

Subjekt

URI (oder leere Knoten)

Prädikat

URI

Objekt

URI oder Literal (oder leere Knoten)

RDFS - Folgerungen (Regeln)

BNodes:

$$\frac{u \ a \ x \ .}{u \ a \ _n \ .} \text{ se1}$$

$$\frac{_n \ a \ x \ .}{u \ a \ x \ .} \text{ se2}$$

$$\frac{u \ a \ l^{\wedge d} \ .}{d \ \text{rdf:type} \ \text{rdfs:Datatype} \ .} \text{ rdfs1 Datentypen}$$

$$\frac{a \ \text{rdfs:domain} \ x \ . \quad u \ a \ y \ .}{u \ \text{rdf:type} \ x \ .} \text{ rdfs2} \quad \text{Definitionsbereiche von Properties}$$

$$\frac{a \ \text{rdfs:range} \ x \ . \quad u \ a \ v \ .}{v \ \text{rdf:type} \ x \ .} \text{ rdfs3} \quad \text{Wertebereiche von Properties}$$

$$\frac{u \ a \ x \ .}{u \ \text{rdf:type} \ \text{rdfs:Resource} \ .} \text{ rdfs4a}$$

Das Subjekt jedes Tripels ist eine Ressource

$$\frac{u \ a \ v \ .}{v \ \text{rdf:type} \ \text{rdfs:Resource} \ .} \text{ rdfs4b}$$

Ein Objekt, das kein Literal ist, ist eine Ressource

$$\frac{u \ \text{rdfs:subPropertyOf} \ v \ . \quad v \ \text{rdfs:subPropertyOf} \ x \ .}{u \ \text{rdfs:subPropertyOf} \ x \ .} \text{ rdfs5} \quad \text{Transitivität}$$

$$\frac{u \ \text{rdf:type} \ \text{rdf:Property} \ . \quad u \ \text{rdfs:subPropertyOf} \ u \ .}{u \ \text{rdfs:subPropertyOf} \ u \ .} \text{ rdfs6} \quad \text{Reflexivität}$$

$$\frac{a \ \text{rdfs:subPropertyOf} \ b \ . \quad u \ a \ y \ .}{u \ b \ y \ .} \text{ rdfs7} \quad \text{Unterproperties und Instanzen}$$

$$\frac{u \ \text{rdf:type} \ \text{rdfs:Class} \ . \quad u \ \text{rdfs:subClassOf} \ \text{rdfs:Resource} \ .}{u \ \text{rdfs:subClassOf} \ \text{rdfs:Resource} \ .} \text{ rdfs8} \quad \text{Klassen enthalten nur Ressourcen}$$

$$\frac{u \ \text{rdfs:subClassOf} \ x \ . \quad v \ \text{rdf:type} \ u \ .}{v \ \text{rdf:type} \ x \ .} \text{ rdfs9} \quad \text{Unterklassen und Instanzen}$$

$$\frac{u \ \text{rdf:type} \ \text{rdfs:Class} \ . \quad u \ \text{rdfs:subClassOf} \ u \ .}{u \ \text{rdfs:subClassOf} \ u \ .} \text{ rdfs10} \quad \text{Reflexivität}$$

$$\frac{u \ \text{rdfs:subClassOf} \ v \ . \quad v \ \text{rdfs:subClassOf} \ x \ .}{u \ \text{rdfs:subClassOf} \ x \ .} \text{ rdfs11} \quad \text{Transitivität}$$

```

u rdf:type rdfs:ContainerMembershipProperty .
u rdfs:subPropertyOf rdfs:member .
u rdf:type rdfs:Datatype .
u rdfs:subClassOf rdfs:Literal .

```

CMPs
und rdfs:member

Datentypen
und rdfs:Literal

RDF-Tripel

$$\left(\begin{array}{ccc} \text{type} & \text{entspricht} & \in \\ \text{subClassOf} & \text{entspricht} & \subseteq \end{array} \right)$$

Subjekt	Prädikat	Objekt
domain	(property)	range

Notiz: rdf:type kann durch a abgekürzt werden

SPARQL

Grundstruktur
PREFIX
SELECT Ausgabe
WHERE Suchkonfig.

Graph-Muster
Basic Graph Patterns
{...}
OPTIONAL muss nicht vorhanden
UNION entweder oder

```

PREFIX ex: <http://example.org/>
SELECT ?buch WHERE
{ ?buch ex:verlegtBei <http://springer.com> .
  ?buch ex:preis ?preis
  FILTER (?preis < 35)
}

```

```

PREFIX ex: <http://example.org/>
SELECT ?buch ?text WHERE
{ ?buch ex:kritik ?text .
  FILTER ( langMATCHES( LANG(?text), "de") ) }

```

Filter
BOUND
isURI
isBLANK
isLITERAL
sameTERM
langMATCHES
REGEX

Modifikatoren
ORDER BY
LIMIT
OFFSET
DISTINCT

COUNT Zählen der Lösungen

MIN Finden des Minimalwerts

MAX Finden des Maximalwerts

SUM Summieren der Werte

AVG Bilden des Durchschnitts

GROUP_CONCAT Stringkonkatenation

Z.B.: GROUP_CONCAT(?x ; separator=", ")

SAMPLE Wählen eines beliebigen Wertes

Gegeben sei der folgende Graph:

```

ex:Paul ex:note 2.0 .
ex:Paul ex:note 3.0 .
ex:Mary ex:note 2.0 .
ex:Peter ex:note 3.5 .

```

Welche Antwort erzeugt die folgende Anfrage?

```

SELECT ?student (AVG(?note) AS ?avg)
WHERE { ?student ex:note ?note }
GROUP BY ?student
HAVING (?avg > 2.0)

```

ex:Paul 2.5
ex:Peter 3.5

Klammersetzung relevant!

↳ Beispiel 3.13

```

{ {s1 p1 o1} OPTIONAL {s2 p2 o2} UNION {s3 p3 o3} OPTIONAL {s4 p4 o4}
  OPTIONAL {s5 p5 o5}
}

```

bedeutet:

Beispiel 3.14 (Äquivalentes Muster mit expliziter Klammerung)

```

{ { { { {s1 p1 o1} OPTIONAL {s2 p2 o2}
      } UNION {s3 p3 o3}
      } OPTIONAL {s4 p4 o4}
      } OPTIONAL {s5 p5 o5}
}

```

LIMIT maximale Anzahl von Ergebnissen (Tabellenzeilen)

~~ Ende abschneiden

OFFSET Position des ersten gelieferten Ergebnisses

~~ Anfang abschneiden

DISTINCT Entfernung von doppelten Tabellenzeilen

Beispiel 3.19

```
SELECT DISTINCT ?buch ?preis  
WHERE { ?buch ex:preis ?preis . }  
ORDER BY ?preis LIMIT 5 OFFSET 25
```

~~ Ausgabe: Ergebnisse 26 – 30, Duplikate entfernt

LIMIT und OFFSET meist nur mit ORDER BY sinnvoll.

$$\text{NNF}(A \sqsubseteq B) = \neg A \cup B \quad \text{und} \quad \text{NNF}(A \equiv B) = (A \sqsubseteq B) \cap (B \sqsubseteq A) = (\neg A \cup B) \cap (\neg B \cup A)$$

Tableau

$A \setminus B$ wird zu $A, \neg B$

$A \cup B$ wird zu A oder B

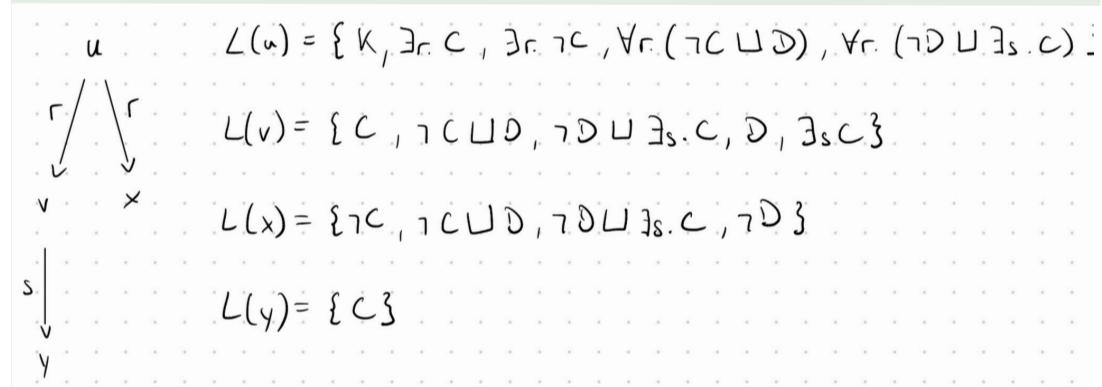
$\exists r. X$ gibt neue Ast

$\forall r. X$ wird auf alle (neuen) Äste übertragen

Übung 4.10

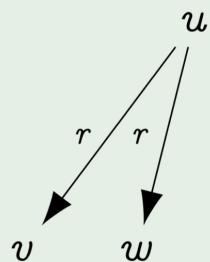
Testen Sie die Konsistenz des Konzepts K mit Hilfe des ALC-Tableau-Algorithmus:

$$K = \exists r. C \sqcap \exists r. \neg C \sqcap \forall r. (\neg C \sqcup D) \sqcap \forall r. (\neg D \sqcup \exists s. C)$$



ALCQ

Beispiel 4.17 (Tableau für $\exists r. C \sqcap \exists r. \neg C \sqcap \geq 2r. D \sqcap \leq 2r. \top$)



$$L(u) = \{\exists r. C \sqcap \exists r. \neg C \sqcap \geq 2r. D \sqcap \leq 2r. \top, \exists r. C, \exists r. \neg C, \geq 2r. D, \leq 2r. \top\}$$

$$L(v) = \{C, D, \neq w\}$$

$$L(w) = \{\neg C, D, \neq v\}$$

Negations-Normalform

$$\blacksquare \neg(\leq n r. C) \rightsquigarrow \geq (n+1) r. C$$

$$\blacksquare \neg(\geq n r. C) \rightsquigarrow \leq (n-1) r. C$$

$$\blacksquare \text{„Anna hat nicht 3 oder mehr Katzen.“} \rightsquigarrow \text{„Anna hat 2 oder weniger Katzen.“}$$

■ Variablen können aus anderen Variablen berechnet werden

■ Arithmetische Operatoren wie bei Filtern

■ Syntax: BIND (<ausdruck> AS ?var)

Beispiel 3.20

Graph: ex:Buch1 ex:title "SPARQL Tutorial"; ex:preis 50 ; # in Euro ex:rabatt 10 . # in Prozent

Anfrage: SELECT ?titel ?endpreis WHERE { ?buch ex:title ?titel; ex:preis ?preis ; ex:rabatt ?rabatt BIND (?preis * (1 - ?rabatt / 100) AS ?endpreis) }

Ergebnis:

titel	endpreis
"SPARQL Tutorial"	45

- \mathcal{ALCQ} erlaubt $\geq n r.C$ und $\leq n r.C$
- schränkt Anzahl der r -Nachfolger ein
- Modifikationen des Tableau-Algorithmus
 - \geq -Regel erzeugt neue Nachfolger mit Ungleichheits-Assertionen $\neq x$
 - \leq -Regel verschmilzt Knoten
 - choose-Regel für $\leq n r.C$ erzwingt, dass alle r -Nachfolger mit C oder $\neg C$ beschriftet sind
 - Clash-Bedingung für Knoten, die mit $\leq n r.C$ beschriftet sind, aber mehr als n ungleiche r -Nachfolger haben, die mit C beschriftet sind
- \leq -Restriktionen können zu Ineffizienz führen

Wissensbanken

TBox \mathcal{T}
Klassen

Informationen über Konzepte und ihre taxonomischen Abhängigkeiten

ABox \mathcal{A}
Individuen

Informationen über Individuen, ihre Konzepte und Rollenverbindungen

RBox \mathcal{R}
Rollen

Informationen über Rollen und ihre Abhängigkeiten

TBox

Übung 4.22

Formalisieren Sie die folgenden Aussagen als GCIs in \mathcal{ALC} :

- 1 Jeder Junge und jedes Mädchen ist ein Kind.
- 2 Jedes Kind bekommt ein Auto, eine Puppe oder eine Rute.
- 3 Kein Junge bekommt eine Puppe.
- 4 Kein braves Kind bekommt eine Rute.
- 5 Kein Kind bekommt ein Auto.

1. Junge \sqsubseteq Maedche, \sqsubseteq Kind
2. Kind \sqsubseteq Eb. Auto \sqcup Eb. Puppe \sqcup Eb. Rute
3. Junge $\sqsubseteq \neg$ Eb. Puppe
4. Brav \sqcap Kind $\sqsubseteq \neg$ Eb. Rute
5. Kind $\sqsubseteq \neg$ Eb. Auto

Verwenden Sie hierzu

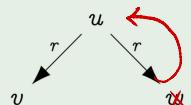
- den Rollennamen bekommt und
- die Konzeptnamen Junge, Maedchen, Kind, Auto, Puppe, Rute und Brav.

Blockierung

Ein Knoten y ist blockiert, wenn

- Ein älterer Knoten x existiert,
- dessen Label alle Konzepte für y schon enthält.

Beispiel 4.26



$K = A \sqcap \exists r.B$	$\tau = \{A \sqsubseteq \exists r.A\}$
$L(u) = \{A \sqcap \exists r.B, A, \exists r.B, \neg A \sqcup \exists r.A, \exists r.A\}$	
$L(v) = \{B, \neg A \sqcup \exists r.A, \neg A\}$	
$L(w) = \{A, \neg A \sqcup \exists r.A, \exists r.A\}$	

Lösung: Einfach blockierenden Knoten löschen

- w ist durch u blockiert
- x und seine Nachfolger werden nicht erzeugt

TBox Tableau

Konzept R wird mit TBox T zum ersten Knoten

ABox

TBox T

$$\begin{array}{l} \text{Gesund} \sqsubseteq \neg \text{Tot} \\ \text{Katze} \sqsubseteq \text{Tot} \sqcup \text{Lebendig} \end{array}$$

„Gesunde sind nicht tot.“
 „Jede Katze ist tot oder lebendig.“
 „Ein glücklicher Katzenbesitzer hat eine Katze und alles, was er hat, ist gesund.“

$$\text{GlücklicherKatzenbesitzer} \sqsubseteq \exists \text{hat.Katze} \sqcap \forall \text{hat.Gesund}$$

ABox \mathcal{A}

$$\begin{array}{ll} (\text{Schrödinger}, \text{Kitty}) : \text{hat} & \text{„Schrödinger hat Kitty.“} \\ \text{Schrödinger} : \text{GlücklicherKatzenbesitzer} & \text{„Schrödinger ist ein glücklicher Katzenbesitzer.“} \end{array}$$

ABox Tableau

Vorverarbeitung: Transformiere Konzepte in Konzept-Fakten in NNF

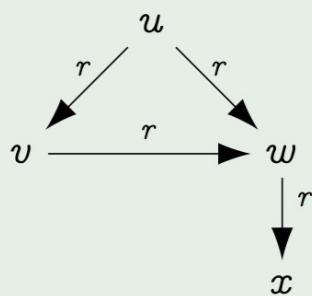
Algorithmus:

- erzeuge für jede Individuenamen a einenen Knoten; beschrifte diesen mit a
- Für $a : C \in \mathcal{A}$: beschrifte Knoten für a mit C
- Für $(a, b) : r \in \mathcal{A}$: erzeuge r -Kante vom Knoten für a zum Knoten für b
- Wende Tableau-Regeln an

Beispiel 4.34

$$T = \{\exists r. T \sqsubseteq \forall r. \exists r. T\}$$

$$\mathcal{A} = \{(a, b) : r, (b, c) : r, (a, c) : r\}$$



$$\begin{aligned} L(u) &= \{a, \forall r. \perp \sqcup \forall r. \exists r. T, \forall r. \exists r. T\} \\ L(v) &= \{b, \forall r. \perp \sqcup \forall r. \exists r. T, \exists r. T, \forall r. \exists r. T\} \\ L(w) &= \{c, \forall r. \perp \sqcup \forall r. \exists r. T, \exists r. T, \forall r. \exists r. T\} \\ L(x) &= \{\forall r. \perp \sqcup \forall r. \exists r. T, \exists r. T, \forall r. \exists r. T\} \end{aligned}$$

x ist durch v blockiert \rightsquigarrow Algorithmus terminiert

WICHTIG: T muss hier auch in alle Folgeknoten eingesetzt werden!

→ siehe Beispiel (Auch bei TBox!?)

Gegeben sei die folgende Signatur:

Konzeptnamen: $\{C, D\}$

Rollennamen: $\{r\}$

Individuennamen: $\{a, b\}$

Testen Sie die Konsistenz der Abox

$$\mathcal{A} = \{a : C, b : D, (a, b) : r, (b, a) : r\}$$

bzgl. der TBox

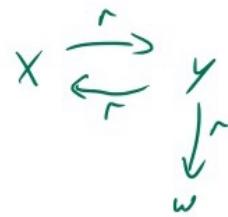
$$\mathcal{T} = \{C \sqsubseteq \exists r.D, D \sqsubseteq \forall r.\forall r.C\}$$

mit Hilfe des in der Vorlesung gezeigten Tableaualgorithmus.

Tip: Wenden Sie die \sqcup -Regel so spät wie möglich an, und beginnen Sie dann mit den Disjunktionen, bei denen nur ein Disjunkt nicht zu einem sofortigen Clash führt.

Ist \mathcal{A} erfüllbar bzgl. \mathcal{T} ?

$$\begin{aligned} L(x) &= \{a, C, \neg C \sqcup \exists r.D, \neg D \sqcup \forall r.\forall r.C, \\ &\quad \forall r.C, \exists r.D, \neg D\} \\ L(y) &= \{b, D, \neg C \sqcup \exists r.D, \neg D \sqcup \forall r.\forall r.C, \\ &\quad \forall r.\forall r.C, C, \exists r.D\} \\ L(w) &= \{D, \neg C \sqcup \exists r.D, \neg D \sqcup \forall r.\forall r.C, \\ &\quad \neg C, \forall r.\forall r.C, \forall r.C\} \end{aligned}$$



↳ Blockiert nicht wegen diesem Allquantor
→ terminiert

Normaler Tableau

Übung 4.10

Testen Sie die Konsistenz des Konzepts K mit Hilfe des \mathcal{ALC} -Tableau-Algorithmus:

$$K = \exists r.C \sqcap \exists r.\neg C \sqcap \forall r.(\neg C \sqcup D) \sqcap \forall r.(\neg D \sqcup \exists s.C)$$

$$L(u) = \{K, \exists r.C, \exists r.\neg C, \forall r.(\neg C \sqcup D), \forall r.(\neg D \sqcup \exists s.C)\}$$

$$\begin{array}{c} r \\ \diagup \quad \diagdown \\ v \quad x \end{array} \quad L(v) = \{C, \neg C \sqcup D, \neg D \sqcup \exists s.C, D, \exists s.C\}$$

$$\begin{array}{c} s \\ \downarrow \\ y \end{array} \quad L(x) = \{\neg C, \neg C \sqcup D, \neg D \sqcup \exists s.C, \neg D\}$$

$$L(y) = \{C\}$$

Übungsklausur

Aufgabe 1 (12 Punkte)

Gegeben sei ein RDF-Graph mit den folgenden Tripeln. Präfix-Deklarationen sind hier nicht extra angegeben, aber Sie können annehmen, dass die Präfixe wie am Anfang der Klausur angegeben deklariert sind. Die Zahlen vor den Tripeln dienen nur dazu, dass Sie auf die Tripel Bezug nehmen können.

- (1) ex:Klaus ex:hatFreund ex:Anna .
- (2) ex:hatFreund rdfs:subPropertyOf ex:mag .
- (3) ex:hatFreund rdfs:domain ex:Mensch .
- (4) ex:hatFreund rdfs:range ex:Mensch .
- (5) ex:Mann rdfs:subClassOf ex:Mensch .
- (6) ex:besitzt rdfs:domain ex:Mensch .
- (7) ex:besitzt rdfs:range ex:Objekt .
- (8) ex:Anna ex:besitzt _:bn1 .
- (9) _:bn1 rdf:type ex:Haus .
- (10) ex:mag rdfs:subPropertyOf ex:kennt .

Geben Sie für jedes der folgenden Tripel an, ob das Tripel aus dem oben gegebenen RDF-Graphen unter der RDFS-Semantik folgt oder nicht. Für Tripel, die unter RDFS folgen, geben Sie jeweils an, durch welche Abfolge der Regeln (RDFS, RDF oder einfach) diese Folgerung abgeleitet werden kann und welche Tripel dabei verwendet wurden.

- a) ex:Klaus rdf:type ex:Mann . *folgt nicht* ✓
- b) ex:Anna rdf:type ex:Mensch . *rdfs 3 auf (1)(4)* ✓
rdfs 7 auf (1)(2) ✓
- c) ex:Klaus ex:mag ex:Anna . *rdfs 7 auf (1)(2)* ✓
- d) ex:Anna ex:hatFreund ex:Klaus . *folgt nicht*
- e) _:n rdf:type ex:Objekt . (*_:n ist ein neuer bnode*) ~~rdfs 3 (7)(8)~~ X *rdfs 3 (8)(7) liefert _:bn1 als (11) => SE-Regel*
- f) ex:Haus rdfs:subClassOf ex:Objekt . *folgt nicht* ✓
- g) ex:hatFreund rdfs:subPropertyOf ex:kennt . *rdfs 5 (2)(10)* ✓
- Worin liegt der Fehler?*

■ Klassen

rdfs:Resource Klasse aller Ressourcen (sämtliche Elemente der Domäne)
rdfs:Class Klasse aller Klassen

rdfs:Container Oberklasse von **rdf:Seq**, **rdf:Bag** und **rdf:Alt**

rdfs:ContainerMembershipProperty

Klasse aller Properties, die eine Enthaltsseinsbeziehung darstellen, z. B. Container, Collections

rdfs:Literal Klasse aller Literalwerte (Oberklasse aller Datentypen)

rdfs:Datatype Klasse aller Datentypen

■ Properties

rdfs:subClassOf Unterklassenbeziehung

rdfs:subPropertyOf Unterpropertybeziehung

rdfs:domain Definitionsbereich

rdfs:range Zielbereich

rdfs:member Oberproperty aller ContainerMembershipProperties

rdfs:label, **rdfs:comment**, **rdfs:isDefinedBy**, **rdfs:seeAlso**

Annotationen

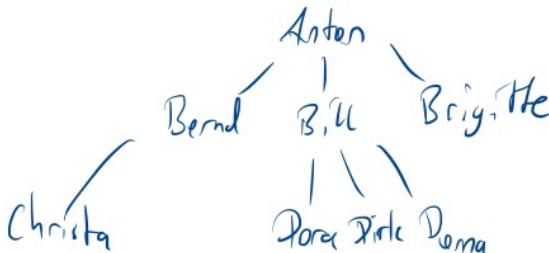
- **rdfs:ContainerMembershipProperty**
Instanzen sind keine Individuen, sondern selbst Properties
- Intendierte Semantik: jede Property, die aussagt, dass das Objekt im Subjekt enthalten ist, ist Instanz von **rdfs:ContainerMembershipProperty**
- Es gilt also insbesondere
`rdf:_1 rdf:type rdfs:ContainerMembershipProperty .`
`rdf:_2 rdf:type rdfs:ContainerMembershipProperty .`
etc.
- **rdfs:member**: „universelle Enthaltsseinsrelation“
Oberproperty aller ContainerMembershipProperties
- **ex:teil** `rdf:type rdfs:ContainerMembershipProperty .`
`ex:Auto ex:teil ex:Motor .`
impliziert
`ex:Auto rdfs:member ex:Motor .`

Aufgabe 2 (3+6 Punkte)

Gegeben sei der folgende RDF-Graph (in Turtle-Syntax):

```

ex:Anton ex:hatKind ex:Bernd .
ex:Anton ex:hatKind ex:Bill .
ex:Anton ex:hatKind ex:Brigitte .
ex:Bernd ex:hatKind ex:Christa .
ex:Bill ex:hatKind ex:Dora .
ex:Bill ex:hatKind ex:Dirk .
ex:Bill ex:hatKind ex:Donna .
  
```



- a) Welche Ausgabe erzeugt die folgende SPARQL-Anfrage? Beschreiben Sie in natürlicher Sprache, in welcher Beziehung die zurückgegebenen Variablenbindungen zueinander stehen.

```

select ?d ?e where
{ ?a ex:hatKind ?b.
  ?a ex:hatKind ?c.
  ?b ex:hatKind ?d.
  ?c ex:hatKind ?e.
  filter (?b != ?c) }
  
```

d	e	→ Cousins/Kusine ✓
Christa	Dora	
Christa	Dirk	
Christa	Donna	
Dora	Christa	
Dirk	Christa	
Donna	Christa	

- b) Schreiben Sie eine SPARQL-Anfrage, durch die alle Individuen ausgegeben werden, die mindestens 2 Geschwister haben. (Wir gehen davon aus, dass unterschiedliche URIs unterschiedliche Menschen bezeichnen.) Beachten Sie: Niemand ist sein eigener Bruder / seine eigene Schwester.

```

select DISTINCT ?a where
{ ?x ex:hatKind ?a.
  ?x ex:hatKind ?b.
  ?x ex:hatKind ?c.
  filter(?a != ?b).
  filter(?b != ?c).
  filter(?a != ?c) } } filter(?a != ?b & & ?b != ?c & & ?c != ?a)
  
```

Übung 4.19

Wenden Sie den Tableau-Algorithmus für \mathcal{ALCQ} auf das folgende Konzept an:

$$K = \exists r. \neg C \sqcap \geq 3r.C \sqcap \geq 2r.D \sqcap \leq 3r.D \sqcap \leq 4r.T$$

Anmerkungen:

- T wird per Definition von jedem Domänelement erfüllt ($T^x = \Delta^x$)
- Jeder Knoten ist implizit mit T beschriftet
- Choose-Regel muss für Konzepte $\leq n r. T$ nicht angewendet werden, da es nur $T \sqcup \perp$ hinzufügen würde

$$L(u) = \{ K, \exists r. \neg C, \geq 3r.C, \geq 2r.D, \leq 3r.D, \leq 4r.T \}$$

$$L(v) = \{ \neg C, D_u \rightarrow D, D, \neq v \}$$

$$L(w) = \{ C, \neq x, \neq y, D_u \rightarrow D, D, \neq v \}$$

$$L(x) = \{ C, \neq w, \neq y, D_u \rightarrow D, \neg D \}$$

$$L(y) = \{ C, \neq w, \neq y, D_u \rightarrow D, \neg D \}$$

$$L(z) = \{ D, \neq z, D_u \rightarrow D \}$$

$$L(o) = \{ D, \neq z, D_u \rightarrow D \}$$

Aufgabe 3 (8 + 12 Punkte)

Gegeben seien die folgenden Aussagen:

1. Jede Katze hat genau vier Beine.
2. Wer (mindestens) ein Bein hat, hat auch (mindestens) einen Fuß.
3. Die Gruppe aller Katzen ist eine Spezies.
4. Alle Katzen sind Raubtiere.
5. Katzen haben keine Schlüsselbeine.
6. Anna mag keine schwarzen Katzen.
7. Anna hat eine kleine Katze.

} TBox

} ABox

- Welche der Aussagen 1–7 lassen sich in RDFS angemessen formalisieren? Geben Sie die entsprechenden Formalisierungen in TTL-Syntax an. Verwenden Sie hierzu die Ressourcen `ex:Raubtier`, `ex:Bein`, `ex:Fuss`, `ex:Spezies`, `ex:Katze`, `ex:Schlüsselbein`, `ex:hat`, `ex:mag`, `ex:Schwarz`, `ex:Klein`.
- Welche der Aussagen 1–7 lassen sich als \mathcal{ALCQ} -Wissensbasis (TBox und ABox) angemessen formalisieren? Geben Sie die entsprechenden TBox-Axiome und ABox-Fakten an. Verwenden Sie hierzu
 - die Konzeptnamen `Raubtier`, `Bein`, `Fuss`, `Spezies`, `Katze`, `Schlüsselbein`, `Schwarz`, `Klein`
 - die Rollennamen `hat`, `mag` und
 - den Individuennamen `Anna`.

Anmerkung: Nicht jede Aussage kann in beiden Formalismen modelliert werden.

- a) 1) geht nicht ✓
- 2) $\exists b1 \ ex:hat \ ex:Bein$ würde implizieren dass auch $\exists b2 \ ex:hat \ ex:Fuss$ hat ✓
 $\exists b1 \ ex:hat \ ex:Fuss$ ein Bein hat \rightarrow also geht nicht.
- 3) $\exists ex:Katze \ rdfs:subClassOf ex:Spezies$. X type weil Gruppe aller Katzen
- 4) $\exists ex:Katze \ rdf:type \ subClassOf ex:Raubtier$. ✓
- 5.) geht nicht \rightarrow rdfs lässt keine Verneinung zu ✓
- 6) geht nicht \rightarrow "
- 7) Anna $ex:hat \ \exists b2$.
 $\exists b2$ ↳ geht nicht. X

$ex:Anna \ ex:hat [rdf:type Klein, Katze]$. \rightarrow das ist ein bröckiger D

- b) 1) Katze $\Sigma \leq 4$ hat. Bein $n \geq 4$ hat. Bein ✓ ✓
- 2) $\exists \text{hat. Bein} \Sigma \exists \text{hat. Fuss}$ ✓
- 3) $\exists ex:Katze \ rdfs:subClassOf ex:Spezies$ geht nicht X
- 4.) $\exists ex:Katze \ rdfs:subClassOf ex:Raubtier$ ✓ ✓
- 5.) $\exists ex:Katze \ rdfs:subClassOf ex:Schlüsselbein$
- 6.) $\exists b3 \Sigma \text{Schwarz} \cap \text{Katze} \ X \rightarrow \text{Anna: } \exists \text{mag } \neg (\text{Katze} \cap \text{Schwarz})$
 $(\text{Anna}, \exists b3): \neg \text{mag}$
- 7.) $\exists b4 \Sigma \text{Katze} \cap \text{Klein} \ X \rightarrow \text{Anna: } \exists \text{hat} (\text{Katze} \cap \text{Klein})$
 $(\text{Anna}, \exists b4): \text{hat}$

$\exists ex:Anna \ ex:hat \ \exists b1$.
 $\exists b1 \ rdfs:subClassOf ex:Klein$.
 $\exists b1 \ ex:Katze$.

Filter

BOUND(A)	true falls A eine gebundene Variable ist (Variable hat einen Wert)
isURI(A)	true falls A eine URI ist
isBLANK(A)	true falls A ein leerer Knoten ist (ein BNode ist)
isLITERAL(A)	true falls A ein Literal ist (ohne Predikat)
sameTERM(A,B)	true falls A und B dieselben Terme sind
langMATCHES(A,B)	true falls die Sprachangabe A auf das Muster B passt
REGEX(A,B)	true falls die Zeichenkette A auf den regulären Ausdruck B passt
STR(A)	lexikalische Darstellung (xsd:string) von URIs oder Literalen
LANG(A)	Sprachcode eines Literalen (rdf:langString) (leerer String falls kein Sprachcode)
DATATYPE(A)	Datentyp-URI eines Literalen (xsd:string bei untypisierten Literalen)

- Entfernt Variablenbindungen aus Ergebnis
- Syntax: <Muster1> MINUS { <Muster2> }
 - 1 Berechne Antwort für <Muster1>
 - 2 Entferne Variablenbindungen, die <Muster2> erfüllen

Beispiel 3.25

Minus

"hat nicht"

Graph:
ex:Peter rdf:type foaf:Person .
ex:Peter foaf:name "Peter" .
ex:Mary rdf:type foaf:Person .

Anfrage:
SELECT ?x
WHERE { ?x rdf:type foaf:Person .
MINUS { ?x foaf:name ?name } }

Ergebnis:

x
ex:Mary

PropertyPaths

Regulärer Ausdrücke über Prädikaten:

- Verkettung von Pfaden: ?s ex:p1 / ex:p2 ?o
- Alternative Pfade: ?s (ex:p1 | ex:p2) ?o
- Pfade mit unbestimmter Länge: ?s ex:p+ ?o, ?s ex:p* ?o

Weitere Konstruktoren:

- Pfade bestimmter Länge: ?s ex:p{2, 4} ?o
- Negation von Pfaden: ?s !ex:p ?o
- Inverse Pfade: ?s ^ex:p ?o (entspricht ?o ex:p ?s)

Interne Verarbeitung von Property Paths

- wenn möglich: Übersetzung in bestehende SPARQL-Konstrukte (|, ^, /)
- eigene Verfahren für +, *, !

Beispiel 3.23 (Verkettung von Pfaden)

```
SELECT ?xName WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:name ?xName .
  ?x foaf:knows{2}/foaf:name "Bill Gates" .
}
```

Beispiel 3.24 (Transitiver Abschluss)

```
SELECT ?s WHERE {
  ?s rdf:type ?type .
  ?type rdfs:subClassOf* ex:Carnivore .
}
```