



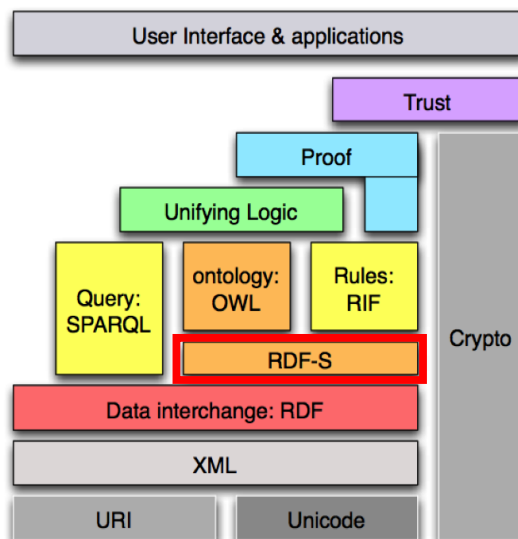
Semantic Web Grundlagen Semantik von RDF(S)

Birte Glimm
Institut für Künstliche Intelligenz | 7. Nov 2011

Organisatorisches: Inhalt

Einleitung und XML	17. Okt	SPARQL Syntax & Intuition	12. Dez
Einführung in RDF	20. Okt	Übung 4	15. Dez
RDF Schema	24. Okt	SPARQL Semantik	19. Dez
fällt aus	27. Okt	SPARQL 1.1	22. Dez
Logik – Grundlagen	31. Okt	Übung 5	9. Jan
Übung 1	3. Nov	SPARQL Entailment	12. Jan
Semantik von RDF(S)	7. Nov	SPARQL Implementierung	16. Jan
RDF(S) & Datalog Regeln	10. Nov	Abfragen & RIF	19. Jan
OWL Syntax & Intuition	14. Nov	Übung 6	23. Jan
Übung 2	17. Nov	Ontology Editing	26. Jan
OWL & BLs	21. Nov	Ontology Engineering	30. Jan
OWL 2	24. Nov	Linked Data	2. Feb
Tableau	28. Nov	Übung 7	6. Feb
Übung 3	1. Dez	SemWeb Anwendungen	9. Feb
Blocking & Unravelling	5. Dez	Wiederholung	13. Feb
Hypertableau	8. Dez	Übung 8	16. Feb

RDF Schema



Agenda

- ▶ Motivation
- ▶ Vorbetrachtungen
- ▶ Einfache Folgerung
- ▶ RDF-Folgerung
- ▶ RDFS-Folgerung
- ▶ Unzulänglichkeiten von RDF(S)

Warum formale Semantik?

- ▶ Nach Einführung von RDFS Kritik von Tool-Herstellern: verschiedene Tools – Inkompatibilitäten (trotz Spezifikation)
- ▶ Z.B. bei triple stores:
 - ▶ Gleiches RDF-Dokument
 - ▶ Gleiche SPARQL-Anfrage
 - ▶ Verschiedene Antworten
- ▶ Daher: modelltheoretische Semantik für RDF(S)

Agenda

- ▶ Motivation
- ▶ **Vorbetrachtungen**
- ▶ Einfache Folgerung
- ▶ RDF-Folgerung
- ▶ RDFS-Folgerung
- ▶ Unzulänglichkeiten von RDF(S)

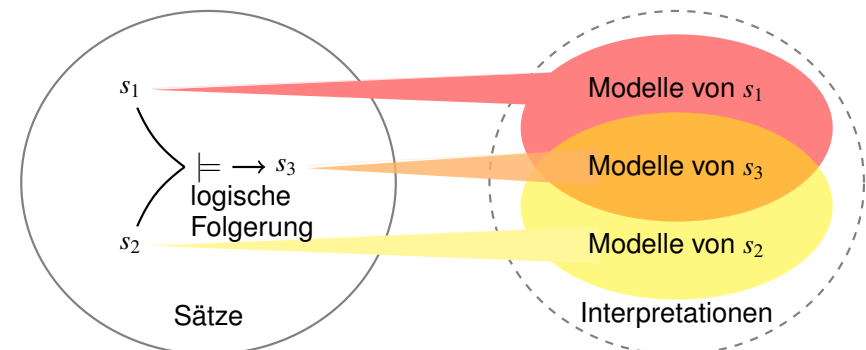
Was ist die Syntax?

- ▶ Also: was sind die Sätze in RDF(S)?
 - ▶ Grundelemente (Vokabular V): **IRIs**, **leere Knoten** und **Literale** (sind selbst keine Sätze)
 - ▶ Jedes Tripel

$$(s, p, o) \in (\text{IRI} \cup \text{bnode}) \times \text{IRI} \times (\text{IRI} \cup \text{bnode} \cup \text{Literal})$$
 ist ein Satz
 - ▶ Jede endliche Menge von Tripeln (genannt Graph) ist ein Satz

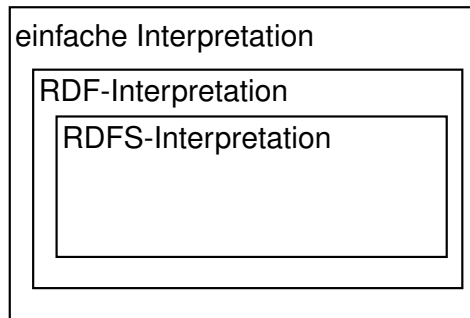
Was ist die Semantik?

- ▶ Konsequenzrelation, die sagt, wann ein RDF(S)-Graph G' aus einem RDF(S)-Graphen G folgt, d.h. $G \models G'$
- ▶ Modelltheoretische Semantik: wir definieren Menge von Interpretationen und legen fest, wann eine Interpretation Modell eines Graphen ist



Was ist die Semantik?

- ▶ Vorgehen schrittweise:



- ▶ Je eingeschränkter die Interpretationen umso stärker die Folgerungsrelation

Semantik der einfachen Folgerung

Definition

Eine **einfache Interpretation** \mathcal{I} für ein Vokabular V besteht aus

- ▶ IR , einer nichtleeren Menge von *Ressourcen*,
- ▶ IP , der Menge der *Property*s von \mathcal{I} ,
- ▶ I_S , einer Funktion, welche IRIs aus V in die Vereinigung der Mengen IR und IP abbildet, also $I_S: V \rightarrow IR \cup IP$,
- ▶ I_{EXT} , einer Funktion, welche jeder Property eine Menge von Paaren aus IR zuordnet, also $I_{EXT}: IP \rightarrow 2^{IR \times IR}$,
- ▶ I_L , einer Funktion von den getypten Literalen aus V in die Menge IR der Ressourcen und
- ▶ LV , einer speziellen Teilmenge von IR , genannt Menge der Literalwerte, die (mindestens) alle ungetypten Literale aus V enthält.

Agenda

- ▶ Motivation
- ▶ Vorbetrachtungen
- ▶ **Einfache Folgerung**
- ▶ RDF-Folgerung
- ▶ RDFS-Folgerung
- ▶ Unzulänglichkeiten von RDF(S)

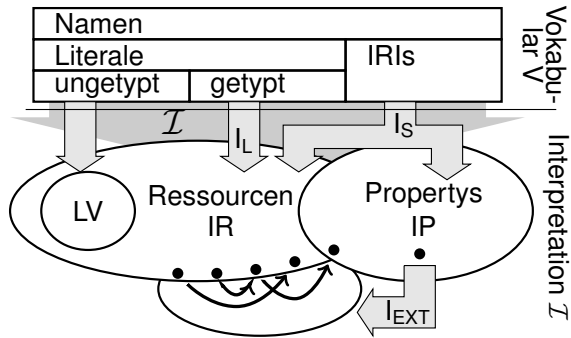
Semantik der einfachen Folgerung

- ▶ IR wird auch Domäne oder (Diskurs-)Universum von \mathcal{I} genannt
- ▶ $I_{EXT}(p)$ wird auch die Extension der Property p genannt

Semantik der einfachen Folgerung

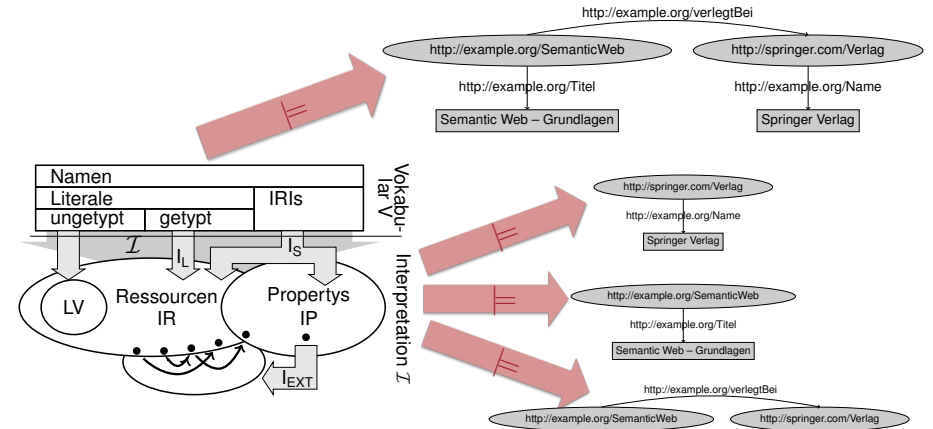
- ▶ Ein ungetyptes Literal "a" wird auf a abgebildet: $(\text{"a"})^I = a$
- ▶ Ein ungetypte Literal mit Sprachangabe "a"@t wird auf das Paar $\langle a, t \rangle$ abgebildet: $(\text{"a"@t})^I = \langle a, t \rangle$,
- ▶ Ein getyptes Literal l wird auf $I_L(l)$ abgebildet: $l^I = I_L(l)$ und
- ▶ Eine IRI i wird auf $I_S(i)$ abgebildet: $i^I = I_S(i)$.

Interpretation (schematisch):



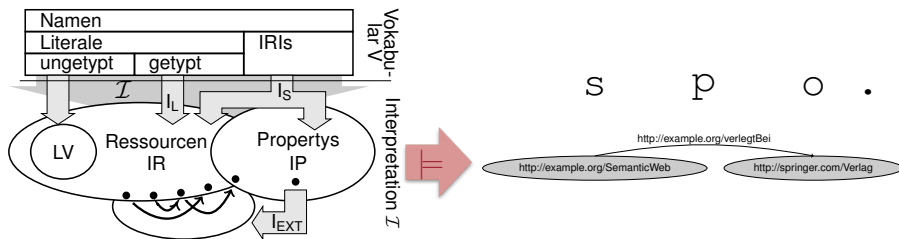
Semantik der einfachen Folgerung

- ▶ Frage: Wann ist eine gegebene Interpretation Modell eines Graphen?
- ▶ ... wenn sie Modell jedes Triples des Graphen ist!



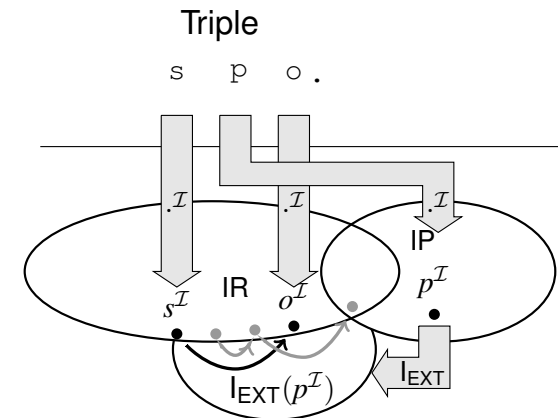
Semantik der einfachen Folgerung

- ▶ Frage: Wann ist eine gegebene Interpretation Modell eines Tripels?
- ▶ ... wenn Subjekt, Prädikat und Objekt in V enthalten sind und außerdem $\langle s^I, o^I \rangle \in I_{EXT}(p^I)$



Semantik der einfachen Folgerung

Schematisch:



Einfache Folgerung

- ▶ Ableitungsregeln für einfache Folgerung:

$$\frac{u \quad a \quad x \quad .}{u \quad a \quad _ :n \quad .} \text{ se1}$$

$$\frac{u \quad a \quad x \quad .}{_ :n \quad a \quad x \quad .} \text{ se2}$$

- ▶ Bedingung für Anwendung: leerer Knoten nicht bereits anderer IRI/anderem Literal zugeordnet

Agenda

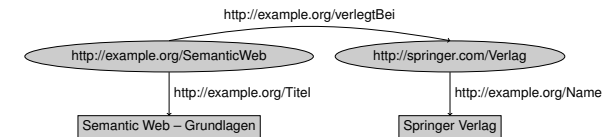
- ▶ Motivation
- ▶ Vorbetrachtungen
- ▶ Einfache Folgerung
- ▶ **RDF-Folgerung**
- ▶ RDFS-Folgerung
- ▶ Unzulänglichkeiten von RDF(S)

Einfache Folgerung

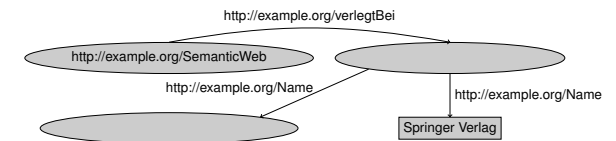
Satz

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.

Bsp.: Aus



folgt einfach



RDF-Interpretationen

- ▶ ... RDF-Interpretationen sind spezielle einfache Interpretationen, wobei für die URIs des RDF-Vokabulars

```

rdf:type rdf:Property rdf:XMLLiteral rdf:nil
rdf:List rdf:Statement rdf:subject rdf:predicate
rdf:object rdf:first rdf:rest rdf:Seq rdf:Bag
rdf:Alt rdf:_1 rdf:_2 ...

```

zusätzliche Forderungen gestellt werden, die die intendierte Semantik der RDF-Bezeichner realisieren:

RDF-Interpretationen

Eine RDF-Interpretation für ein Vokabular V ist eine einfache Interpretation für das Vokabular $V \cup V_{\text{RDF}}$, welche zusätzlich folgende Bedingungen erfüllt:

- ▶ $x \in IP$ genau dann, wenn $\langle x, \text{rdf:Property}^I \rangle \in I_{\text{EXT}}(\text{rdf:type}^I)$.
- ▶ Wenn " s " `rdf:XMLLiteral` in V enthalten und s ein wohlgeformtes XML-Literal ist, dann
 - ▶ $I_L("s" \text{^^} \text{rdf:XMLLiteral})$ ist der XML-Wert von s ;
 - ▶ $I_L("s" \text{^^} \text{rdf:XMLLiteral}) \in LV$;
 - ▶ $\langle I_L("s" \text{^^} \text{rdf:XMLLiteral}), \text{rdf:XMLLiteral}^I \rangle \in I_{\text{EXT}}(\text{rdf:type}^I)$
- ▶ Wenn " s " `rdf:XMLLiteral` in V enthalten und s ein *nicht* wohlgeformtes XML-Literal ist, dann
 - ▶ $I_L("s" \text{^^} \text{rdf:XMLLiteral}) \notin LV$ und
 - ▶ $\langle I_L("s" \text{^^} \text{rdf:XMLLiteral}), \text{rdf:XMLLiteral}^I \rangle \notin I_{\text{EXT}}(\text{rdf:type}^I)$.

RDF-Interpretationen

- ▶ **Zusätzliche Forderung:** jede RDF-Interpretation muss Modell der folgenden, "axiomatischen" Tripel sein:

<code>rdf:type</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:subject</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:predicate</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:object</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:first</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:rest</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:value</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:_1</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:_2</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>...</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:nil</code>	<code>rdf:type</code>	<code>rdf:Property</code> .

RDF-Interpretationen

- ▶ **Merke:** x ist eine Property genau dann, wenn es mit der durch `rdf:Property` bezeichneten Ressource über die `rdf:type`-Property verbunden ist (dies führt auch automatisch dazu, dass für jede RDF-Interpretation $IP \subseteq IR$ gilt).
- ▶ Der Wertebereich des `rdf:XMLLiteral` Datentyps enthält für jede wohlgeformte XML-Zeichenkette genau einen sogenannten XML-Wert. Die RDF-Spezifikation fordert lediglich, dass ein XML-Wert keine XML-Zeichenkette, kein Datenwert irgendeines XML-Schema-Datentyps und auch keine Unicode-Zeichenkette ist. Für den intuitiven Gebrauch kann man sich die XML-Werte einfach als XML-Zeichenketten vorstellen.

RDF-Folgerungen

- ▶ Automatische Folgerungen werden wieder über Ableitungsregeln realisiert:

$\frac{}{u \ a \ x}$	<code>rdfax</code>	Jedes axiomatische Tripel " $u \ a \ x$." kann immer abgeleitet werden
$\frac{u \ a \ l}{u \ a \ _:n}$	<code>lg</code>	Literale dürfen durch nicht anderweitig gebundene <code>bnodes</code> ersetzt werden
$\frac{u \ a \ y}{a \ \text{rdf:type} \ \text{rdf:Property}}$	<code>rdff1</code>	Für jedes Tripelprädikat kann abgeleitet werden, dass es eine Entität aus der Klasse der Properties ist
$\frac{u \ a \ l}{_:n \ \text{rdf:type} \ \text{rdf:XMLLiteral}}$	<code>rdff2</code>	Wenn <code>_:n</code> durch <code>lg</code> dem wohlgeformten XML-Literal / zugewiesen wurde

RDF-Folgerung

- ▶ Satz: Ein Graph G_2 RDF-folgt aus einem Graphen G_1 , wenn es einen Graphen G'_1 gibt, so dass
 - ▶ G'_1 aus G_1 via lg, rdf1, rdf2 und rdfs hergeleitet werden kann und
 - ▶ G_2 aus G'_1 einfach folgt.
- ▶ Beachte: zweistufiger Folgerungsprozess

RDFS-Interpretationen

... RDFS-Interpretationen sind spezielle RDF-Interpretationen, wobei für die URIs des RDFS-Vokabulars

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

zusätzliche Forderungen gestellt werden, die die intendierte Semantik der RDF-Bezeichner realisieren:

Agenda

- ▶ Motivation
- ▶ Vorbetrachtungen
- ▶ Einfache Folgerung
- ▶ RDF-Folgerung
- ▶ **RDFS-Folgerung**
- ▶ Unzulänglichkeiten von RDF(S)

RDFS-Interpretationen

- ▶ Aus Gründen der einfacheren Darstellung führen wir für eine gegebene RDF-Interpretation die Funktion I_{CEXT} ein, die Ressourcen auf Mengen von Ressourcen abbildet (also: $I_{\text{CEXT}}: \text{IR} \rightarrow 2^{\text{IR}}$). Dabei enthalte $I_{\text{CEXT}}(y)$ genau die Elemente x , für die $\langle x, y \rangle$ in $I_{\text{EXT}}(I(\text{rdf:type}))$ enthalten ist. $I_{\text{CEXT}}(y)$ nennt man auch die (*Klassen-Extension*) von y .
- ▶ Weiterhin definieren wir IC als die Extension der speziellen IRI `rdfs:Class`, also: $\text{IC} = I_{\text{CEXT}}(\text{rdfs:Class}^{\mathcal{I}})$.
- ▶ Merke: Sowohl I_{CEXT} als auch IC sind durch $\cdot^{\mathcal{I}}$ sowie I_{EXT} bereits eindeutig festgelegt.

RDFS-Interpretationen

Eine *RDFS-Interpretation* für ein Vokabular V ist eine RDF-Interpretation des Vokabulars $V \cup V_{\text{RDFS}}$, welche zusätzlich die folgenden Kriterien erfüllt:

- ▶ $IR = I_{\text{CEXT}}(\text{rdfs:Resource}^{\mathcal{I}})$
Jede Ressource ist vom Typ `rdfs:Resource`.
- ▶ $LV = I_{\text{CEXT}}(\text{rdfs:Literal}^{\mathcal{I}})$
Jedes ungetypte und jedes wohlgeformte getypte Literal ist vom Typ `rdfs:Literal`.
- ▶ Wenn $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:domain}^{\mathcal{I}})$ und $\langle u, v \rangle \in I_{\text{EXT}}(x)$, dann $u \in I_{\text{CEXT}}(y)$.
Ist x mit y durch die Property `rdfs:domain` verbunden und verbindet die Property x die Ressourcen u und v , dann ist u vom Typ y .

RDFS-Interpretationen

- ▶ Wenn $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:subPropertyOf}^{\mathcal{I}})$, dann $x, y \in IP$ und $I_{\text{EXT}}(x) \subseteq I_{\text{EXT}}(y)$.
Wird x mit y durch `rdfs:subPropertyOf` verbunden, dann sind sowohl x als auch y Property's und jedes in der Extension von x enthaltene Ressourcenpaar ist auch in der Extension von y enthalten.
- ▶ Wenn $x \in IC$, dann $\langle x, \text{rdfs:Resource}^{\mathcal{I}} \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$.
Bezeichnet x eine Klasse, dann muss es eine Unterklasse der Klasse aller Ressourcen sein, d.h., das Paar aus x und `rdfs:Resource` ist in der Extension von `rdfs:subClassOf`.

RDFS-Interpretationen

- ▶ Wenn $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:range}^{\mathcal{I}})$ und $\langle u, v \rangle \in I_{\text{EXT}}(x)$, dann $v \in I_{\text{CEXT}}(y)$.
Ist x mit y durch die Property `rdfs:range` verbunden und verbindet die Property x die Ressourcen u und v , dann ist v vom Typ y .
- ▶ $I_{\text{EXT}}(\text{rdfs:subPropertyOf}^{\mathcal{I}})$ ist reflexiv und transitiv auf IP .
Die `rdfs:subPropertyOf`-Property verbindet jede Property mit sich selbst.
Darüber hinaus gilt: Verbindet `rdfs:subPropertyOf` die Property x mit Property y und außerdem y mit der Property z , so verbindet `rdfs:subPropertyOf` auch x direkt mit z .

RDFS-Interpretationen

- ▶ Wenn $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$, dann $x, y \in IC$ und $I_{\text{CEXT}}(x) \subseteq I_{\text{CEXT}}(y)$.
Stehen x und y in der `rdfs:subClassOf`-Beziehung, sind sowohl x als auch y Klassen und die (Klassen-)Extension von x ist Teilmenge der (Klassen-)Extension von y .
- ▶ $I_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$ ist reflexiv und transitiv auf IC .
Die `rdfs:subClassOf`-Property verbindet jede Klasse mit sich selbst.
Darüber hinaus folgt, wann immer diese Property Klasse x mit Klasse y und Klasse y mit Klasse z verbindet, dass sie x auch direkt mit z verbindet.

RDFS-Interpretationen

- ▶ **Wenn**
 $x \in \mathbb{I}_{\text{CEXT}}(\text{rdfs:ContainerMembershipProperty}^{\mathcal{I}})$,
dann $\langle x, \text{rdfs:member}^{\mathcal{I}} \rangle \in \mathbb{I}_{\text{EXT}}(\text{rdfs:subPropertyOf}^{\mathcal{I}})$.
Ist x eine Property vom Typ
`rdfs:ContainerMembershipProperty`, **so steht sie in**
der `rdfs:subPropertyOf`-Beziehung zur
`rdfs:member`-**Property.**
- ▶ **Wenn $x \in \mathbb{I}_{\text{CEXT}}(\text{rdfs:Datatype}^{\mathcal{I}})$, dann**
 $\langle x, \text{rdfs:Literal}^{\mathcal{I}} \rangle \in \mathbb{I}_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$.
Ist ein x als Element der Klasse `rdfs:Datatype` “getypt”,
dann muss dieses auch eine Unterklasse der Klasse aller
Literalwerte (bezeichnet mit `rdfs:Literal`) sein.
- ▶ ... dazu kommen dann noch jede Menge weitere
 axiomatische Tripel:

RDFS-Interpretationen

<code>rdf:type</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:domain</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:range</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:subPropertyOf</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:subClassOf</code>	<code>rdfs:domain</code>	<code>rdfs:Class</code> .
<code>rdf:subject</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdf:predicate</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdf:object</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdfs:member</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:first</code>	<code>rdfs:domain</code>	<code>rdf:List</code> .
<code>rdf:rest</code>	<code>rdfs:domain</code>	<code>rdf:List</code> .
<code>rdfs:seeAlso</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:isDefinedBy</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:comment</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:label</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:value</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .

RDFS-Interpretationen

<code>rdf:type</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdfs:domain</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdfs:range</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdfs:subPropertyOf</code>	<code>rdfs:range</code>	<code>rdf:Property</code> .
<code>rdfs:subClassOf</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdf:subject</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:predicate</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:object</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdfs:member</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:first</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:rest</code>	<code>rdfs:range</code>	<code>rdf:List</code> .
<code>rdfs:seeAlso</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdfs:isDefinedBy</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdfs:comment</code>	<code>rdfs:range</code>	<code>rdfs:Literal</code> .
<code>rdfs:label</code>	<code>rdfs:range</code>	<code>rdfs:Literal</code> .
<code>rdf:value</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .

RDFS-Interpretationen

<code>rdfs:ContainerMembershipProperty</code>	<code>rdfs:subClassOf</code>	<code>rdf:Property</code> .
<code>rdf:Alt</code>	<code>rdfs:subClassOf</code>	<code>rdfs:Container</code> .
<code>rdf:Bag</code>	<code>rdfs:subClassOf</code>	<code>rdfs:Container</code> .
<code>rdf:Seq</code>	<code>rdfs:subClassOf</code>	<code>rdfs:Container</code> .
<code>rdfs:isDefinedBy</code>	<code>rdfs:subPropertyOf</code>	<code>rdfs:seeAlso</code> .
<code>rdf:XMLLiteral</code>	<code>rdf:type</code>	<code>rdfs:Datatype</code> .
<code>rdf:XMLLiteral</code>	<code>rdfs:subClassOf</code>	<code>rdfs:Literal</code> .
<code>rdfs:Datatype</code>	<code>rdfs:subClassOf</code>	<code>rdfs:Class</code> .
<code>rdf:_1</code>	<code>rdf:type</code>	<code>rdfs:ContainerMembershipProperty</code> .
<code>rdf:_1</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:_1</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:_2</code>	<code>rdf:type</code>	<code>rdfs:ContainerMembershipProperty</code> .
...		

RDFS-Folgerung

Automatische Folgerungen werden wieder über Ableitungsregeln realisiert:

$\frac{}{u \text{ a } x .}$	rdfsax	Jedes axiomatische Tripel "u a x ." kann immer abgeleitet werden	
$\frac{u \text{ a } _ : n .}{u \text{ a } l .}$	gl	Die Umkehrung von Regel lg: $_ : n$ wurde von der Regel lg für das ungetypte Literal l vergeben	
$\frac{u \text{ a } l .}{_ : n \text{ rdf:type rdfs:Literal}}$	rdfs1	$_ : n$ wurde von der Regel lg für das ungetypte Literal l vergeben	
$\frac{a \text{ rdfs:domain } x . u \text{ a } y .}{u \text{ rdf:type } x .}$	rdfs2	Implementiert die Semantik von Domänen für Propertys	
$\frac{a \text{ rdfs:range } x . u \text{ a } v .}{v \text{ rdf:type } x .}$	rdfs3	Implementiert die Semantik von Ranges für Propertys	
a, b	IRIs	x, y	IRI, blank node oder Literal
u, v	IRI oder blank node	l	Literal
		$_ : n$	blank nodes

RDFS-Folgerung

$\frac{u \text{ a } x .}{u \text{ rdf:type rdfs:Resource .}}$	rdfs4a	Das Subjekt jedes Triples ist eine Ressource
$\frac{u \text{ a } v .}{v \text{ rdf:type rdfs:Resource .}}$	rdfs4b	Objekte, die keine Literale sind, sind ebenfalls eine Ressourcen
$\frac{u \text{ rdfs:subPropertyOf } v . v \text{ rdfs:subPropertyOf } x .}{u \text{ rdfs:subPropertyOf } x .}$	rdfs5	Transitivität
$\frac{u \text{ rdf:type rdf:Property .}}{u \text{ rdfs:subPropertyOf } u .}$	rdfs6	Reflexivität
$\frac{a \text{ rdfs:subPropertyOf } b . u \text{ a } y .}{u \text{ b } y .}$	rdfs7	Sub-Property Schlussfolgerungen für Instanzen
$\frac{u \text{ rdf:type rdfs:Class .}}{u \text{ rdf:type rdfs:Resource .}}$	rdfs8	Klassen sind Ressourcen

RDFS-Folgerung

$\frac{u \text{ rdfs:subClassOf } x . v \text{ rdf:type } u .}{v \text{ rdf:type } x .}$	rdfs9	Sub-Klassen Schlussfolgerungen für Instanzen
$\frac{u \text{ rdf:type rdfs:Class .}}{u \text{ rdfs:subClassOf } u .}$	rdfs10	Reflexivität
$\frac{u \text{ rdfs:subClassOf } v . v \text{ rdfs:subClassOf } x .}{u \text{ rdfs:subClassOf } x .}$	rdfs11	Transitivität
$\frac{u \text{ rdf:type rdfs:ContainerMembershipProperty .}}{u \text{ rdfs:subPropertyOf rdfs:member .}}$	rdfs12	
$\frac{u \text{ rdf:type rdfs:Datatype .}}{u \text{ rdfs:subClassOf rdfs:Literal .}}$	rdfs10	Jeder Datentyp ist eine Unterklasse von rdfs:Literal

RDFS-Folgerung

► Wichtige Definition: XML-Clash

ex:hatSmiley rdfs:range rdf:Literal .

ex:böseBemerkung ex:hatSmiley " >: - > "^^rdf:XMLLiteral .

► Tritt auf, wenn einem Knoten vom Typ rdf:Literal ein nicht-wohlgeformter Literalwert zugewiesen werden muss

RDFS-Folgerung

Satz:

Ein Graph RDFS-folgt aus G_1 genau dann, wenn es einen Graphen G'_1 gibt, der durch Anwendung der Regeln lg, gl, rdfsax, 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.

Agenda

- ▶ Motivation
- ▶ Vorbetrachtungen
- ▶ einfache Folgerung
- ▶ RDF-Folgerung
- ▶ RDFS-Folgerung
- ▶ Unzulänglichkeiten von RDF(S)

Was kann RDF(S) nicht?

- ▶ Bestimmte (vernünftig) scheinende Folgerungen können nicht RDFS-gefolgert werden, z.B.

```
ex:sprichtMit    rdfs:domain    ex:Homo .
ex:Homo         rdfs:subClassOf ex:Primates .
```

impliziert

```
ex:sprichtMit    rdfs:domain    ex:Primates .
```

- ▶ Mögliche Lösung: noch stärkere, “extensionale”, Semantik
- ▶ Keine Möglichkeit, Negation auszudrücken