Erfüllt

Eine Interpretation I erfüllt ein Tripel (s,p,o),
genau dann wenn s,p,o im Vokabular V vorhanden sind und

$(I(s),I(o))2\ I_{EXT}(I(p))$

Oft und gerne auch geschrieben als:

$(s^I,o^I)2\ I_{EXT}(p^I)$

Ein Graph ist wahr in Bezug auf I, falls alle seine Tripel von I erfüllt werden, d.h.

G^I ist wahr, falls T^I wahr für alle T 2 G.

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xml:base="http://example.org/"
         xmlns:ex="http://a.b.org/">

<rdf:Description rdf:about="ethik">
    <ex:gelesenVon>
        <rdf:Description
        rdf:about="http://athen.org/sokrates/#me" />
    </ex:gelesenVon>

    <ex:gehoertVon
    rdf:resource ="http://eressos.org/theophrastos/#me"/>
</rdf:Description>
</rdf:RDF>
```

I_S = {http://example.org/ethik ↦ α,
ex:gelesenVon ↦ β,
http://athen.org/sokrates/#me ↦ γ,
ex:gehoertVon ↦ δ,
http://eressos.org/theophrastos/#me ↦ ε}

I_L=;

Beispiel

I_S = {http://example.org/ethik ↦ α,
ex:gelesenVon ↦ β,
http://athen.org/sokrates/#me ↦ γ,
ex:gehoertVon ↦ δ,
http://eressos.org/theophrastos/#me ↦ ε}

I_L=;

$I_{EXT}(\beta)=\{(\alpha, \gamma), (\gamma, \varepsilon)\}$
$I_{EXT}(\delta)=\{(\alpha, \varepsilon), (\varepsilon, \varepsilon)\}$
$I_{EXT}(\eta)=\{(\alpha, \varepsilon), (\varepsilon, \varepsilon)\}$

IR={α,β, γ, δ, ε, ϕ, η}
IP={β,δ,η}
LV=;

„sinnlos“, aber mathematisch korrekt und nicht „störend“

Wird für das Beispiel benötigt, damit I den Graphen wahr macht

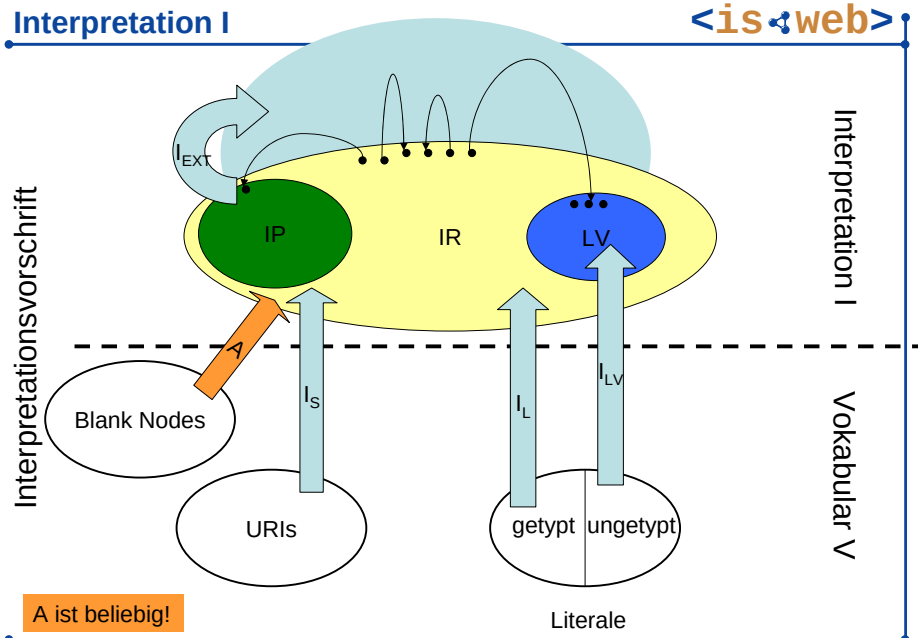
Blank Nodes

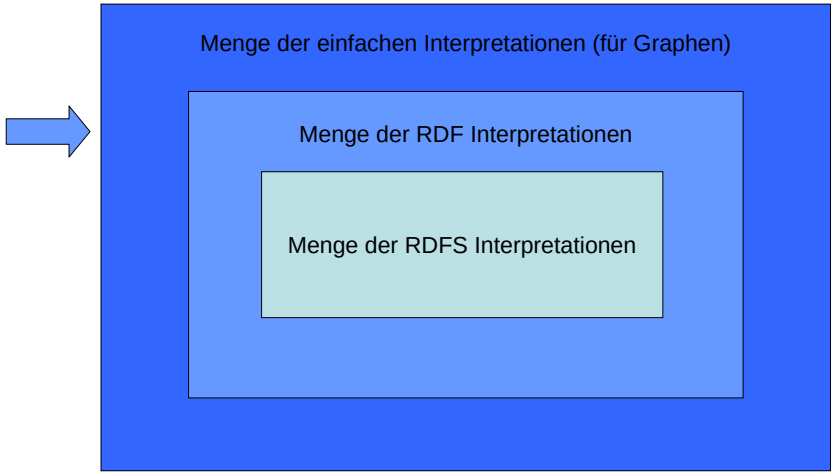
- Entsprechen existentiell quantifizierten Variablen in Logik
- Ein Graph G mit Blank Nodes B ist wahr für I, falls es so eine Abbildung A der Blank Nodes B nach IR gibt,

  A: B ! IR,

  dass G^{I+A} wahr ist.

Interpretation I





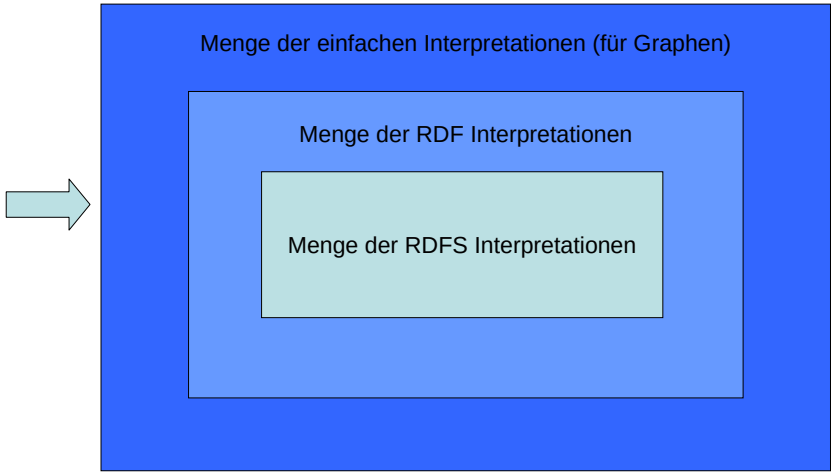
RDF Interpretationen

- Wörtliche Bedeutung
- ex:Ethik ex:gelesenVon ex:Kant
- ex:Kant rdf:type ex:Professor
- Implizite Bedeutung der RDF Modellierungsprimitive
- ex:gelesenVon rdf:type rdf:Property
- rdf:type rdf:type rdf:Property

RDF Interpretation

- Einfache Interpretation &
  - X 2 IP , (X,rdf:Property^I)2 I_{EXT}(rdf:type^I)
  - [..hier einige Spezialregeln für wohlgeformte und nicht-wohlgeformte Literale…vgl. Buch]
  - Folgende Tripel sind wahr:
    - rdf:type rdf:type rdf:Property.
    - rdf:subject rdf:type rdf:Property.
    - rdf:predicate rdf:type rdf:Property.
    - …
    - rdf:_1 rdf:type rdf:Property.
    - rdf:_2 rdf:type rdf:Property.
    - …
    - rdf:nil rdf:type rdf:List.

- Ein Graph G_2 RDF-folgt aus einem Graph G_1, falls jede RDF-Interpretation von G_1 auch eine RDF-Interpretation von G_2 ist.



RDFS Interpretationen

- Wörtliche Bedeutung
- ex:Kant rdf:type ex:Professor

- Implizite Bedeutung der RDFS Modellierungsprimitive
- ex:Professor rdf:type rdfs:Class
- ex:Kant rdf:type ex:Angestellter
- ex:Kant rdf:type rdfs:Resource

RDFS Vokabular

- Rdfs:domain rdfs:range rdfs:Resource rdfs:Literal rdfs:Datatype rdfs:Class rdfs:subClassOf rdfs:subPropertyOf rdfs:member rdfs:Container rdfs:ContainerMembershipProperty rdfs:comment rdfs:seeAlso rdfs:isDefinedBy rdfs:label

- Neue Funktion: I_{CEXT}: IR ! 2^{IR}
  - Bildet Klassen ab auf die Mengen ihrer Objekte
  - $I_{CEXT}(C)$ ist die „Extension“ von C
  - $IC=I_{CEXT}(rdfs:Class^I)$
  - I_{CEXT} bereits durch I_{EXT} und I definiert!

RDFS-Interpretation

Grundlegende Mengen

- IR=I_{CEXT}(rdfs:Resource^I)
- LV=I_{CEXT}(rdfs:Literal^I)

Rdfs:domain und rdfs:range

- Wenn (x,y)2 I_{EXT}(rdfs:domain^I) und (u,v)2 I_{EXT}(x), dann u2 I_{CEXT}(y)
- analog für rdfs:range

RDFS-Interpretation

rdfs:subPropertyOf

- I_{EXT}(rdfs:subPropertyOf^I) ist reflexiv und transitiv auf IP
- Wenn (x,y)2 I_{EXT}(rdfs:subPropertyOf^I), dann x,y2 IP und I_{EXT}(x)µ I_{EXT}(y).

rdfs:Class

- Wenn x2 IC, dann (x,rdfs:Resource^I)2 I_{EXT}(rdfs:subClassOf^I).
- Wenn (x,y)2 I_{EXT}(rdfs:subClassOf^I), dann x,y 2 IC und I_{CEXT}(x)µ I_{CEXT}(y).
- I_{EXT}(rdfs:subClassOf^I) ist reflexiv und transitiv auf IC.

RDFS-Interpretation

Sonstige

- Wenn x2 I_{CEXT}(rdfs:ContainerMembershipProperty^I), dann (x,rdfs:member^I)2 I_{EXT}(rdfs:subPropertyOf^I)
- Wenn x2 I_{CEXT}(rdfs:Datatype^I), dann (x,rdfs:Literal^I) 2 I_{EXT}(rdfs:subClassOf^I)

RDFS-Interpretation

Folgende Tripel müssen gelten:

Typisierung von Tripel-Subjekten:

- rdfs:type rdfs:domain rdfs:Resource
- rdfs:domain rdfs:domain rdf:Property
- …

Typisierung von Tripel-Objekten:

- rdf:type rdfs:range rdfs:Class
- …

+ einige weitere

Listenelement-Propertys
+ unendlich viele

RDFS-folgt

- Ein Graph G_2 RDFS-folgt aus einem Graphen G_1, wenn jede RDFS-Interpretation, die Modell von G_1 ist auch Modell von G_2 ist.

Syntaktisches Schlußfolgern

- Modelltheoretische Semantik ! Gut für Modellierung
- Operationale Semantik ! Gut für Operationalisierung
  - Ableitungsregeln

$$\frac{s_1 \ldots\ldots s_n}{s}$$

  - Gesamtheit der Ableitungsregeln: Deduktionskalkül

Konventionen

- a,b sind URIs
- _:n ist die ID eines leeren Knoten
- u,v stehen für URIs oder IDs von leeren Knoten
- l für ein beliebiges Literal
- x,y für beliebige URIs, IDs von leeren Knoten oder Literale

Einfache Folgerung

se1:

u	a	x	.
u	a	_:n	.

se2:

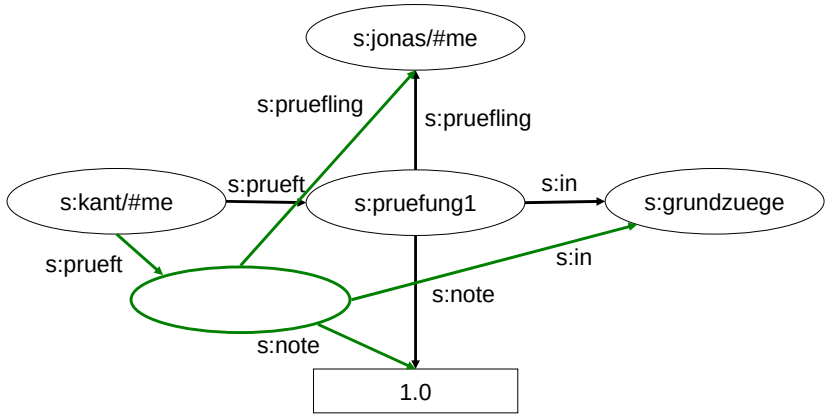
u	a	x	.
_:n	a	x	.

Satz: Ein Graph G_2 folgt einfach aus einem Graphen G_1, wenn G_1 mithilfe der Regeln se1 und se2 zu einem Graphen G_1' ergänzt werden kann, so dass G_2 in G_1' enthalten ist.

Wobei jedes konkrete „_:n“ nur für diesen einen Knoten substitutiert werden darf.

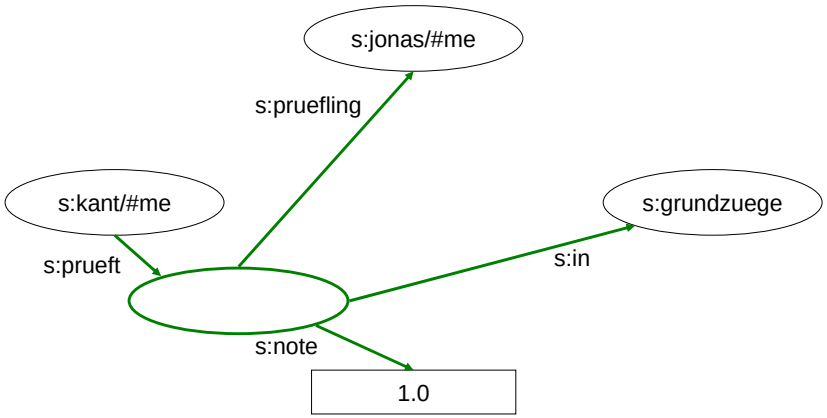
Beispiel von G_1 nach G_1'

„Kant“ prüft „Jonas“ in „Grundzüge“ und gibt die Note „1.0“



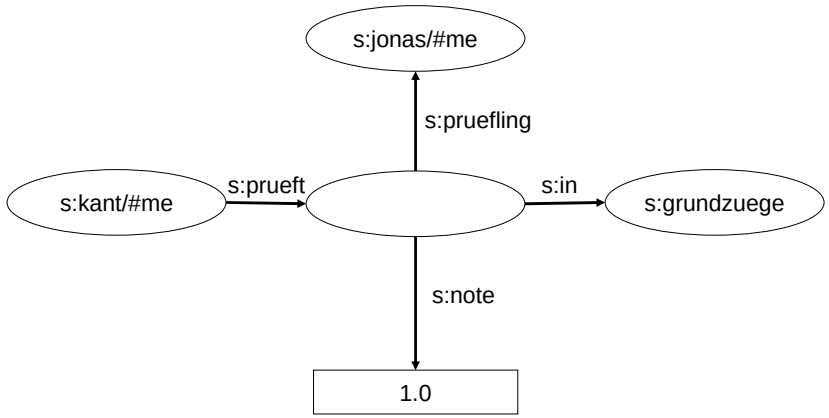
Beispiel G_1'

„Kant“ prüft „Jonas“ in „Grundzüge“ und gibt die Note „1.0“



Beispiel G_2

„Kant“ prüft „Jonas“ in „Grundzüge“ und gibt die Note „1.0“



RDF-Folgerung

Für grundlegende RDF Axiome

rdfax:

u a x .

---

u a x .

lg:

u a l .

u a _:n .

---

rdf1:

u a y .

---

a rdf:type rdf:Property .

(lg+) rdf2:

u a l .

---

u a _:n .

_:n rdf:type rdf:XMLLiteral .

Satz: Ein Graph G_2 RDF-folgt aus einem Graphen G_1 genau dann, wenn es einen Graphen G_1' gibt, welcher aus G_1 mithilfe der Regeln lg, rdf1, rdf2, und rdfax hergeleitet werden kann und aus dem G_2 einfach folgt.

Beachte Reihenfolge!

RDFS-Folgerung

1. Axiomatische Tripel
2. Behandlung von Literalen
3. Auswirkung von Property-Einschränkungen
4. Unterproperties
5. Alles ist eine Ressource
6. Unterklassen
7. Container
8. Literale

RDFS-Folgerung: 1. Axiomatische Tripel

rdfsax:

u a x .

---

u a x .

RDFS-Folgerung: 2. Behandlung von Literalen

gl:

u a _:n .

---

u a l .

falls _:n dem Literal l zuvor zugewiesen wurde.

(lg+) rdfs1:

u a l .

---

u a _:n

_:n rdf:type rdfs:Literal .

RDFS-Folgerung: 3. Property-Einschränkungen

rdfs2:

a rdfs:domain x. u a y .
u rdf:type x .

rdfs3:

a rdfs:range x. u a v .
v rdf:type x .

RDFS-Folgerung: 4. Unterproperties

Rdfs5 (Transitivität):

u rdfs:subPropertyOf v . v rdfs:subPropertyOf x .
u rdfs:subPropertyOf x.

rdfs6 (Reflexivität):

u rdf:type rdf:Property .
u rdfs:subPropertyOf u .

rdfs7 (Anwendung auf Instanzenebene):

a rdfs:subPropertyOf b . u a y .
u b y .

RDFS-Folgerung: 5. Alles ist eine Ressource

rdfs4a (Subjekt ist Resource):

u a x .
u rdf:type rdfs:Resource .

rdfs4b (Objekt ist Resource):

u a v .
v rdf:type rdfs:Resource .

Implizit – Predicate ist Resource:

Rdf1 & rdfs4a:

u a x . rdf1
a rdf:type rdfs:Property . rdfs4a
a rdf:type rdfs:Resource .

RDFS-Folgerung: 6. Unterklassen

rdfs8 (Subklassen von Resource):

u rdf:type rdfs:Class .
u rdfs:subClassOf rdfs:Resource .

rdfs9 (Anwendung):

v rdf:type u . u rdfs:subClassOf x .
v rdf:type x.

rdfs10 (Reflexivität):

u rdf:type rdfs:Class .
u rdfs:subClassOf u .

rdfs11 (Transitivität):

u rdfs:subClassOf v . v rdfs:subClassOf x .
u rdfs:subClassOf x.

RDFS-Folgerung: 7. Container

rdfs12:

u rdf:type rdfs:ContainerMemberShipProperty .

u rdfs:subPropertyOf rdfs:member .

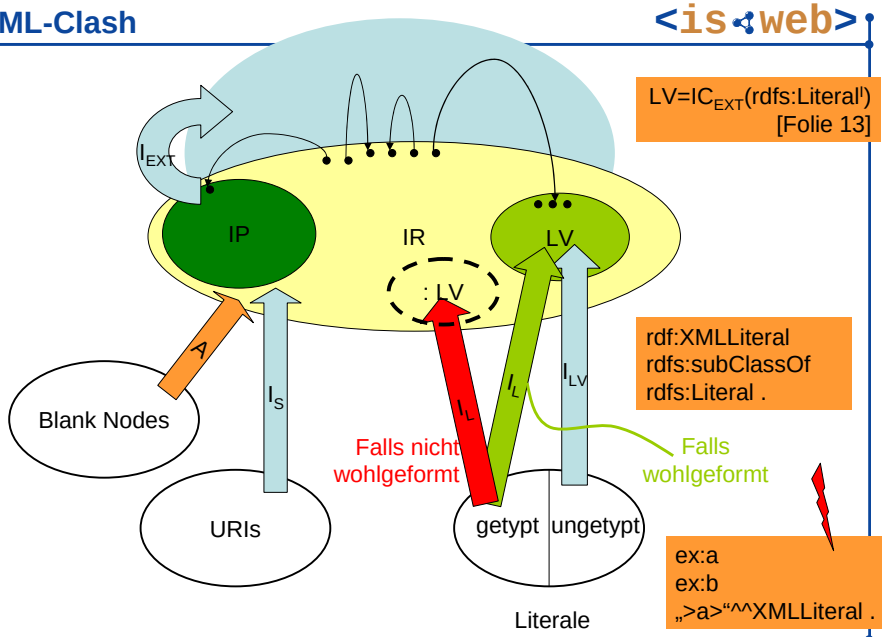
RDFS-Folgerung: 8. Literale/Datatypes

rdfs13:

u rdf:type rdfs:Datatype .

u rdfs:subClassOf rdfs:Literal .

XML-Clash

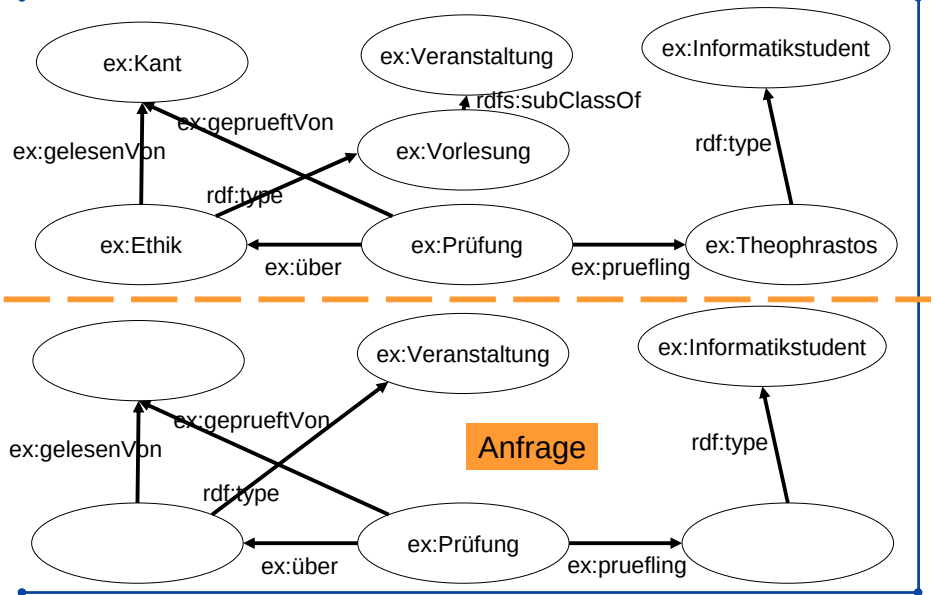


RDFS-Folgerung

Satz: Ein Graph G_2 RDFS-folgt aus G_1 genau dann, wenn es einen Graphen G_1' gibt, der durch Anwendung der Regeln lg, gl, rdfax, rdf1, rdf2, rdfs1-rdfs13 und rdfsax aus G_1 folgt, so dass

- G_2 aus G_1' einfach folgt oder
- G_1' einen XML-Clash enthält

Beispiel



1. rdfs9 auf ex:Vorlesung

2. 6x Blanknodegenerierung
   3x se1, 3x se2

Einschränkungen

Nicht alles, was in PL1 folgen würde, folgt.

Z.B.:

ex:gelesenVon rdfs:domain ex:Vorlesung .

ex:Vorlesung rdfs:subClassOf ex:Veranstaltung .

Folgt nicht:

~~ex:gelesenVon rdfs:domain ex:Veranstaltung .~~

Negation nicht möglich:

Z.B.:

ex:sebastian rdf:type ex:Nichtraucher.

ex:sebastian rdf:type ex:Raucher.

ergibt keinen Widerspruch.

- Welches wichtige Konstrukt fehlt?