons ⊔ ∃hasChild.Man)(Peter), ¬ParentsOfSons(Peter)
hasChild(Peter,Julia), Woman(Julia), (¬ParentsOfSons ⊔ ∃hasChild.Man)(Peter), ∃hasChild.Man(Peter), hasChild(Peter,b0),Man(b0), hasChild(Peter,b1),Man(b1), ...

Nr	Aussage	Aktion
6	(∃R.C)(a)	Füge R(a,b) und C(b) für ein *neues* Individuum b hinzu

Tableau-Algorithmus mit Blocking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Terminiert nicht unbedingt
- Man kann beliebig viele neue Axiome erzeugen

Nr	Aussage	Aktion
6	(∃R.C)(a)	Füge R(a,b) und C(b) für ein *neues* Individuum b hinzu

- Idee: wenn keine neue Information erzeugt wird, dann verhindere die Anwendung von Regel 6
  - d.h., wenn schon eine gleichartige Instanz b0 erzeugt wurde, dann blockiere die Auswahl von Regel 6 in diesem Fall.

Tableau-Algorithmus mit Blocking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: eine Wissensbasis W in NNF

  Solange nicht alle Teiltableaus abgeschlossen sind und weitere Aussagen erzeugt werden können

  * Wähle ein nicht abgeschlossenes Teiltableau T und ein A ∈ W ∪ T, das nicht blockiert ist
  Wenn A noch nicht in T enthalten
  Wenn A eine A-Box-Aussage ist
  füge A zu T hinzu
  zurück zu *
  Wenn A eine T-Box-Aussage ist
  Wähle ein Individuum a ∈ W ∪ T
  Füge A(a) zu T hinzu
  zurück zu *
  sonst
  Erweitere das Tableau mit Konsequenzen aus A
  zurück zu *

Tableau-Algorithmus: Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Algorithmus für Beschreibungslogiken
  - passt auf OWL Lite und DL
- Wir haben Beispiele für einige OWL-Konstrukte gesehen
  - Auch andere OWL-DL-Ausdrücke können in DL "übersetzt" werden
  - und auch hier gibt es Expansionsregeln
  - allerdings machen die das Reasoning nicht immer leichter
    - benötigen teilweise komplizierte Strategien
    - dynamisches Blockieren und Deblockieren

Tableau-Algorithmus: Optimierungsansätze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: eine Wissensbasis W in NNF

  Solange nicht alle Teiltableaus abgeschlossen sind und weitere Aussagen erzeugt werden können

  * Wähle ein nicht abgeschlossenes Teiltableau T und ein A ∈ W ∪ T, das nicht blockiert ist
  Wenn A noch nicht in T enthalten
  Wenn A eine A-Box-Aussage ist
  füge A zu T hinzu
  zurück zu *
  Wenn A eine T-Box-Aussage ist
  Wähle ein Individuum a ∈ W ∪ T
  Füge A(a) zu T hinzu
  zurück zu *
  sonst
  Erweitere das Tableau mit Konsequenzen aus A
  zurück zu *

Tableau-Algorithmus: Implementierungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fact
  - University of Manchester, kostenlos
  - SHIQ
- Fact++/JFact
  - Weiterentwicklung von Fact, kostenlos
  - SHOIQ(mit etwas D), OWL-DL + OWL2
- Pellet
  - Clark & Parsia, kostenlos für akademische Nutzung
  - SHOIN(D), OWL-DL + OWL2
- RacerPro
  - Racer Systems, kommerzielles Produkt
  - SHIQ(D)

Reasoning mit OWL: Alternative Ansätze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wiederverwendung von Reasonern aus anderen Bereichen
  - z.B. Datalog-Reasoner (KAON)
  - z.B. First Order Logic Theorem Prover (Hoolet)

Reasoning: aktuelle Forschung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gute Heuristiken für Auswahl von Axiomen
- Unterstützte Expressivität erhöhen
  - z.B. auch Regeln ermöglichen
  - dabei Skalierbarkeit erlauben
- Paralleles Reasoning

Zur Erholung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gönnen wir unserem Reasoner doch mal etwas Zerstreuung...

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku in OWL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was haben wir?
- Zunächst mal eine abgeschlossene Klasse von Zahlen:

```
:Zahl a owl:Class ;
  owl:oneOf (:1 :2 :3 :4 :5 :6 :7 :8 :9) .
```

- Und eine ganze Menge von Feldern
  - die wir mit diesen Zahlen füllen wollen
  - machen wir's uns einfach: die Felder sind auch Zahlen
  - wir wollen wissen, welches Feld gleich welcher Zahl ist

Sudoku als OWL-Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- 81 Felder:

```
c1_11 a :Zahl .
c1_21 a :Zahl .
…
c1_33 a :Zahl .
c2_11 a :Zahl .
…
c9_33 a :Zahl .
```

c1_11	c1_12		c2_11	c2_12				
c1_21								
c4_11								

Sudoku als OWL-Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Felder in einem Quadranten sind verschieden:

```
c1_11 owl:differentFrom
  c1_12, c1_13, …, c1_33 .
c1_12 owl:differentFrom
  c1_13, c1_21, …, c1_33 .
…
c1_32 owl:differentFrom
  c1_33 .
c2_11 owl:differentFrom
  c2_12, c2_13, …, c1_33 .
…
c9_32 owl:differentFrom
  c9_33 .
```

c1_11	c1_12		c2_11	c2_12				
c1_21								
c4_11								

Sudoku als OWL-Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Felder in einer Zeile sind verschieden:

```
c1_11 owl:differentFrom
  c1_12, c1_13, …, c3_13 .
…
```

c1_11	c1_12		c2_11	c2_12				
c1_21								
c4_11								

Sudoku als OWL-Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Felder in einer Spalte sind verschieden:

```
c1_11 owl:differentFrom
  c1_21, c1_31, …, c3_31 .
…
```

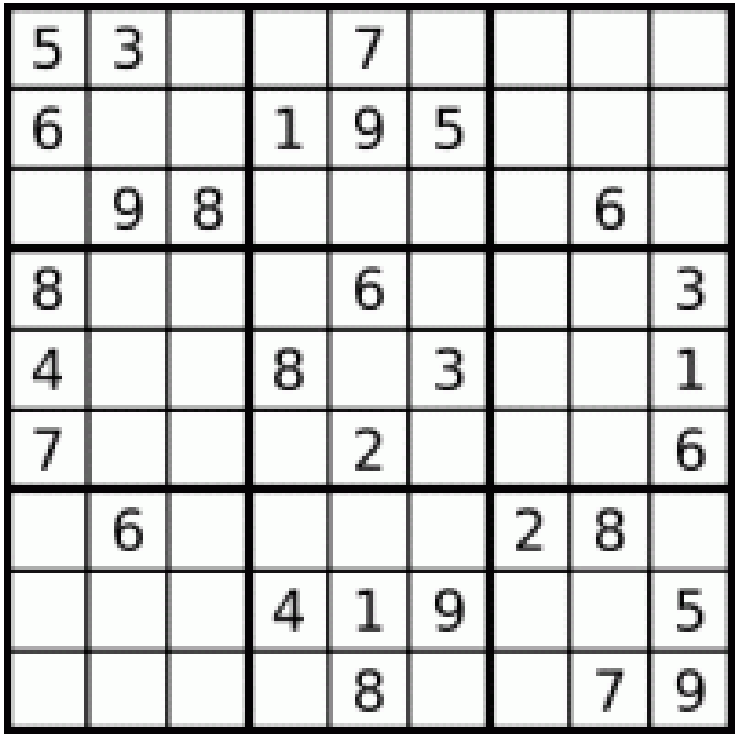
c1_11	c1_12		c2_11	c2_12				
c1_21								
c4_11								

Sudoku in OWL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Letzter Schritt: bekannte Zahlen einsetzen

```
c1_11 owl:sameAs :5 .
c1_12 owl:sameAs :3 .
c1_21 owl:sameAs :6 .
…
```



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Komplexität unseres OWL-Problems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wir haben
  - Klassen, und zwar eine abgeschlossene
- d.h., eine Ontologie der Komplexität SO.
- Wer kann das?
  - Fact: SHIQ :-(
  - RacerPro: SHIQ(D) :-(
  - Pellet: SHOIN(D) :-)
  - HermiT: SHOIQ :-)

Sudoku in OWL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probe aufs Exempel in Protégé
  - mit HermiT-Reasoner
- Vereinfacht, damit wir nicht so viel tippen müssen

	4		
		3	
		2	
1			

Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reasoning auf Ontologien
  - lässt sich auf Konsistenzprüfung zurückführen
  - in OWL DL: Description Logic Reasoning
- Tableau-Verfahren
  - erzeuge alle möglichen Aussagen
  - verzweige bei "oder"
  - blockiere unproduktive Axiome

Vorlesung Semantic Web

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering