ulm university universität ulm

Semantic Web Grundlagen

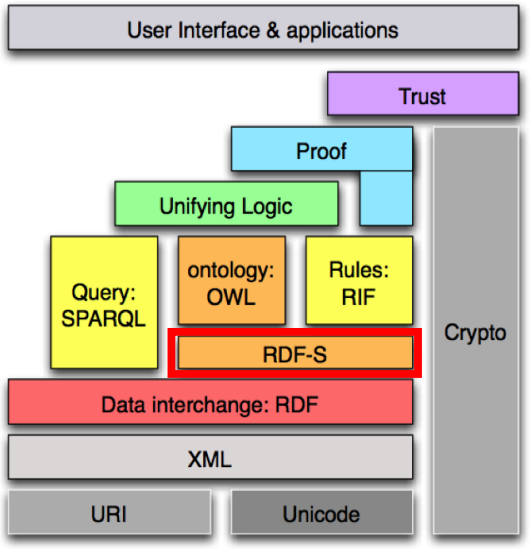
RDF Schema

Birte Glimm
Institut für Künstliche Intelligenz | 24. Okt 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 Implemetierung	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

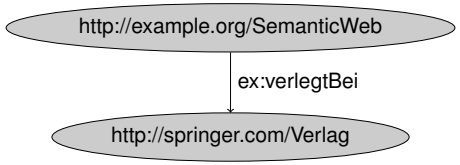


Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Schemawissen mit RDFS

- RDF bietet universelle Möglichkeit zur Kodierung von faktischen Daten im Web:



- = Aussagen über einzelne Ressourcen (Individuen) und deren Beziehungen
- wünschenswert: Aussagen über generische Mengen von Individuen (Klassen), z.B. Verlage, Organisationen, Personen etc.

Schemawissen mit RDFS

- Weiterhin wünschenswert: Spezifikation der logischen Zusammenhänge zwischen Individuen, Klassen und Beziehungen, um möglichst viel Semantik des Gegenstandsbereiches einzufangen, z.B.:
  “Verlage sind Organisationen.”
  “Nur Personen schreiben Bücher.”
- In Datenbanksprache: Schemawissen

Schemawissen mit RDFS

RDF Schema (RDFS):

- Teil der W3C Recommendation zu RDF
- Ermöglicht Spezifikation von schematischem (auch: terminologischem) Wissen
- Spezielles RDF-Vokabular (also: jedes RDFS-Dokument ist ein RDF-Dokument
- Namensraum (i.d.R. abgekürzt mit `rdfs`):
  `http://www.w3.org/2000/01/rdf-schema#`

Schemawissen mit RDFS

RDF Schema (RDFS):

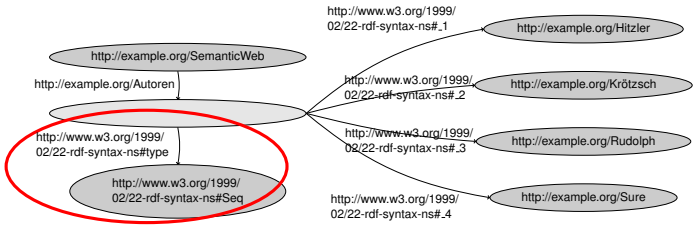
- jedoch: Vokabular nicht themengebunden (wie z.B. bei FOAF), sondern generisch
- erlaubt die Spezifikation (von Teilen) der Semantik beliebiger RDF-Vokabulare (ist also eine Art “Metavokabular”)
- Vorteil: jede Software mit RDFS-Unterstützung interpretiert jedes vermittels RDFS definierte Vokabular korrekt
- Funktionalität macht RDFS zu einer Ontologiesprache (für leichtgewichtige - engl.: lightweight - Ontologien)
- “A little semantics goes a long way.”

Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Klassen und Instanzen

- Typisierung von Ressourcen bereits in RDF zur Kennzeichnung von Listen:



- Prädikat `rdf:type` weist dem Subjekt das Objekt als Typ zu
- Objekt aufgefasst als Bezeichner für Klasse, der die durch das Subjekt bezeichnete Ressource (als sog. Instanz) angehört

Klassen und Instanzen

```
ex:SemanticWeb rdf:type ex:Lehrbuch .
```

- Charakterisiert “Semantic Web - Grundlagen” als Instanz der (neu definierten) Klasse “Lehrbuch”
- Klassenzugehörigkeit ist nicht exklusiv, z.B. mit o.g. Tripel gleichzeitig möglich:
  `ex:SemanticWeb rdf:type ex:Unterhaltsam .`
- Allgemein: a priori syntaktisch keine eindeutige Unterscheidung zwischen Individuen- und Klassenbezeichnern möglich
- Auch in der Realität Charakterisierung manchmal schwierig, beispielsweise für
  `http://www.un.org/#URI`

Die Klasse aller Klassen

- Jedoch manchmal eindeutige Kennzeichnung einer URI als Klassenbezeichner wünschenswert
- Möglich durch Typung der betreffenden URI als `rdfs:Class`

  ```
  es:Lehrbuch rdf:type rdfs:Class .
  ```

- `rdfs:Class` ist also die “Klasse aller Klassen” und enthält sich damit auch selbst, d.h. das folgende Tripel ist immer wahr:

  ```
  rdfs:Class rdf:type rdfs:Class .
  ```

Unterklassen - Motivation

- Gegeben Tripel
  `ex:SemanticWeb rdf:type ex:Lehrbuch .`
- Problem: Suche nach Instanzen der Klasse ex:Buch liefert kein Resultat
- Möglichkeit: Hinzufügen von Tripel
  `ex:SemanticWeb rdf:type ex:Buch .`
- Löst das Problem aber nur für die eine Ressource
  `ex:SemanticWeb`
- Automatisches Hinzufügen für alle Instanzen führt zu unnötig großen RDF-Dokumenten

Unterklassen

- Sinnvoller: einmalige Aussage, dass jedes Lehrbuch auch ein Buch ist, d.h. jede Instanz der Klasse `ex:Lehrbuch` ist automatisch auch eine Instanz der Klasse `ex:Buch`
- Realisiert durch die `rdfs:subClassOf`-Property:

```
ex:Lehrbuch rdfs:subClassOf ex:Buch .
```

"Die Klasse der Lehrbücher ist eine Unterklasse der Klasse der Bücher."

Unterklassen

- `rdfs:subClassOf`-Property ist reflextiv, d.h. jede Klasse ist Unterklasse von sich selbst, so dass z.B. gilt:

```
ex:Lehrbuch rdfs:subClassOf ex:Lehrbuch .
```

- Umgekehrt: Festlegung der Gleichheit zweier Klassen durch gegenseitige Unterklassenbeziehung, etwa:

```
ex:Hospital rdfs:subClassOf ex:Krankenhaus .
ex:Krankenhaus rdfs:subClassOf ex:Hospital .
```


Klassenhierarchien

- Üblich: nicht nur einzelne Unterklassenbeziehungen sondern ganze Klassenhierarchien (auch: Taxonomien) z.B.:

```
ex:Lehrbuch rdfs:subClassOf ex:Buch .
ex:Buch rdfs:subClassOf ex:Printmedium .
ex:Zeitschrift rdfs:subClassOf ex:Printmedium .
```

- In RDFS-Semantik verankert: Transitivität der `rdfs:subClassOf`-Property, d.h. es folgt automatisch

```
ex:Lehrbuch rdfs:subClassOf ex:Printmedium .
```

Klassenhierarchien

- Klassenhierarchien besonders ausgeprägt, etwa in Biologie (z.B. Klassifikation von Lebewesen)
- Z.B. zoologische Einordnung des modernen Menschen

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ex="http://www.semantic-web-grundlagen.de/Beispiele#">
  <rdfs:Class rdf:about="&ex;Animalia"/>
  <rdfs:Class rdf:about="&ex;Chordata">
    <rdfs:subClassOf rdfs:resource="&ex;Animalia"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="&ex;Mammalia">
    <rdfs:subClassOf rdfs:resource="&ex;Chordata"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="&ex;Primates">
    <rdfs:subClassOf rdfs:resource="&ex;Mammalia"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="&ex;Hominidae">
    <rdfs:subClassOf rdfs:resource="&ex;Primates"/>
  </rdfs:Class>
  ...
```


Klassen

- Intuitive Parallele zur Mengenlehre:

`rdf:type`	entspricht	$\in$
`rdfs:subClassOf`	entspricht	$\subseteq$

- Rechtfertigt beispielsweise auch die Reflexivität und Transitivität von `rdfs:subClassOf`

Klassen in RDF/XML-Syntax

- Verkürzte Darstellungen bei Angabe von Klasseninstanzen möglich:

  ```
  <ex:HomoSapiens rdf:about="&ex;BirteGlimm"/>
  ```

  an Stelle von

  ```
  <rdf:Description rdf:about="&ex;BirteGlimm">
    <rdf:type rdf:resource="&ex;HomoSapiens">
  </rdf:Description>
  ```

- Dementsprechend auch

  ```
  <rdfs:Class rdf:about="&ex;HomoSapiens"/>
  ```


Vordefinierte Klassenbezeichner

- `rdfs:Resource`
  Klasse aller Ressourcen (also sämtliche Elemente des Gegenstandsbereiches)
- `rdf:Property`
  Klasse aller Beziehungen
  (= die Ressourcen, die durch Prädikats-URIs referenziert werden)
- `rdf:List`, `rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdfs:Container`
  Klassen verschiedener Arten von Listen
- `rdfs:ContainerMembershipProperty`
  Klasse aller Beziehungen, die eine Enthaltenseinsbeziehung darstellen

Vordefinierte Klassenbezeichner

- `rdf:XMLLiteral`
  Klasse aller Werte des vordefinierten Datentyps XMLLiteral
- `rdfs:Literal`
  Klasse aller Literalwerte (enthält also alle Datentypen als Unterklassen)
- `rdfs:Datatype`
  Klasse aller Datentypen (ist also wie `rdfs:Class` eine Klasse von Klassen)
- `rdf:Statement`
  Klasse aller reifizierten Aussagen (s. dort)

Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Propertys

- Andere Bezeichnungen: Relationen, Beziehungen
- Achtung: Propertys sind in RDF(S) nicht (wie in OOP) speziellen Klassen zugeodnet
- Property-Bezeichner in Tripeln üblicherweise an Prädikatsstelle
- Charakterisieren, auf welche Art zwei Ressourcen zueinander in Beziehung stehen
- Mathematisch oft dargestellt als Menge von Paaren: verheiratetMit = $\{(\text{Adam}, \text{Eva}), (\text{Brad}, \text{Angelina}), \ldots\}$
- URI wird als Property-Bezeichner gekennzeichnet durch entsprechende Typung:
  `ex:verlegtBei rdf:type rdf:Property .`

Unterpropertys

- Ähnlich zu Unter-/Oberklassen auch Unter-/Oberpropertys denkbar und sinnvoll
- Darstellung in RDFS mittels `rdfs:subPropertyOf` z.B.:

```
ex:glücklichVerheiratetMit rdf:subPropertyOf
rdf:verheiratetMit .
```

- Erlaubt, aus dem Tripel

```
ex:Markus ex:glücklichVerheiratetMit ex:Anja .
```

zu schlussfolgern, dass

```
ex:Markus ex:verheiratetMit ex:Anja .
```

Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Einschränkung von Propertys

- Häufig: Property kann sinnvoll nur ganz bestimmte Ressourcen verbinden, z.B. verbindet `ex:verlegtBei` nur Publikationen mit Verlagen
- D.h. für alle URIs `a`, `b` folgt aus dem Tripel
  `a ex:verlegtBei b .`
  dass auch gilt:
  `a rdf:type ex:Publikation .`
  `b rdf:type ex:Verlag .`
- Kann in RDFS direkt kodiert werden:
  `ex:verlegtBei rdfs:domain ex:Publikation .`
  `ex:verlegtBei rdfs:range ex:Verlag .`
- Auch zur Angabe von Datentypen für Literale:
  `ex:hatAlter rdfs:range xsd:nonNegativeInteger .`

Einschränkung von Propertys

- Propertyeinschränkungen bieten die einzige Möglichkeit, semantische Zusammenhänge zwischen Propertys und Klassen zu spezifizieren
- Achtung: Propertyeinschränkungen wirken global und konjunktiv, z.B.

  ```
  ex:autorVon rdfs:range ex:Kochbuch .
  ex:autorVon rdfs:range ex:Märchenbuch .
  ```

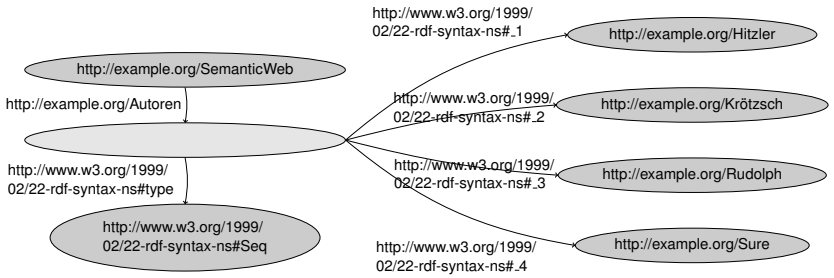
  bedeutet: jede Entität, von der jemand Autor ist, ist gleichzeitig Kochbuch und Märchenbuch
- Daher: als domain/range immer allgemeinste mögliche Klasse verwenden

Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Arbeit mit offenen Listen

Zur Erinnerung: offene Listen in RDF:



Arbeit mit offenen Listen

- Neue Klasse: `rdfs:Container` als Oberklasse von `rdf:Seq`, `rdf:Bag`, `rdf:Alt`
- Neue Klasse: `rdfs:ContainerMembershipProperty` Elemente sind keine Individuen im eigentlichen Sinne sondern selbst Propertys
- Intendierte Semantik: jede Property, die aussagt, dass das Subjekt im Objekt enthalten ist, ist Instanz von `rdfs:ContainerMembershipProperty`
- Es gilt also insbesondere

```
rdf:_1 rdf:type rdfs:ContainerMembershipProperty .
rdf:_2 rdf:type rdfs:ContainerMembershipProperty .
```

etc.

Arbeit mit offenen Listen

- Neue Property: `rdfs:member` Oberproperty aller in `rdfs:ContainerMembershipProperty` enthaltenen Propertys, also die “universelle Enthaltenseinsrelation”
- Damit in RDFS-Semantik verankert: wann immer für eine Property `p` das Tripel

```
p rdf:type rdfs:ContainerMembershipProperty .
```

gilt, folgt aus dem Tripel

```
a p b .
```

sofort das Tripel

```
a rdfs:member b .
```


Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Reifikation

- Problematisch in RDF(S): Modellierung von Aussagen über Aussagen (häufig zu erkennen am Wort “dass”), z.B.: “Der Detektiv vermutet, dass der Butler den Gärtner ermordet hat.”
- Erster Modellierungsversuch:

```
ex:detektiv ex:vermutet "Der Butler hat den
Gärtner ermordet.".
```

  - Ungünstig: auf Literal-Objekt kann schlecht in anderen Aussagen Bezug genommen werden (keine URI)
- Zweiter Modellierungsversuch:

```
ex:detektiv ex:vermutet
ex:derButlerHatDenGärtnerErmordet .
```

  - Ungünstig: innere Struktur der dass-Aussage geht verloren

Reifikation

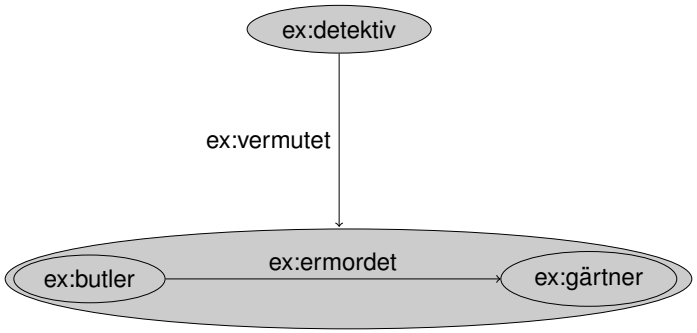
- Problematisch in RDF(S): Modellierung von Aussagen über Aussagen (häufig zu erkennen am Wort “dass”), z.B.: “Der Detektiv vermutet, dass der Butler den Gärtner ermordet hat.”
- Einzelne dass-Aussage leicht in RDF modellierbar:

```
ex:butler ex:hatErmordet ex:gärtner .
```

- Wünschenswert: ganzes RDF-Tripel als Objekt eines anderen Tripels; ist aber kein gültiges RDF

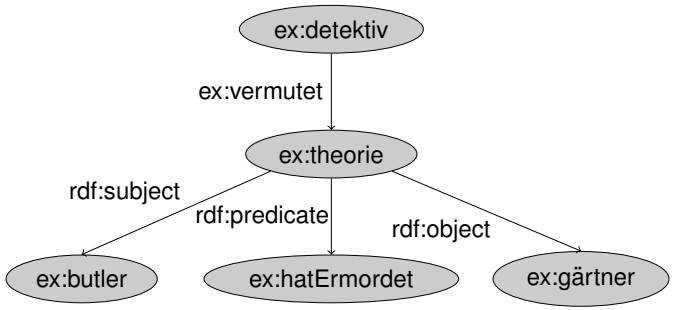
Reifikation

Lösung (ähnlich wie bei mehrwertigen Beziehungen): Hilfsknoten für die geschachtelte Aussage:



Reifikation

Lösung (ähnlich wie bei mehrwertigen Beziehungen): Hilfsknoten für die geschachtelte Aussage:

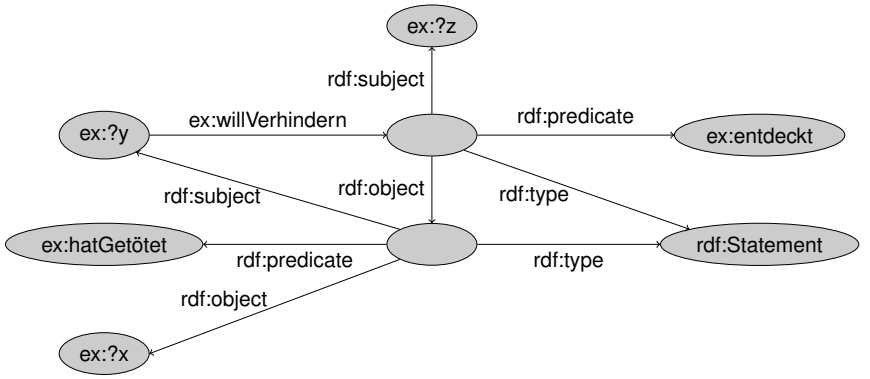


Reifikation

- Achtung: reifiziertes Tripel muss nicht unbedingt gelten (wäre auch nicht immer sinnvoll, z.B. bei Aussagen wie: “Der Detektiv bezweifelt, dass der Butler den Gärtner ermordet hat.”)
- Falls dies gewünscht ist, muss das originale (unreifizierte) Tripel dem RDF-Dokument nochmals hinzugefügt werden
- Der Klassenbezeichner `rdf:Statement` dient zur Kennzeichnung aller solcher Aussagen-Hilfsknoten
- Falls auf eine Aussage nicht (extern) Bezug genommen wird, kann der entsprechende Hilfsknoten ein bnode sein

Reifikation

Übungsaufgabe: noch eine Kriminalgeschichte...



Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Zusatzinformationen

- Wie bei Programmiersprachen manchmal Hinzufügen von Kommentaren (ohne Auswirkung auf Semantik) wünschenswert
- Zweck: Erhöhung der Verständlichkeit für menschlichen Nutzer
- Es empfiehlt sich (z.B. aus Tool-Kompatibilitätsgründen) auch dieses Wissen als Graph zu repräsentieren
- Also: Satz von Propertys, die diesem Zweck dienen

Zusatzinformationen

`rdfs:label`

- Property, die einer (beliebigen) Ressource einen alternativen Namen zuweist (Literal)
- Oftmals sind URIs schwer lesbar; zumindest “unhandlich” durch `rdfs:label` zugewiesener Name wird z.B. häufig von Tools bei der graphischen Darstellung verwendet

Beispiel (inkl. Sprachinformation):

```
<rdfs:Class rdf:about="&ex;Hominidae">
  <rdfs:label xml:lang="de">Menschenaffen</rdfs:label>
</rdfs:Class>
```


Zusatzinformationen

`rdfs:comment`

- Property, die einer (beliebigen) Ressource einen umfangreichen Kommentar zuweist (Literal)
- Beinhaltet z.B. natürlichsprachliche Definition einer neu eingeführten Klasse – erleichtert spätere intentionsgemäße Wiederverwendung

`rdfs:seeAlso, rdfs:definedBy`

- Properties, die Ressourcen (URIs!) angeben, die weitere Informationen bzw. eine Definition der Subjekt-Ressource bereitstellen

Zusatzinformationen

Verwendungsbeispiel

```
:
xmlns:wikipedia="http://de.wikipedia.org/wiki"
:
<rdfs:Class rdf:about="\&ex;Primates">
  <rdfs:label xml:lang="de">Primaten</rdfs:label>
  <rdfs:comment>
    Eine Säugetierordnung. Primaten zeichnen sich durch
    ein hochentwickeltes Gehirn aus. Sie besiedeln
    hauptsächlich die wärmeren Erdregionen.
    Die Bezeichnung Primates (lat. "Herrentierre") stammt
    von Carl von Linné.
  </rdfs:comment>
  <rdfs:seeAlso rdfs:resource="/&wikipedia;Primaten"/>
  <rdfs:subClassOf rdfs:resource="\&ex;Mammalia"/>
</rdfs:Class>
```


Agenda

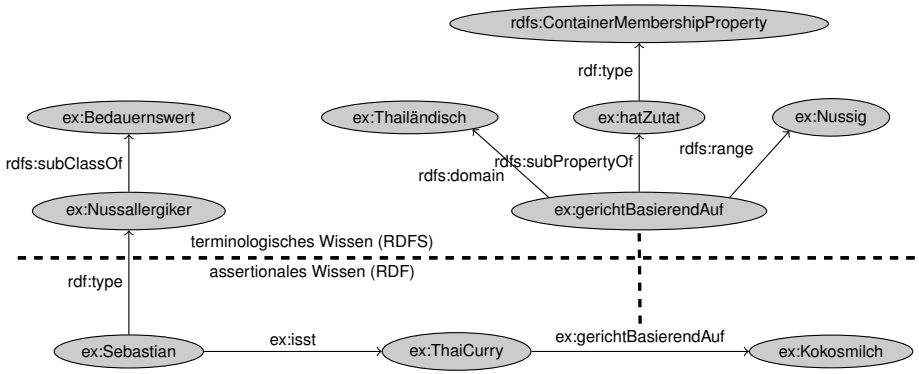
- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien

Einfache Ontologien

- Mit den durch RDFS bereitgestellten Sprachmitteln können bestimmte Gegenstandsbereiche bereits in wichtigen Aspekten semantisch erfasst werden
- Auf der Basis der speziellen Semantik von RDFS kann schon ein gewisses Maß impliziten Wissens geschlussfolgert werden
- Mithin stellt RDFS eine (wenn auch noch vergleichsweise wenig ausdrucksstarke) Ontologiesprache dar

Einfache Ontologien - Beispiel

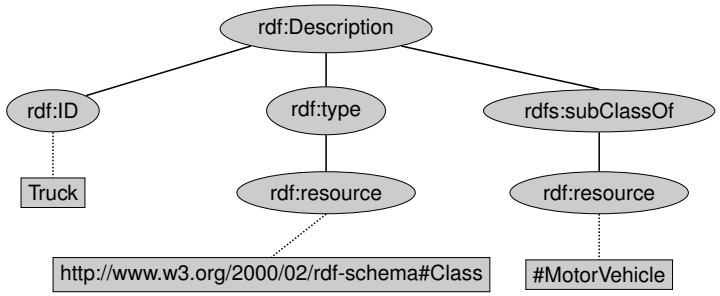
```
ex:ThaiCurry           ex:gerichtBasierendAuf ex:Kokosmilch .
ex:Sebastian           rdf:type               ex:Nussallergiker .
ex:Sebastian           ex:isst                ex:ThaiCurry .
ex:Nussallergiker      rdfs:subClassOf        ex:Bedauernswert .
ex:gerichtBasierendAuf rdfs:domain            ex:Thailändisch .
ex:gerichtBasierendAuf rdfs:range             ex:Nussig .
ex:gerichtBasierendAuf rdfs:subPropertyOf     ex:hatZutat .
ex:hatZutat            rdf:type               rdfs:ContainerMembershipProperty .
```



1 Dokument - 3 Interpretationen

```
<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource=
    "http://http://www.w3.org/2000/02/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

Interpretation als XML:



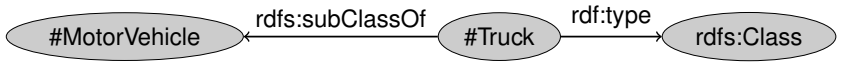
1 Dokument - 3 Interpretationen

```
<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource=
    "http://http://www.w3.org/2000/02/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

Interpretation als RDF:

- Anderes Datenmodell
- `rdf:Description`, `rdf:ID` und `rdf:resource` haben eine festgelegte Bedeutung

subject	predicate	object
`#Truck`	`rdf:type`	`rdfs:Class`
`#Truck`	`rdfs:subClassOf`	`#Motorvehicle`

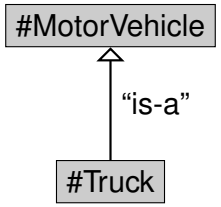


1 Dokument - 3 Interpretationen

```
<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource=
    "http://http://www.w3.org/2000/02/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

Interpretation als RDF Schema:

- Wieder anderes Datenmodell
- `rdf:type` und `rdf:subClassOf` werden speziell interpretiert



Agenda

- Motivation
- Klassen und Klassenhierarchien
- Propertys und Propertyhierarchien
- Einschränkungen auf Propertys
- Offene Listen
- Reifikation
- Zusätzliche Informationen in RDFS
- Einfache Ontologien