Vorlesung Semantic Web

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben folgende Ontologie:

```
:Baby rdfs:subClassOf :Human .
:Adult rdfs:subClassOf :Human .
:Andy a :Baby .
:Andy a :Human .
```

- Zeigen Sie, dass das Tableau-Verfahren darin keinen Widerspruch erkennt.

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Dieselbe Ontologie in DL-NNF:

  $\neg$Baby $\sqcup$ Human
  $\neg$Adult $\sqcup$ Human
  Baby(Andy)
  Adult(Andy)

- Damit können wir jetzt das Tableau-Verfahren starten

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Baby(Andy), Adult(Andy)

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

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Baby(Andy), Adult(Andy),
(¬Baby ⊔ Human)(Andy),
(¬Adult ⊔ Human)(Andy)

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

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Baby(Andy), Adult(Andy),
$(\neg \text{Baby} \sqcup \text{Human})(\text{Andy})$,
$(\neg \text{Adult} \sqcup \text{Human})(\text{Andy})$

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

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Baby(Andy), Adult(Andy), (¬Adult ⊔ Human)(Andy), ¬Baby(Andy)
Baby(Andy), Adult(Andy), (¬Adult ⊔ Human)(Andy), Human(Andy).

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

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Baby(Andy), Adult(Andy), (¬Adult ⊔ Human)(Andy), ¬Baby(Andy)
Baby(Andy), Adult(Andy), Human(Andy), ¬Adult(Andy)
Baby(Andy), Adult(Andy), Human(Andy)

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

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das dritte Tableau bleibt ohne Widerspruch
- weitere Axiome können nicht erzeugt werden

→ damit erkennt der Reasoner keinen Widerspruch!

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fügen Sie nun folgendes zusätzliche Axiom ein:

```
:Baby owl:disjointWith :Adult .
```

- Zeigen Sie, dass das Tableau-Verfahren hier einen Widerspruch finden kann.

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Damit sind unsere DL-NNF so aus:

  ¬Baby ⊔ Human
  ¬Adult ⊔ Human
  ¬Baby ⊔ ¬Adult
  Baby(Andy)
  Adult(Andy)

- ...und das Tableau-Verfahren ist noch nicht fertig

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Baby(Andy), Adult(Andy), (¬Adult ⊔ Human)(Andy), ¬Baby(Andy)
Baby(Andy), Adult(Andy), Human(Andy), ¬Adult(Andy)
Baby(Andy), Adult(Andy), Human(Andy), (¬Adult ⊔ ¬Baby)(Andy)

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

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Baby(Andy), Adult(Andy), (¬Adult ⊔ Human)(Andy), ¬Baby(Andy)
Baby(Andy), Adult(Andy), Human(Andy), ¬Adult(Andy)
Baby(Andy), Adult(Andy), Human(Andy), ¬Adult(Andy)
Baby(Andy), Adult(Andy), Human(Andy), ¬Baby(Andy)

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

Aufgabe 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- In diesem Fall werden alle Zweige des Tableaus zum Widerspruch geführt
- Damit existiert insgesamt ein Widerspruch!

Aufgabe 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das Datenset Linked Open Numbers enthält die Definition für 1 Milliarde Zahlen, inklusive ihrer Vorgänger und Nachfolger und weiterer Informationen, wie den Primfaktoren. Finden Sie einen effizienten(!) Weg, die paarweise Verschiedenheit dieser Zahlen zu definieren.

- *Tipp:* Wenn Sie den Nachfolger bereits für alle Zahlen definiert haben, reicht hier weniger als ein Dutzend zusätzlicher Axiome aus!

Aufgabe 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was haben wir schon?

```
:next a owl:ObjectProperty ;
      rdfs:domain :Number ;
      rdfs:range :Number .
```

- Was können wir damit machen?

```
:smallerThan a owl:ObjectProperty ,
                owl:TransitiveProperty .
:next rdfs:subPropertyOf :largerThan .
```

Aufgabe 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Damit folgt schon mal die allgemeine "kleiner-als"-Relation aus den Nachfolgern:

```
:2 :next :3 .
:3 :next :4 .

:2 :lessThan :3 .
:3 :lessThan :4 .

:2 :lessThan :4 .
```

Aufgabe 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definieren wir jetzt noch folgendes:

```
:lessThan rdfs:subPropertyOf owl:differentFrom .
```

- Damit folgt zusätzlich

```
:2 owl:differentFrom :4 .
```

Aufgabe 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jetzt müssen wir uns noch um den anderen Fall kümmern (erste Zahl größer als die zweite)

```
:largerThan a owl:ObjectProperty .
            owl:inversePropertyOf :previous .
            rdfs:subPropertyOf owl:differentFrom .

:previous a owl:ObjectProperty ;
          owl:inversePropertyOf :next .
          rdfs:subPropertyOf :largerThan .
```

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Zwischen Objekten der Klasse *Number* sind unter anderem die Relationen *previous, next, lessThan, greaterThan* und *primefactor* definiert.

- Formulieren Sie unter Verwendung (nur) dieser Relationen folgende Abfragen in SPARQL:
  - Finde alle geraden Zahlen.
  - Finde alle Zahlen, die direkter Nachfolger eines ihrer Primfaktoren sind.
  - Finde alle ungeraden Zahlen.
  - Finde alle Primzahlen.
  - Finde alle Nicht-Primzahlen.
  - Finde alle Primzahl-Zwillinge (d.h. zwei Primzahlen, bei denen die eine um zwei größer ist als die andere).

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Finde alle geraden Zahlen.
- Idee: gerade Zahlen haben 2 als Primfaktor.

```
SELECT ?n WHERE {?n :primefactor :2}
```

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Finde alle Zahlen, die Nachfolger eines ihrer Primfaktoren sind
- Das geht relativ straight forward:

```
SELECT ?n WHERE   { ?n :primefactor ?x .
                    ?x :next ?n }
```

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Finde alle ungeraden Zahlen
- Idee: ungerade Zahlen haben *nicht* 2 als Teiler
  - da brauchen wir den Verneinungstrick
  - naiv:

```
SELECT ?n WHERE    { ?n a :Number . }
          OPTIONAL { ?n :primefactor :2 }
          FILTER   {!BOUND(?n)}
```

  - Hoppla, das klappt wohl nicht!
    - Wenn wir an n interessiert sind,
      können wir n nicht zur Abfrage mit BOUND verwenden

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Finde alle ungeraden Zahlen
- Idee: ungerade Zahlen haben *nicht* 2 als Teiler
  - da brauchen wir den Verneinungstrick
  - wir brauchen also eine *zweite* Variable für unsere Zahl

```
SELECT ?n WHERE    { ?n :next ?y }
          OPTIONAL { ?y :previous ?z .
                     ?z :primefactor :2 }
          FILTER   {!BOUND(?z)}
```

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wie funktioniert diese Lösung?

?n	?y	?z
1	2	-
2	3	2
3	4	-
4	5	4
...	...	...

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Finde alle Nicht-Primzahlen
- *Idee:* Alle Zahlen, die einen Primfaktor größer als 1 und kleiner sich selbst haben

```
SELECT ?n WHERE { ?n :primefactor ?x .
                  ?x :greaterThan :1 .
                  ?x :lessThan ?n }
```

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

- Finde alle Primzahlen
- *Idee:* es darf keinen Primfaktor zwischen 1 und n geben
  - Verneinungstrick

```
SELECT ?n WHERE     { ?n a :Number }
          OPTIONAL  { ?n :primefactor ?x .
                      ?x :greaterThan :1 .
                      ?x :lessThan ?n }
          FILTER    { !BOUND(?x) }
```

Aufgabe 3

TECHNISCHE UNIVERSITÄT DARMSTADT

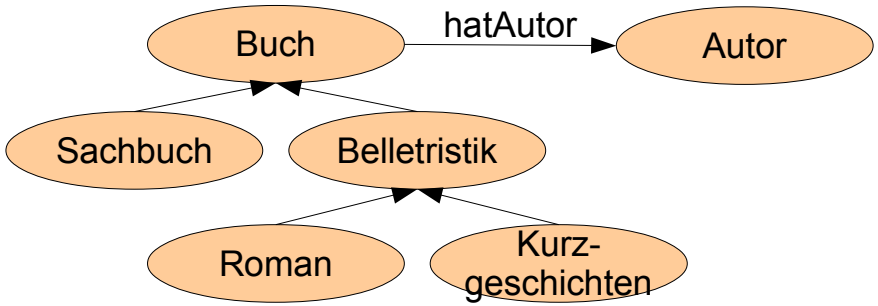
- Finde alle Primzahlzwillinge
- Kombination aus doppelter Lösung der vorherigen

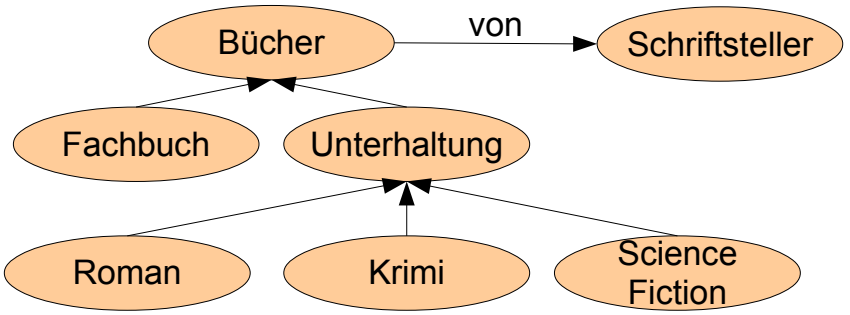
```
SELECT ?n1 ?n2 WHERE    { ?n1 :next ?n . ?n :next ?n2 }
               OPTIONAL { ?n1 :primefactor ?x1 .
                          ?x1 :greaterThan :1 .
                          ?x1 :lessThan ?n1 }
               OPTIONAL { ?n2 :primefactor ?x2 .
                          ?x2 :greaterThan :1 .
                          ?x2 :lessThan ?n2 }
               FILTER   { !BOUND(?x1) && !BOUND(?x2) }
```

Aufgabe 4

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben folgende Ontologien:





- Wie können folgende Abbildungen gefunden werden:
  - Buch = Bücher
  - Autor = Schriftsteller
  - Sachbuch = Fachbuch
  - ...

Aufgabe 4

TECHNISCHE UNIVERSITÄT DARMSTADT

- Buch = Bücher
  - Edit-Distanz: 0.5 (nicht besonders hoch, vgl. 0.6 für Buch/Fisch)
  - 3-Gramme: 0
- Mögliche Lösung: Stemming
  - Buch → Buch
  - Bücher → Buch

- Autor = Schriftsteller
  - mit String-Methoden nicht findbar
  - mögliche Lösung: Synonymlexikon

Aufgabe 4

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sachbuch = Fachbuch
  - mit String-Methode gut findbar:
    - Edit-Distanz: 0.125
    - 3-Gramme: 83%
  - aber ist das nicht eher ein Glückstreffer?
  - vgl. *Handbuch* und *Handtuch*
    - Edit-Distanz: auch 0.125

Aufgabe 4

TECHNISCHE UNIVERSITÄT DARMSTADT

- Belletristik = Unterhaltung
  - z.B. Synonymlexikon
  - Bounded Path Matching mit
    - Buch = Bücher und
    - Roman = Roman

- Roman = Roman
  - trivial

- von = hatAutor
  - über strukturbasiertes Verfahren aus
    - Buch = Bücher und
    - Autor = Schriftsteller

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben drei Häuser, ein pinkes, ein gelbes, und ein blaues, die nebeneinander stehen und von 1-3 nummeriert sind. In jedem der Häuser wohnt eine Person. Jede der Personen hat ein anderes Hobby (Rugby, Lacrosse oder Tennis) und ein anderes Lieblingsessen (Eier, Waffeln oder Eis). Gegeben sind darüber hinaus noch folgende Fakten:
  - Die Person, die gern Eis isst, lebt im pinken Haus.
  - Der Tennisspieler wohnt links vom Lacrossespieler.
  - Die Person in Haus drei spielt nicht Lacrosse.
  - Die Person, die Tennis spielt, wohnt im gelben Haus.
  - Das blaue Haus steht rechts vom gelben Haus.
  - Der Lacrossespieler wohnt direkt neben der Person, die Eier isst.

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Recap: wir hatten schon definiert
  - disjunkte Klassen für Häuser (linkes, mittleres rechtes, sowie pinkes, gelbes, blaues), Hobbies und Essen
  - Zuordnung zwischen Personen und Häusern, Hobbies, Essen
    - Funktional und invers funktional
    - Jede Person hat genau ein Haus, Hobby, Essen
  - Zuordnung von Nachbarhäusern

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was wir zusätzlich brauchen
  - Häuser stehen links und rechts voneinander

```
:leftOf rdfs:subPropertyOf :neighborOf .
:rightOf rdfs:subPropertyOf :neighborOf .
:leftOf owl:inversePropertyOf :rightOf .
```

  - Jedes Haus kann nur *ein* linkes/rechtes Nachbarhaus haben

```
:leftOf a owl:FunctionalProperty,
          owl:InverseFunctionalProperty .
:rightOf a owl:FunctionalProperty,
           owl:InverseFunctionalProperty .
```

  - Instanzbeziehungen:

```
:linkesHaus :leftOf :mittleresHaus .
:mittleresHaus :leftOf :rechtesHaus.
```

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Formalisierung komplexerer Axiome
  - *Der Tennisspieler wohnt links vom Lacrossespieler.*
  - Hierfür brauchen wir zwei Personen und zwei Häuser:

```
:p1a :livesIn :h1a ;
     :plays :Tennis ;
     owl:differentFrom :p1b .
:p1b :livesIn :h2a ;
     :plays :lacrosse .
:h1a :leftOf :h2a ;
     owl:differentFrom :h2a .
```

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das reicht offenbar noch nicht
  - Der Reasoner findet noch nicht alle Lösungen
  - Wahrscheinlich haben wir noch irgend etwas vergessen
    - d.h.: wir nehmen etwas implizit an, was wir nicht explizit gemacht haben
    - Ideen?

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen
  - Jemand wohnt im linken Haus
  - hat einen Nachbarn, der Lacrosse spielt
  - und einen Nachbarn, der im blauen Haus wohnt
- → der Lacrosse-Spieler wohnt im blauen Haus
- Warum?
  - weil das linke und das rechte Haus nur je ein Nachbarhaus haben

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das brauchen wir noch!

```
House owl:disjointUnionOf (:HouseWithOneNeighbor
                           :HouseWithTwoNeighbors)
:HouseWithOneNeighbor rdfs:subClassOf
  :House ,
  [ a owl:Restriction ;
    owl:onProperty :hasNeighbor ;
    owl:cardinality 1^^xsd:integer ] .
:HouseWithTwoNeighbors rdfs:subClassOf
  :House ,
  [ a owl:Restriction ;
    owl:onProperty :hasNeighbor ;
    owl:cardinality 2^^xsd:integer ] .
```

Aufgabe 5

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ob Du wirklich richtig stehst,
  siehst Du, wenn der Reasoner angeht...

Vorlesung Semantic Web

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering