Aufgabenblatt 2
Recherchebericht

Aufgabe 1 - Begriffe

von Simon Hetzer

Inferenz → Deutsch: „*Schlussfolgerung*“, bezeichnet aus einer Reihe von Aussagen (Annahmen) eine weitere Aussage (Konklusion). Die so gewonnene Konklusion sollte korrekt (richtig) sein.
Das Wort Inferenz (englisch: *inference*), welches auch in unserem Projekt „*Easy Inferenz*“ auftaucht weist darauf hin, dass die Schlussfolgerungen automatisch (computergestützt) vollzogen werden sollen.

Ontologie → Eine Sammlung von Begriffen (Klassen), zugehörigen Eigenschaften und logischen Beziehungen zwischen diesen. Im Gegensatz zu klassischen Datenbanken ist hervorzuheben, dass die Ontologie die Zusammenhänge erkennt. Zur Beschreibung von Ontologien existieren wiederum mehrere Sprachen, u.a. RDF-S und OWL.
Bezogen auf Ontowiki entspricht die Ontologie dem Modell der →*Wissensbasis*. Dabei werden gültige Zusammenhänge für die Modelldaten beschrieben.

Plugin → Das Ontowiki soll über einen Plugin-Mechanismus erweitert werden. Dabei bezeichnet das Plugin eine für sich nicht funktionsfähige Softwarekomponente, die die Software um eine Funktionalität erweitert. In unserem Fall wird dazu die von Ontowiki bereitgestellte Plugin-Schnittstelle verwendet. Diese umfasst die drei Komponententypen „*Komponente*“, „*Modul*“ sowie „*Plugin*“. Möglicherweise können in unserem Plugin mehrere dieser Typen verwendet werden.

Benutzer → Das von uns entwickelte Softwareprodukt soll dem Benutzer dienen. Der Benutzer muss sich nicht zwangsweise mit dem Einrichten von →*Plugins* auskennen. Für den Benutzer sollte sich durch das Plugin ein zusätzlicher Nutzen ergeben. Er muss ohne Informatikkenntnisse mit dem Plugin interagieren können (→*Regeln* einrichten und dadurch abgeleitete →*Inferenzen* ansehen).

Regel → Regeln meint hier insbesondere komplexe Regeln zur Schlussfolgerung (→ *Inferenz*). Diese Regeln müssen auf die jeweiligen →*Ontologien* angepasst werden. Die Regeln beschreiben eine logische Verknüpfung von Aussagen und daraus folgenden Konklusionen.

URI → Der *Uniform Resource Identifier* (ursprünglich *Universal Resource Identifier*) dient der Identifikation von Ressourcen und wird u.a. auch im Zusammenhang mit OWL und RDF verwendet.

Klasseneigenschaft → Bezeichnet die Eigenschaften einer Klasse in der →*Ontologie*. Unser →*Plugin* soll solche Eigenschaften auf Basis anderer Aussagen generieren.

Wissensbasis / Wissensdatenbank → Die Daten in der →*Ontologie* bilden die Wissensbasis. Sie besteht aus Objekten (Konkretisierung der Klasse) mit zugehörigen Eigenschaften sowie Beziehungen zwischen diesen Objekten.

Semantic Web → Die Idee des Semantic Web besteht darin, die Daten mit Bedeutung zu verknüpfen. Dies geschieht vornehmlich indem Beziehungen zwischen den Daten hergestellt werden. Dazu dienen wiederum →*Ontologien* bzw. Ontologie-Sprachen wie RDF.

Virtuoso → Eine Datenbank mit RDF Triple Store und SPARQL-Unterstützung und somit eine wichtige Basis für Ontowiki.

implizite/explizite Informationen → In der Wissensbasis werden viele explizite Beziehungen zwischen Objekten angegeben. Diese sind die von den →*Benutzern* ursprünglich eingetragenen Informationen. Häufig existieren neben diesen expliziten Informationen aber noch implizite Informationen, wie z.B. umgekehrte Relationen oder auch komplexere Konklusionen. Diese dem Benutzer zugänglich zu machen kann hilfreich sein und ist Aufgabe unseres Projekts.

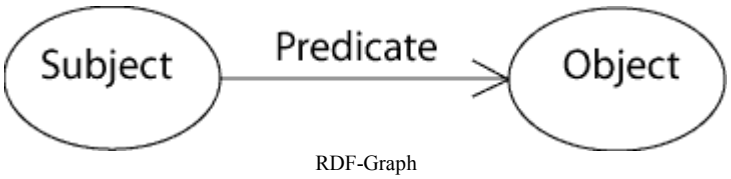
Aufgabe 2 - Konzepte

RDF (Resource Description Framework) von Quan Nguyen

RDF ist eine formale Sprache zur Beschreibung von Ressourcen im Web. Diese Sprache ist darauf ausgelegt Wissen zu repräsentieren und weniger Daten, sprich alle RDF-Ausdrücke haben eine gewisse Bedeutung.

Unter Ressourcen versteht man all das, was durch einen eindeutigen Bezeichner im URI-Format benannt wird. Eine Ressource kann zum Beispiel eine Webseite sein - so identifiziert die URI *http://pcai042.informatik.uni-leipzig.de/~swp09-7* unsere Gruppenhomepage. Die Ressource, die durch eine URI benannt wird muss aber nicht zwangsläufig im Web erreichbar sein. So können E-Mail-Accounts (z.B. mit *mailto:123@example.com*), Bücher (z.B. mit *urn:isbn:978-3898530194*), und generelle Konzepte mit URIs identifiziert werden.

Durch die formale Repräsentation in RDF sind Informationen von Maschinen auswertbar. Dadurch sind sie maschinell durchsuchbar und implizit vorhandene Informationen können durch den Einsatz von genaueren Spezifikationen des modellierten Bereichs, z.B. mit Hilfe von *RDF-Schema* (RDFS) oder der *Web Ontology Language* (OWL), maschinell erschlossen werden obwohl die Information explizit nicht vorliegt.
Es gibt 2 komplementäre Wege RDF-Informationen zu betrachten. Der erste Weg ist die Betrachtung der Informationen mit Hilfe von RDF-Ausdrücken/Tripeln. Jeder Ausdruck repräsentiert einen Fakt. Der Zweite ist durch einen Graphen. RDF als Graph drückt exakt das gleiche aus, nur ist diese Form übersichtlicher und erleichtert das Erkennen von Struktur in den Daten.



RDF-Graph

Das RDF-Modell setzt sich zusammen aus Tripeln der Form (Subjekt, Prädikat, Objekt). Dieses Tripel sagt aus, dass eine gerichtete Beziehung zwischen dem Subjekt und dem Objekt besteht. Das Prädikat stellt dabei eine binäre Relation dar.
Subjekt und Prädikat sind immer Ressourcen, das Objekt kann entweder eine Ressource oder ein Literal sein. Literale sind Zeichenketten welche Datentypwerte repräsentieren. Auf diese Weise lassen sich Wahrheitswerte (z.B. *true* und *false*), Zahlen (z.B. *42* oder *0x2A*) oder Datumsangaben (z.B. *2009-02-01-21:00*) spezifizieren.

RDF Ausdrücke bilden selbst wieder Ressourcen, auf die mit weiteren Tripeln verwiesen werden kann.

Das RDF-Modell ist unabhängig von einer speziellen Darstellungsform - die verbreitetste Notation ist die in *RDF/XML*. Weitere Notationen sind *N3*, *Turtle* und *N-Triples*. Diese unterscheiden sich lediglich in der Darstellungsform der Tripel.

Hier nun ein Beispiel für die unterschiedlichen Notationsmöglichkeiten von ein und demselben Ausdruck:

Tripel

Subjekt	*Prädikat*	*Objekt*
http://www.w3.org/	http://purl.org/dc/elements/1.1/title	"World Wide Web Consortium"

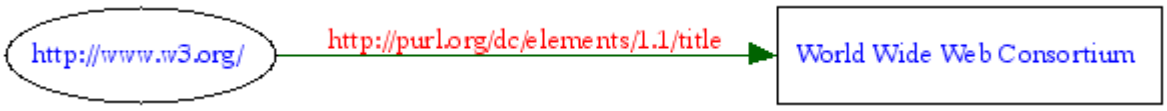
N-Triples

<http://www.w3.org/> <http://purl.org/dc/elements/1.1/title> "World Wide Web Consortium" .

RDF/XML

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:                      xmlns:dc="http://purl.org/dc/elements/1.1/">
4:    <rdf:Description rdf:about="http://www.w3.org/">
5:         <dc:title>World Wide Web Consortium</dc:title>
6:    </rdf:Description>
7: </rdf:RDF>
```

Graph



Für ausführlichere Informationen und Tutorials:

- http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-blank-node
- http://www.w3schools.com/rdf/default.asp
- http://rdfabout.com/quickintro.xpd

RDF-Schema

von Martin Strack

Das RDF-Schema (RDF-S) ist ein vom W3C Konsortium festgelegter Standard und dient als Semantikerweiterung für das RDF Modell.

Da RDF nur Aussagen über Ressourcen, deren Eigenschaften und Werte macht, benötigt man ein erweitertes Vokabular um Beziehungen zwischen Ressourcen und Eigenschaften zu erstellen. Zu den Erweiterungen, die das RDF Schema mit sich bringt zählen vor allem die Klassenkonzepte, die dazu dienen Gruppen ähnlicher Ressourcen und deren Beziehung untereinander zu beschreiben. Das RDF-Schema selbst beschreibt also den Aufbau und die Zugehörigkeit von Klassen und RDF die Bedeutung der einzelnen Klassen.

Beispiel:

```
<rdfs:Class rdf:ID="human" />
<rdfs:Class rdf:ID="male">
     <rdfs:subClassOf rdf:resource="#human"/>
</rdfs:Class>
<rdf:Property rdf:ID="name">
```

```
        <rdfs:domain rdf:resource="#human"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </rdf:Property>
```

Die Ressourcen und die Bedeutung der oben dargestellten Klassen sind durch RDF definiert. Die Hierarchie der Klassen - und damit auch die Beziehungen zwischen diesen - beschreibt hingegen das RDF-Schema.

Für unser Projekt bringt die Semantikerweiterung einige Vorteile, da der Einsatz von Vererbungen und die Verwendung von Unterklassen die Möglichkeit bietet, Ressourcen untereinander zu verknüpfen.

Ausführliche Beispiele und Erläuterungen findet man auf den folgenden Seiten:
http://www.w3.org/TR/rdf-schema/#ch_summary
http://www.isi.edu/expect/web/semanticweb/rdfs.html#range

OWL

von Andreas Krause

Die *Web Ontology Language* (OWL) ist eine Spezifikation vom *World Wide Web Consortium* (W3C) zur Definition und Realisierung von Ontologien. OWL basiert technisch auf der RDF-Syntax, bietet aber mächtigere Sprachkonstrukte als das RDF-Schema, wodurch die Modellierung komplexer Wissensbasen vereinfacht wird. Die OWL Sprache stellt drei zunehmend ausdrucksstarke Untersprachen bereit, da aufgrund der Komplexität von OWL vollständige Inferenz nicht immer möglich ist. Es wurden die Sprachebenen *OWL Lite*, *OWL DL* und *OWL Full* definiert, wobei OWL Lite und OWL DL Einschränkungen unterliegen um vollständige Inferenz zu ermöglichen und OWL Full die maximale Ausdrucksstärke besitzt, wodurch Ontologien unentscheidbar seien könnten.

Eine OWL-Ontologie besteht aus Beschreibungen von Klassen, Properties sowie ihren Instanzen aus denen sich logische Konsequenzen ableiten lassen. Klassen dienen hierbei zur Begriffsdefinition (Definition von Konzepten), deren Eigenschaften durch Properties bestimmt werden. Instanzen sind Individuen einer Klasse, zum Beispiel könnte es zur Klasse „*Stadt*“ die Instanzen „*Berlin*“ und „*Hamburg*“ geben.

Mit Hilfe von Klassen können in OWL nicht nur einfache Taxonomien erstellt werden, sondern es lassen sich auch komplexe Klassen aus bereits bestehenden Klassen durch die Mengenoperationen Schnittmenge, Vereinigungsmenge und Komplementärmenge definieren. Ein weiteres Werkzeug zur Erstellung komplexer Klassen sind Einschränkungen (Restrictions). Klassen, die einer Einschränkung unterliegen, besitzen nur Individuen, die die Restriktionsbedingungen erfüllen. Man könnte beispielsweise die Klasse „*Rotwein*“ so definieren, dass Sie aus allen „*Weinen*“ besteht, deren Wert der Eigenschaft Farbe „*rot*“ ist.

In OWL können Properties noch weiter spezifiziert werden. Dadurch ist es möglich Properties durch Mechanismen wie Inverse einer anderen Property, Transitivität, Symmetrie und Funktionalität zu erweitern, wodurch man ein erweitertes Folgern aus Properties erreichen kann.

OWL wird für unser Projekt „*Easy Inferenz PlugIn*“ in vielerlei Hinsicht von Bedeutung sein. Zum einen muss das PlugIn in der Lage sein, bestehende Ontologien, die OWL verwenden (z.B. Professorenkatalog), zu verarbeiten. Weiterhin werden uns die speziellen Properties (TransitiveProperty, SymmetricProperty, FunctionalProperty, inverseOf), bei der Generierung impliziter Informationen unterstützen. Eine weitere mögliche Anwendung von OWL in unseren Projekt ist es die Gleichheit von Individuen (durch owl:sameAs) auszudrücken. Dies könnte notwendig sein, wenn vorhandene Ontologien integriert werden.

Eine sehr gute Einführung in das Thema OWL bietet der "*OWL Web Ontology Language Guide*" des W3C. Eine deutsche Übersetzung ist unter folgender URL zu finden:
http://www.semaweb.org/dokumente/w3/TR/2004/REC-owl-guide-20040210-DE.html

SPARQL

von Maik Bärwald

Die „***S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage*“ ist - wie der Name schon sagt – eine Anfragesprache für RDF. Im Januar 2008 wurde dieser Nachfolger der „*RDF Query Language*“ von der W3C[1] zum Standard erklärt. SPARQL ermöglicht es, die durch RDF implizit gespeicherten Metadaten gezielt aus Webseiten und anderen Ressourcen herauszusuchen und darzustellen. Sowohl im Aufbau als auch in der Funktion ähnelt es dabei stark der Anfragesprache SQL für Datenbanken.

Der Aufbau einer typischen SPARQL-Anfrage gliedert sich in drei Bestandteile:

```
PREFIX books:   <http://example.org/book/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
SELECT ?book ?title
WHERE { ?book dc:title ?title }
```

Der PREFIX gibt die zu durchsuchenden Ressourcen an. Dabei handelt es sich in der Regel um URIs, die aufgrund ihrer Länge durch eine Variable (hier: dc) abgekürzt werden. Im Zuge der Standardisierung wurden von einigen Arbeitsgruppen Kernmengen von Metadaten definiert, um ein wenig Ordnung in das Chaos der frei definierbaren Metadaten zu bringen. Jeder hat Zugriff auf diese Standard-Datentypen und kann sie über das PREFIX einbinden. Im obigen Beispiel werden die „*Dublin Core Metadaten*[2]“ genutzt.

Der SELECT-Teil definiert Variablen, die nach der Anfrage angezeigt werden sollen. Variablen kennzeichnen sich stets durch ein vorangestelltes „?“ oder „$“.

In der WHERE-Klausel wird die eigentliche Anfrage durch Ausnutzung der RDF-Tripel definiert. Dabei werden für jeden Teil der Anfrage eine Ressource, ein Prädikat (bzw. eine Beziehung) und eine weitere Ressource oder ein Literal benötigt. Darüber hinaus lassen sich in die WHERE-Klausel viele Anfrageoptionen integrieren. Einige Beispiele sind:

- FILTER => Beschränkt die Auswahl der Kriterien (z.B. FILTER(?menge < 5))
- UNION => Verbindet zwei Anfragen in einer WHERE-Klausel ({Anfrage} UNION {Anfrage})
- DISTINCT => Unterbindet Duplikate
- OPTIONAL => Zusätzliche Anfragen oder Filteroptionen, wenn die optionale Bedingung erfüllt ist

Im Anschluss an die WHERE-Klausel lassen sich noch verschiedene Optionen zur Ausgabemanipulation setzen. So zum Beispiel die Sortierfunktion ORDER BY oder die Ergebnisbeschränkung LIMIT.

Durch Ausnutzung bereits bestehender Metadaten-Strukturen, wie sie durch RDF geschaffen werden, lassen sich durch entsprechende Anfragen somit komplexe Zusammenhänge in Webdaten und Wissensbasen wie dem Professorenkatalog erkennen und anzeigen. Eine Automatisierung dieser Anfragen ermöglicht die Generierung impliziter Informationen ohne den genauen Datenbestand kennen zu müssen. Da SPARQL graphbasierte Ergebnisse liefert (bedingt durch die Menge von RDF-Tripeln („*RDF-Graph*“)), lassen sich so nicht nur gezielte Informationen, sondern ganze Netzwerke von Inferenzen generieren.

SPARQL bildet somit die Schnittstelle zwischen den Techniken des „Semantic Web“ und dem Benutzer, der erst durch die entsprechenden Anfragen in die Lage versetzt wird die inferenten Informationen zu nutzen.

Nützliche Links:

- Tutorial (engl.) von Grundlagen über erste Anfragen → http://www.xml.com/pub/a/2005/11/16/introducing-

1 **W3C** („*World Wide Web Consortium*“) ist ein Gremium, dessen Aufgabe die Standardisierung Web-basierter Techniken ist → http://www.w3.org/

2 Ein weit verbreitetes Metadaten-Schemata der „*Dublin Core Metadata Initiative*“ (**DCMI**), bestehend aus 15 Kernfeldern → http://dublincore.org/documents/dces/

sparql-querying-semantic-web-tutorial.html?page=1
- einfache Beispielanfragen → http://www.sparql.org/query.html

Aufgabe 3 – Beschreibung der zu studierenden Applikation

OntoWiki von Patrick Beer

Aktuell wird das System des OntoWikis an der Universität Leipzig zur Visualisierung des Professorenkatalogs verwendet. Jedoch können derzeitig nur explizit gespeicherte Daten aus dem Modell heraus angezeigt werden - obwohl auch schon impliziter Datenbestand vorhanden ist.

Unsere Aufgabe ist es, die Möglichkeiten der Navigation durch das OntoWiki unabhängig des zu Grunde liegenden Modells deutlich zu erhöhen. Dies gelingt durch die sinnvolle Nutzung durch Inferenzen (Schlussfolgerungen) aus den schon gespeicherten Daten. Durch das Einrichten von Regeln kann so Speicherplatz gespart werden, der sonst durch weitere explizite Daten belegt werden würde. Dabei sollen die Ontologiesprachen RDF und OWL diese Inferenzen beschreiben. Neue implizite Informationen sollen dann über die Weboberfläche dem Benutzer zusätzlich, getrennt und hervorgehoben, angezeigt werden. Es können dabei verschiedene Regeltypen in Betracht gezogen werden. Die einfachste Methode wäre die inverse Relation („hatLebensabschnitt“ vs. „istLebensabschnittVon“). Aber auch andere logische Verknüpfungen könnten dabei in Betracht gezogen werden (z.B. Transitivität). Dies soll dann das von unserem Team zu entwickelte Plugin realisieren.

Am Ende sollen zwei von einander getrennte Modelle als Basis der Informationsanzeige dienen. Durch die Vernetzung der Referenzen aus dem Grundmodell mit den Regeln aus dem Inferenz-Modell bleibt es möglich, das Plugin ab- und zu zuschalten, ohne das dabei logische Verknüpfungsprobleme auftreten. Dieses Modell soll sich per Knopfdruck automatisch (neu) generieren. Da das Grundmodell einer permanenten Erweiterung unterliegt, können zwischen den Aktualisierungsvorgängen bestimmte Informationen dem Benutzer falsch angezeigt werden. Durch eine gewisse Synchronisierung soll das Plugin in der Lage sein, solche Fehler aufzudecken und dementsprechend anzuzeigen. Nach einer weiteren Aktualisierung würden diese Probleme dann im Modell behoben werden.