Repräsentationssprachen für Ontologien

Kay Girmann

Universität Leipzig

13. Januar 2009

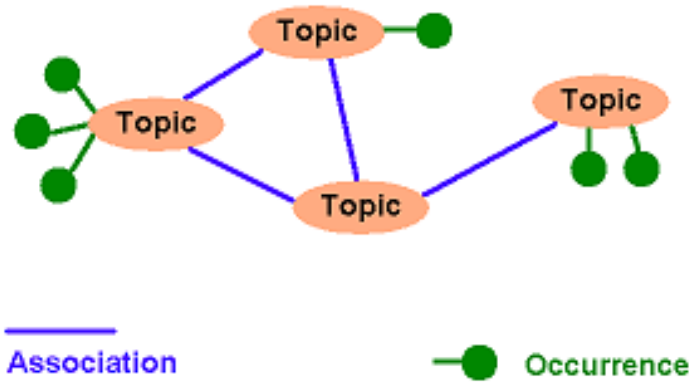
Gliederung

Sprachen für Ontologien

- Topic Maps
- DAML+OIL
- RDF(S)
- OWL

Topic Maps

- internat. Industriestandard (ISO 13250) zur Darstellung und zum Austausch von Wissen
- beschreibt Wissensstrukturen und verknüpft sie mit relevanten Informationsquellen
- Ziel: bessere Suche in Internetressourcen und Dokumenten ermöglichen
- de facto Standard ist XTM mit XML basierte Syntax
- Benutzt:
  - topics - Konzepte
  - associations - Verknüpfungen, Beziehungen
  - occurrences - Beziehungen zwischen Themen und Informationsquellen

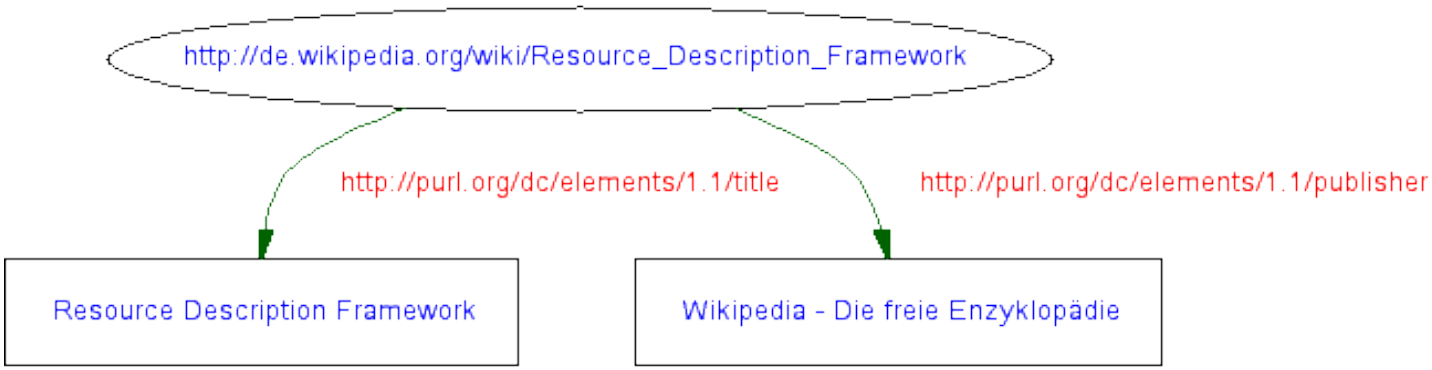


DAML+OIL

- DAML
  - von der DARPA entwickelt
  - fürte zur Entwicklung einer WebOnt Sprache durch das W3C
  - Syntax basiert auf XML und RDF und erweitert diese
  - ein groteil von DAML ist in OWL eingeflossen
- OIL
  - basiert auf Konzepten aus der Beschreibungslogik
  - ist kompatibel zu RDFS
- DAML+OIL
  - kombiniert Eigenschaften von DAML und OIL
  - wurde durch OWL abgelöst

RDF

- wurde vom W3C zusammen mit OWL als Grundstein für das Semantische Web entwickelt
- dient der Beschreibung strukturierter Daten
- stellt Metadaten im Internet bereit
- maschinenlesbar und maschinenverständlich
- es gibt verschiedene Darstellungsformen
  - Graph
  - XML
  - Notation 3 (N3)



RDF-Modell

- besteht aus drei Objekttypen: Subjekt, Prädikat und Objekt
- Subjekte: Dinge die beschrieben werden, erhalten eindeutige Bezeichnung URI
- Prädikate: erläutert was das Subjekt macht
- Objekte: beschreiben was das Prädikat macht
- Beispiel:

  `Buch hat Titel.`

RDF Notation 3

- Notation 3 besteht aus Tupeln
- Subjekt Prädikat Objekt .

```
<http://example.org/Mann>
     <http://example.org/istEine>
          <http://example.org/Person>.

<http://example.org/Buch>
     <http://example.org/hat>
          <http://example.org/Autor>,
          <http://example.org/Titel>.
```

RDF XML

- XML-Syntax
- nutzt Namespaces
- hierarchisch aufgebaut
- Datentypen wie in XML
  - `"Faust"^^xsd:string`
- Angabe der Sprache
  - `"Buch"@de`

Beispiel

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex ="http://example.org/">
<rdf:Description rdf:about="http://example.org/Buch">
  <ex:hat>
    <rdf:Description rdf:about="http://example.org/Autor" />
    <rdf:Description rdf:about="http://example.org/Titel" />
  </ex:hat>
</rdf:Description>
```

RDF Schema

- Empfehlung des W3C
- ermöglicht Ressourcen aus RDF in Beziehungen zu setzen
- basiert auf der Idee eines mengenorienterten Klassensystems

Konzepte

- Klassen
  - `Class` - legt abstraktes Objekt fest
  - `Resource` - jede Entität in RDF ist Instanz dieser Klasse
  - `Property` - Basisklasse für Eigenschaften, Unterklasse von `Resource`
  - `Literal` - Klasse für Literalwerte, Zeichenketten etc.
- Eigenschaften
  - `subClassOf` - legt Vererbungshierachien von Klassen fest
  - `subPropertyOf` - legt Vererbungshierarchien von Eigenschaften fest
  - `domain` - legt Datentyp des Subjekts einer Eigenschaft im Bezug auf eine Klasse fest
  - `range` - legt Datentyp des Objekts einer Eigenschaft fest

Beispiele

- Klassen
  - `<rdfs:Class rdf:about="Person" />`
- Individuen sind Instanzen von Klassen
  - `<Person rdf:about="Max" />`
- Unterklassen
  - `ex:Autor rdf:type ex:Person .`
  - `ex:Lehrbuch rdfs:subClassOf ex:Buch .`
- Offene Listen
  - `ex:SemanticWeb rdfs:member ex:Buch .`

OWL - Web Ontologie Language

- wurde im Februar 2004 vom W3C als Ontologiesprache standardisiert
- basiert auf RDF-Syntax und historisch auf DAML+OIL
- Ausdrucksstärker als RDF-Schema
- kann Ausdrücke ähnlich der Prädikatenlogik formulieren
- es gibt 3 Teilsprachen: OWL Full, OWL DL und OWL Lite

OWL Full

- enthält OWL DL und OWL Lite
- enthält ganz RDF(S), d.h. alle Sprachelemente aus OWL und RDF(S) sind uneingeschränkt erlaubt
- sehr ausdrucksstark
- es gibt keine Abgrenzung in Rollen, Klassen und Individuen
- unentscheidbar
- von aktuellen Softwarewerkzeugen nur bedingt unterstützt

OWL DL

- enthält OWL Lite und ist Teilsprache von OWL Full
- konzipiert um entscheidbar zu sein
- ob eine Aussage gefolgert werden kann, ist mit Algorithmus beantwortbar, Terminierung ist garantiert
- wird bevorzugt in wissenschaftlichen Umfeld eingesetzt
- Einschränkungen gegenüber OWL Full
  - RDF(S): Verwendung von `rdfs:Class` und `rdf:Property` verboten
  - klare Unterscheidung zwischen Klassen, Individuen, abstrakten Rollen, konkreten Rollen, Datentypen und im Kopf zu deklarierenden Ontologieeigenschaften
  - verwendete Klassen und Rollen müssen als solche deklariert werden

OWL Lite

- Idee: einfach zu implementierendes Sprachfragment zu bieten
- enthält wichtigste Sprachelemente
- Einschränkungen für OWL DL gelten
- Verwendung von `owl:oneOf`, `owl:unionOf`, `owl:complementOf`, `owl:hasValue`, `owl:disjointWith`, `owl:DataRange` verboten
- Zahlenrestriktionen nur mit 0 und 1 erlaubt

Grundelemente in OWL-DL

- Klassen
  - `<owl:Class rdf:about="Person" />`
- Individuen sind Instanzen von Klassen
  - `<Person rdf:about="Max" />`
- abstrakte Rollen: verbinden Individuen mit Individuen
  - `<owl:ObjektProperty rdf:about="hatKind"/>`
- konkrete Rollen: verbinden Individuen mit Datentypen
  - `<owl:DatatypeProperty rdf:about="Nachname"/>`

OWL und Prädikatenlogik

OWL Konstruktoren

Konstruktor	DL Syntax	Beispiel
intersectionOf	$C_1 \sqcap \cdots \sqcap C_n$	$\texttt{Person} \sqcap \texttt{Männlich}$
unionOf	$C_1 \sqcup \cdots \sqcup C_n$	$\texttt{Professor} \sqcup \texttt{Doktor}$
complementOf	$\neg C$	$\neg\texttt{Männlich}$
oneOf	$\{x_1 \ldots x_n\}$	$\{\texttt{max,petra}\}$
allValuesFrom	$\forall P.C$	$\forall\texttt{hatKind.Doktor}$
someValuesFrom	$\exists r.C$	$\exists\texttt{hatKind.Professor}$
hasValue	$\exists r.\{x\}$	$\exists\texttt{BürgerVon.}\{\texttt{Leipzig}\}$
minCardinality	$(\geq nr)$	$(\geq\texttt{2hatKind})$
maxCardinality	$(\leq nr)$	$(\leq\texttt{1hatKind})$
inverseOf	r^-	$\texttt{hatKind}^-$

OWL und Prädikatenlogik

OWL Axiome

Axiom	DL Syntax	Beispiel
subClassOf	$C_1 \sqsubseteq C_2$	$\texttt{Person} \sqsubseteq \texttt{Tier} \sqcap \texttt{Zweibeiner}$
equivalentClass	$C_1 \equiv C_2$	$\texttt{Mann} \equiv \texttt{Person} \sqcap \texttt{Männlich}$
subPropertyOf	$P_1 \sqsubseteq P_2$	$\texttt{hatTochter} \sqsubseteq \texttt{hatKind}$
equivalentProperty	$P_1 \equiv P_2$	$\texttt{Kosten} \equiv \texttt{Preis}$
disjointWith	$C_1 \sqsubseteq \neg C_2$	$\texttt{Männlich} \sqsubseteq \neg\texttt{Weiblich}$
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	$\{\texttt{Max}\} \sqsubseteq \neg\{\texttt{Peter}\}$

Beispiel

- Ein Mann ist eine Person und nicht weiblich

```
<owl:Class rdf:about="Mann">
  <rdfs:subClassOf rdf:resource="Person" />
  <owl:disjointWith rdf:resource="weiblich" />
</owl:Class>
```

- Person ist Teilmenge von Tier und Zweibeiner

```
<owl:Class rdf:about="Person">
  <rdfs:subClassOf>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="Tier" />
      <owl:Class rdf:about="Zweibeiner" />
    </owl:intersectionOf>
  </rdfs:subClassOf>
</owl:Class>
```