Institut für Computerlinguistik
Universität Zürich

Wintersemester 02/03

Seminar: „Aspekte der Wissensrepräsentation in der Computerlinguistik“

Dr. Kai-Uwe Carstensen

DAS SEMANTIC WEB

MOTIVATION, KONZEPTE UND SYNTAX

Christos Bräunle
Hügelstrasse 25
8002 Zürich
Email: c.bra@freesurf.ch

Abgabetermin: 06.01 2003

Inhalt:

Das Semantic Web

1. Einleitung

Das Semantic Web (SW) ist eine Vision von Tim Berners Lee[1] und als Erweiterung des bestehenden World Wide Webs (WWW) gedacht. Es handelt sich also nicht um eine neue Netzwerktechnologie, sondern um eine Weiterentwicklung des WWW.

Was soll sich am WWW weiterentwickeln?
Das WWW ist eigentlich eine riesige Datenbank, deren Daten aber bisher wenig strukturiert sind. Wenn man die am meisten verbreitete Sprache für das WWW betrachtet, nämlich die html Syntax, so fällt auf, dass diese Sprache eigentlich nur für die Gestaltung einer Webseite geeignet ist, aber nur sehr beschränkte Mittel zur Beschreibung von ihrem Inhalt bietet. So kann man einer Webseite in html eigentlich nur mit sogenannten Metatags stichwortartige Inhaltsangaben beifügen, die den Inhalt der Seite einigermassen umschreiben können.

Um eine so unterschiedlich strukturierte Datenbank wie das WWW zu beschreiben, wird nun auf Methoden der künstlichen Intelligenz (KI) und der Computerlinguistik (CL) zurückgegriffen. In beiden Disziplinen wird die Modellierung einer Welt mittels einer Ontologie versucht. In einer Ontologie werden die Dinge einer darzustellenden Welt aufgeführt und derart in Relation zueinander gebracht, dass sie ein Modell abgeben, in dem Aussagen gemacht werden können, die den Realitäten der modellierten Welt entsprechen. Das Konzept der Ontologie, das ursprünglich aus der Philosophie kommt, hat nun auch im WWW Eingang gefunden. Man hat entdeckt, dass man eigentlich genau so etwas bräuchte, um eine effiziente Suche im WWW zu ermöglichen.

Die Kernidee für das SW ist, dass nicht eine einzelne, das ganze Universum des WWW erfassenden Ontologie, sondern viele, domänenspezifische Ontologien das WWW langsam zu einer strukturierten Datenbank machen. In einer Ontologie, im Sinne des SW, werden die Metadaten der Datenbank strukturiert. Was das genau heisst, soll im Laufe dieser Arbeit noch klar werden. Während die verschiedenen Ontologien sehr unterschiedliche Konzepte und

[1] http://www.w3.org/People/Berners-Lee/ die Homepage von Tim Berners-Lee für weitere Informationen zu seiner Person.

Themen behandeln, sind sie über eine einheitliche Syntax definiert, was sie miteinander verbindet. Virtuelle Agenten sollen dann bei einer Anfrage, aus der jeweils relevanten Ontologie die nötigen Informationen gewinnen, um die Anfrage richtig zu interpretieren. Wie das genau geschehen soll, wird momentan noch nicht ausführlich behandelt. Vielmehr fokussiert das W3C, das Initiant des SW Projektes ist, die Diskussion auf dessen technische Umsetzung. So wird das Konzept domänenspezifischer Ontologien vor allem in Bezug auf deren syntaktische Realisation hin besprochen.

Kardinalziel dieser Arbeit ist es daher, die Resource Description Framework (RDF) Syntax für das SW zu erläutern, wie sie vom World Wide Web Konsortium (W3C)[2] vorgeschlagen wird und dabei den Fokus auf die Realisierung einer Ontologie in RDF für das WWW zu richten. Weil das SW vor allem von Informatikern entwickelt wird, gibt es relativ wenig allgemeintheoretische Arbeiten zu diesem Thema. Um nun auf die zugrunde liegenden Konzepte zu kommen, müssen wir den Umweg über die RDF Syntax nehmen, obwohl es im Grunde bei einer Ontologie immer um Wissensrepräsentation geht und die damit verbundenen Probleme nicht primär mit der Syntax zusammenhängen, in der die Ontologie dann tatsächlich realisiert wird.

Im zweiten Kapitel möchte ich noch einmal auf den Sinn und Zweck eines SW's zu sprechen kommen und die Grundlagen einer Wissensrepräsentation für das WWW erläutern. Es stellt sich nämlich bei jeder Ontologie die Frage, welcher Art die Dinge sind, über die wir etwas aussagen wollen. Weiter möchte ich in diesem Kapitel auch kurz den Aufbau des SW betrachten.

Beim Konzipieren einer Wissensrepräsentation bedient man sich häufig einer syntaxunabhängigen Darstellungsweise, die mittels Knoten und Kanten die Aussagen darstellt, die dann implementiert werden sollen. Auch für das SW existiert so eine Darstellungsform: das RDF data model. In Kapitel drei werde ich dieses Modell vorstellen und aufbauend darauf anschliessend auf die RDF Syntax zu sprechen kommen. Ich möchte eine vereinfachte Version von RDF verwenden und damit aufzeigen, wie prinzipiell eine Wissensrepräsentation in RDF aussehen könnte.

[2] http://www.w3.org ist das Portal des W3C. Hier findet man Links und Informationen zu allen ihren Aktivitäten.

Da in Kapitel drei die RDF Syntax noch relativ trocken und sehr theoretisch besprochen wird, werde ich in Kapitel vier eine Miniontologie entwerfen, die zeigen soll, wie eine konkrete Anwendung von RDF im WWW aussehen könnte. Dabei werde ich die in Kapitel 3 eingeführten Komponenten verwenden. Obwohl das Beispiel einen vereinfachten Fall darstellt, sollte damit aber doch die prinzipielle Vorgehensweise beim Erstellen einer Ontologie und ihrer Verknüpfung mit Aussagen auf einer Webseite verdeutlicht werden. Das SW steckt noch in den Kinderschuhen und natürlich sind längst nicht alle Probleme gelöst. In Kapitel fünf werde ich einige dieser Probleme ansprechen und auch einige Kritikpunkte aufführen und mit einem Fazit in Kapitel sechs meine Arbeit abschliessen.

2. Grundkonzepte und Architektur des Semantic Web

Wie schon gesagt, ist es beim heutigen Stand des WWW nur möglich, über Stichworte Informationen zu suchen.[3] Der Grund dafür liegt u.a. in der html Syntax, die fast keine Möglichkeiten bietet, den Inhalt einer Seite relational zu beschreiben.
Wenn wir zum Beispiel eine Webseite zu einem Forschungsprojekt betrachten, so möchte ein Besucher der Seite vielleicht wissen, welche Personen an diesem Projekt mitarbeiten, unter wessen Federführung das Projekt geleitet wird, welche Firmen das Projekt unterstützen und vieles Anderes mehr. Ist eine solche Seite in html codiert, so hat der User keine Möglichkeit direkt auf diese Daten zuzugreifen. Er muss die Seite, wenn nicht gar die ganze Domain, nach dieser Information durchsuchen und zwar „von Hand“.
Viel angenehmer wäre es da, wenn beim Suchen einer Information im WWW das Resultat der Suchanfrage nicht eine meist riesige Anzahl von Webseiten wäre, die man dann nach der gewünschten Information durchsuchen muss, sondern einfach die Antwort. Das wäre eigentlich auch nur zu erwarten, wenn man bedenkt, dass praktisch jede Information, die man sucht, irgendwo in dieser riesigen Datenbank namens WWW vorhanden ist. Das Problem ist nur, dass man diese Antwort nicht automatisch extrahieren kann, weil die Informationen im WWW bisher praktisch nicht semantisch strukturiert sind.

Um nun eine automatische Antwortextraktion zu ermöglichen, benötigen wir Daten über die gesuchten Daten, sog. Metadaten. Metadaten erleichtern die Suche in einer Datenbank, indem

[3] Es gibt natürlich auch noch die Möglichkeit, über Directories, Linksammlungen oder Weblogs nach Informationen zu suchen. Das ist aber keine maschinelle Suche.

sie Informationen über die Daten der Datenbank zur Verfügung stellen. Solche Informationen können im einfachsten Fall Angaben zum Typ eines Datensatzes sein. Wenn ich in meiner Telephonagenda meine Einträge nach Name, Adresse und Telephonnummer geordnet habe, sind diese Angaben (also „Name“, „Adresse“ , usw.) eigentlich schon Metadaten.

Metadaten können aber weit komplexere Informationen zur Verfügung stellen. Um nochmals auf das obige Beispiel mit den Mitarbeitern eines Forschungsprojektes zurückzukommen, so liesse sich mittels einer Prädikation, wie sie in der Logik verwendet wird, aussagen, dass „Ora Lassila“ Mitarbeiterin am „Semantic Web Projekt“ ist, z.B. indem ich schreibe:

MITARBEITER(„Ora Lassila“, „Semantic Web Projekt“)

Damit haben wir aber für die Maschine noch nicht viel an Information dazu gewonnen. Wir wissen jetzt zwar, dass zwischen der Buchstabenfolge „Ora Lassila“ und „Semantic Web Projekt“ die Relation MITARBEITER gilt, die Maschine aber kann mit der Buchstabenfolge MITARBEITER als solche relativ wenig anfangen. Daher müssen wir auch noch angeben, von welchem Typ die Argumente sind, die die Relation MITARBEITER hat. So handelt es sich bei „Ora Lassila“ um eine PERSON und bei „Semantic Web Projekt“ um ein PROJEKT. Die Begriffe PERSON und PROJEKT können mittels Ober- und Unterbegriffshierarchisierung weiter spezifiziert werden. Indem ich das mache, beginne ich im Prinzip mit der Realisation einer Ontologie.
Bei der Spezifikation einer Ontologie, geht es darum, die minimalen Bedingungen zu charakterisieren, unter denen eine Aussage über die von der Ontologie beschriebene Welt wahr ist. Eine mögliche Einschränkung wäre, die Art der Argumente zu charakterisieren, die ein bestimmtes Prädikat überhaupt haben kann. Bei der Relation MITARBEITER liesse sich z.B. sagen, dass die erste Argumentsposition ein Objekt aus der Klasse LEBEWESEN oder einer Unterklasse von dieser sein muss, während die zweite Argumentsposition ein Objekt aus einer der Klassen PROJEKT, FIRMA, INSTITUTION, etc. sein muss. Mit derselben Hierarchie könnte man dann auch herausfinden, welche HUNDE MITARBEITER bei der STADTPOLIZEI ZüRICH sind[4].
Eine Hierarchisierung geht von einem einzelnen Begriff aus, dem alle weiteren Begriffe untergeordnet sind. Typischerweise bezeichnet man diesen obersten Begriff mit ENTITÄT oder einem ähnlich neutralen Begriff. Während man in der wirklichen Welt von real existierenden

[4] Sofern die Stadtpolizei die Daten über ihre Hunde in RDF strukturiert hat.

Dingen ausgehen kann, hat man es im WWW aber mit virtuellen Objekten zu tun. Der allgemeinste Begriff für ein Objekt im WWW ist in der Terminologie der Informatiker die Ressource.
Wie können diese Ressourcen aber fassbar gemacht werden?

2.1 URI's und Literals: Die Grundkonstituenten des Webs

In der Welt des WWW wird alles durch einen Uniform Resource Identifier (URI)[5] bezeichnet. Ein URI bezeichnet ein Objekt im WWW eindeutig. Es steht jedem frei, wann immer er will ein neues URI zu kreieren. Ein URI kann der Pfad zu einer Webseite sein, z.B. http://www.66-99.ch, oder eine Emailadresse, wie c.bra@freesurf.ch. Erwähnenswert ist, dass diese URI's eindeutig sind, was ein nicht zu unterschätzender Vorteil ist.
Die Objekte, über die wir etwas Aussagen wollen, sind URI's. Neben den URI's können wir aber auch literals verwenden, das heisst, wir können auch Buchstabenfolgen, also Zeichenketten verwenden, die allerdings wie ein Stichwort behandelt werden. Wenn wir etwas über eine Seite im Web aussagen wollen, dann haben wir auch einen eindeutigen Namen für diese Seite, nämlich deren URL, was einfach eine spezielle Form eines URI's ist. Da man jederzeit ein neues URI kreieren kann, ist es auch Möglich, einzelne Elemente einer Seite mit einem URI zu identifizieren. Um nun den Macher einer Seite zu bezeichnen, können wir entweder seinen Namen als Zeichenkette, oder auch bspw. seine Emailadresse verwenden. Wir können aber auch eine Zeichenkette auf der Webseite mit einem URI beschriften und dieses URI als Namen verwenden.

2.2 Der Aufbau des SW's

Wie aus Abb.1 ersichtlich, soll das SW aus verschiedenen Schichten aufgebaut werden, die in ihrer Komplexität von Stufe zu Stufe zunehmen.
Auf der Untersten Stufe befinden sich die URI's und der Unicode[6]. URI's haben wir bereits kennen gelernt und Unicode ist einfach eine Standardisierung, um Zeichenketten zu schrei-

[5] http://www.w3.org/Addressing/ hier findet man die formalen Kriterien zur Definition von URI's.
[6] http://www.unicode.org/ alles zum Thema Unicode.

ben. Gleich darüber ist die eXtensible Markup Language (XML)[7], die eine Erweiterung der html Syntax ist.

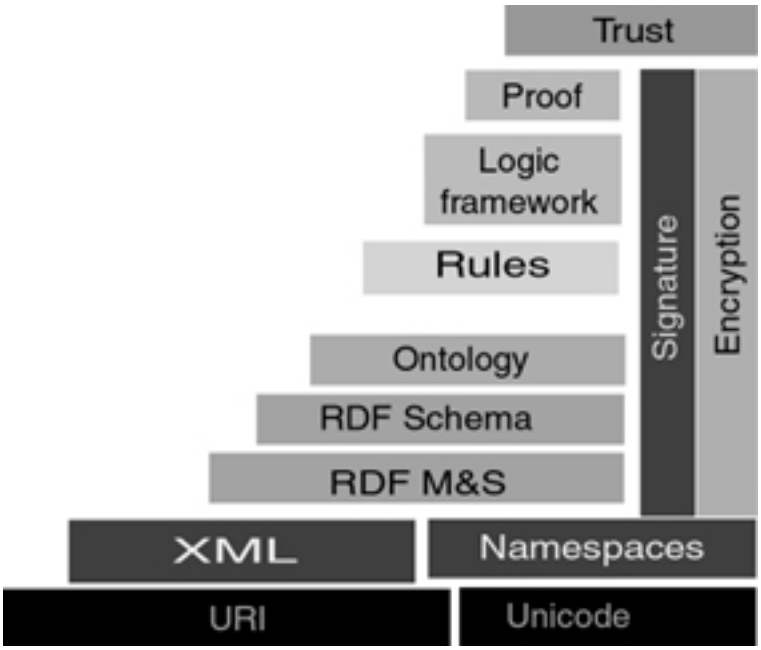

Abb.1

Dann folgen RDF, RDF Schema und die Ontologie. Wie wir später noch ausführlicher sehen werden, kann man in RDF nicht nur Aussagen machen, sondern auch die Ontologie wird in RDF realisiert. Auf diese Stufen konzentriert sich auch der momentane Stand der Entwicklung des SW. Die nächsten Stufen sind noch Zukunftsmusik. Die verschiedenen Ontologien können durch Regeln noch mächtiger gemacht werden, da Regeln erlauben würden, Wissen automatisch in eine Ontologie einzufügen. Dazu braucht es aber auch einen logische Konzeptualisierung und einen Beweismechanismus.

Als letztes kommt noch die digitale Signatur und das Web of Trust ins Spiel. Ein Grundkonzept des Webs ist es, dass Jeder Alles sagen kann. Niemand kann mich daran hindern, auch falsche Informationen ins Netz zu stellen. Mit der digitalen Signatur kann nun ein Benutzer die Sicherheit haben, dass die Informationen, die er gefunden hat, aus einer vertrauenswürdigen Quelle stammen. Das Web of Trust dagegen funktioniert nach dem Schneeballprinzip. Ich wähle einige andere User aus, denen ich besonders vertraue. Diese haben die Möglichkeit eine Seite nach ihrem qualitativen Gehalt zu klassifizieren und ich kann mich dann an ihren Angaben orientieren.

[7] http://www.w3.org/XML/ alles zum Thema XML.

Wie schon gesagt, sind diese Dinge aber noch nicht realisiert. Meine Arbeit konzentriert sich daher auch auf die untere Hälfte dieses Schichtenmodells.

3. Wie soll das SW realisiert werden?

Wir haben gesehen, dass das WWW keine relationalen Informationen über seine Inhalte zur Verfügung stellt. Gleichzeitig haben wir aber die Möglichkeit, eindeutig auf die einzelnen Elemente des WWW zuzugreifen, da sie mit einem URI identifiziert werden können. D.h. wir können alles, worüber wir im SW etwas aussagen wollen, entweder durch ein URI bezeichnen oder als eine Zeichenkette behandeln. Mit einem URI haben wir einen direkten Verweis auf einen Referenten, bzw. auf eine Ressource.
Ich habe gezeigt, dass man Metadaten braucht, um eine Ressource leichter auffindbar zu machen. Dann habe ich den Aufbau des SW vorgestellt, aber noch nichts darüber gesagt, wie das SW nun tatsächlich realisiert werden soll.

In diesem Kapitel möchte ich nochmals darauf zurückkommen, wie heute Webseiten annotiert werden und wir werden sehen, dass diese Informationen nicht ausreichen, um eine Antwort automatisch aus dem WWW zu extrahieren.
Anschliessend stelle ich das RDF data model vor. Das ist ein Darstellungsmittel, das auch in anderen Wissensrepräsentationen verwendet wird. Es dient dazu, die Relationen, die man implementieren will, graphisch darzustellen. Dann gehen wir aber mal direkt zur Sache, d.h. zur RDF Syntax. Besonders wichtig wird sein, zu sehen, dass in RDF nicht nur Aussagen gemacht werden, sondern dass auch die Ontologien in RDF spezifiziert werden. Um das Konzept der verschiedenen, domänenspezifischen Ontologien zu verstehen, kommt dem XML namespace eine entscheidende Rolle zu. Daher werde ich in diesem Kapitel auch darauf genauer eingehen.

Nun aber zum Problem der Metadaten beim heutigen Stand des WWW. Wie schon gesagt, sind die Möglichkeiten, im html Quellcode einer Webseite Informationen über deren Inhalt zu verankern, relativ beschränkt. So könnten die Metadaten zu dieser Arbeit etwa so aussehen:

```
<head>
     <meta name="title" content="Motivation, Konzepte und
Syntax für das Semantic Web">
```

```
    <meta name="creator" content="Christos Bräunle">
    <meta name="subject" content="Eine Einführung in das
Semantic Web">
    <meta name="description" content="Das Semantic Web">
    <meta name="type" content="Seminararbeit">
    <meta name="date" content="2002-09-11">
    <meta name="language" content="de">
    <meta name="robots" content="all">
</head>
```

Diese Meta-Angaben können ein Dokument zwar charakterisieren, bieten aber keinerlei relationale Informationen. Die Metadaten sind Stichworte, die einem Typ (im Beispiel durch Fettdruck hervorgehoben) zugeordnet worden sind, und dieser Typ bezieht sich immer auf das aktuelle Dokument als ganzes. Auch die Anzahl der Typen, die man verwenden kann, ist auf eine vordefinierte Anzahl beschränkt. Auf eine Suchanfrage wie: „Was ist das Semantic Web" oder „Wer ist der Erfinder des Semantic Web" wird man also niemals eine Antwort erhalten, sonder immer eine Webseite, in der die Antwort möglicherweise enthalten ist.

3.1 Resource Description Framework (RDF)

Um nun ein mächtigeres Werkzeug zu bekommen, hat das W3C Anfang der Jahrtausendwende das Resource Description Framework (RDF) entwickelt, das auf der eXtensible Markup Language (XML) aufbaut. XML ist das Werkzeug, das uns erlaubt beliebige Typen, in XML nennt man sie Tags, zu definieren und RDF bietet die Möglichkeit Relationen darzustellen.

RDF ist im Prinzip einfach die Syntax zur technischen Umsetzung einer Wissensrepräsentation für das WWW, die es uns ermöglicht, semantische Informationen im Quellcode der Webseite zu verankern. Zudem kann RDF von allen gängigen Browsern interpretiert werden, weil es eben auf XML basiert, was schon eine Standardsprache ist, die sich für das WWW einigermassen etabliert hat.

Um RDF einsetzen zu können, habe ich zuerst gezeigt, was für Informationen wir darstellen wollen. Nun wird es darum gehen, auf welche Weisen wir diese Informationen darstellen.
Da es sich um eine Wissensrepräsentation handelt, sind auch die Strukturen sehr ähnlich wie in anderen Disziplinen, die sich mit diesem Thema befassen. So geht es auch im SW darum, Aussagen über Dinge, - in diesem Fall Ressourcen -, zu machen.

Um eine Syntaxneutrale Darstellungsform zu haben, verwendet das W3C die in der Wissensrepräsentation bereits bekannte Visualisierung von Aussagen mittels Knoten und Kanten. Hier heisst es einfach das RDF data model.

3.2 Das RDF data model

Um nun über eine Ressource eine Aussage, ein sog. Statement zu machen, können in RDF URI's oder Zeichenketten zu Tripel zusammengefügt werden. Die Aussage, dass ich eine bestimmte Page gemacht habe, könnte bspw. so aussehen:

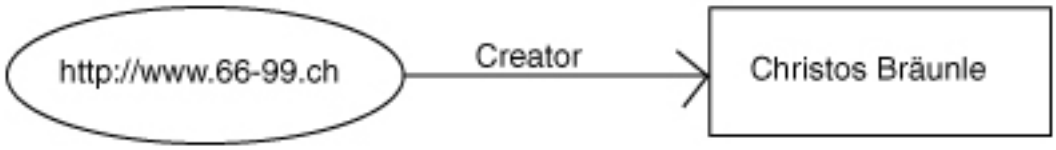

Abb.2

Der Vorteil solcher Darstellungen ist, dass sie Syntax neutral sind. Sie entsprechen dem RDF data model[8] und werden dazu verwendet, die Äquivalenz von Bedeutungen festzustellen. Zwei RDF Ausdrücke sind Äquivalent, wenn und nur wenn ihr data model das selbe ist. Diese Definition erlaubt Variationen in der Syntax von RDF, ohne die Bedeutung der Ausdrücke zu verändern.
Ein data model besteht aus folgenden Elementen:

Resources — Alle Dinge, die in RDF Ausdrücken beschrieben werden können, heissen resource. Eine resource kann ein Dokument sein, wie http//:www.66-99.ch/index.html, aber auch die gesamte Website, also www.google.com. Sie kann auch ein spezifisches Element aus dem html oder XML Quellcode sein. Eine resource kann sogar ein Objekt sein, das nicht direkt über das WWW ver-

[8] Siehe dazu Lassila 1999

fügbar ist, wie ein gedrucktes Buch bspw. Resources werden aber immer durch URI's bezeichnet.

Properties Eine property ist ein spezifischer Aspekt, eine Charakteristik, ein Attribut oder eine Relation, die eine resource beschreibt. Jede property hat eine spezifische Bedeutung, die über die erlaubten Werte, die sie haben kann und über die resources, die sie beschreiben kann definiert ist, sowie dadurch, in welchem Verhältnis sie zu anderen properties steht.

Statement Eine resource mit einer property und einem Wert dieser property für die resource ist ein RDF statment. Diese drei Teile entsprechen dem Subjekt, dem Prädikat und dem Objekt. Das Objekt kann eine weitere resource oder ein literal sein.[9]

Beim obigen Beispiel ist das Subjekt http//:www.66-99.ch (eine resource), das Prädikat ist `Creator` (eine property) und das Objekt ist „Christos Bräunle" (eine literal). In der RDF Terminologie entsprechen resources der Menge aller Objekte über die wir sprechen wollen. Properties entsprechen den Prädikaten, wie wir sie in der Logik verwenden, und literals sind einfach Zeichenketten, die nicht weiter interpretiert werden können. Zudem bietet RDF die Möglichkeit, auch sogenannte blank nodes zu verwenden. Das sind Knoten die noch keinen Referenten haben. Solche blank nodes werden wie eine existentielle Quantifikation interpretiert.

Der Einfachheit halber werde ich im Weiteren von resource als Ressource, von literal als Zeichenkette, von property als Prädikat und von Statment als Aussage sprechen.

Im Diagramm werden Ressourcen durch Ovale, Prädikate durch einen gerichteten Pfeil und Zeichenketten durch ein Rechteck dargestellt.

[9] Quelle Lassila 1999

3.3 Basic RDF Syntax

Das RDF data model bietet ein abstraktes Hilfsmittel, um Aussagen darzustellen. Um solche Aussagen aber tatsächlich im Web verwenden zu können, braucht es eine Syntax, damit diese Metadaten auch erstellt und ausgetauscht werden können. Zu diesem Zweck verwendet RDF die XML Syntax und den XML namspace, in dem ein Schema realisiert wird, mit dem ein Prädikat definiert werden kann. Dazu später mehr.

Wenn man in RDF Aussagen bildet, so möchte man meistens mehrere Dinge über eine Ressource aussagen. In RDF kann das gemacht werden, indem man mehrere Aussagen über eine Ressource zusammenfasst. Das `Description` Element nennt in einem `about` Attribut die Ressource, über die alle Aussagen gemacht werden. Wenn die Ressource noch nicht existiert (d.h. wenn sie kein URI hat) kann man ein `ID` Attribut verwenden.

Die vereinfachte Serialisierung einer RDF Syntax sieht folgendermassen aus:

```
[1]  RDF                ::= ['<rdf:RDF>']  description*
                            ['</rdf:RDF>']
[2]  description        ::= '<rdf:Description' idAboutAttr?'>'
                            propertyElt* '</rdf:Description>'
[3]  idAboutAttr        ::= idAttr | aboutAttr
[4]  aboutAttr          ::= 'about="' URI-Reference '"'
[5]  idAttr             ::= 'ID="' IDSymbol '"'
[6]  propertyElt        ::= '<' propName '>' value
                            '</' propName '>'
                            | '<' propName resourceAttr '/>'
[7]  propName           ::= Qname
[8]  value              ::= description | string
[9]  resourceAttr       ::= 'resource="' URI-Reference '"'
[10] Qname              ::= [NSprefix ':'] name
[11] URI-reference      ::= string, interpreted per [URI]
[12] IDsymbol           ::= (any legal XML name symbol)
[13] name               ::= (any legal XML name symbol)
[14] NSprefix           ::= (any legal XML namspace prefix)
[15] string             ::= (any legel text, with „<“, „>“
                            and „&“ escaped)[10]
```

Das `RDF` Element zeigt einfach an, wo die Begrenzungen des RDF Codes liegen. Da solche Codeblöcke ja in ein html Dokument eingebunden werden, muss man dem Interpreter auch anzeigen, wo der RDF Code beginnt und wo er aufhört. In html, in XML und somit auch in

[10] Quelle Lassila 1999

RDF geschieht dies mit einem Starttag, das durch eckige Klammern gekennzeichnet ist <*rdf*:RDF> und dem Endtag, das einfach noch einen slash „/“ hat </*rdf*:RDF>.

Mit Description hat man, wie schon oben erwähnt, die Möglichkeit, mehrere Aussagen über dieselbe Ressource zu machen. Das Description Element, bietet einfach einen Platz, um diese Ressource zu nennen.

Wenn innerhalb von Description ein about Attribut spezifiziert ist, wird die Ressource, auf die sich die Aussagen beziehen, als URI Referenz interpretiert. Wenn in Description kein about spezifiziert ist, bedeutet dies, dass es sich um eine unbekannte Ressource handelt.
In diesem Fall wird die Ressource durch ein ID Attribut repräsentiert, welches als Platzhalter für die eigentliche, bzw. noch zu findende Ressource steht.

Das ID Attribut signalisiert die Generierung einer neuen Ressource, während das about Attribut auf eine bestehende Ressource referiert.
ID und about Attribut können nicht in ein und derselben Aussage im Description Element vorkommen. Was im RDF data model als blank nodes (also leere Knoten) bezeichnet wurde, entspricht diesem ID Attribut, das damit die existenzielle Quantifikation möglich macht. Ein Knoten ohne Referent zeigt an, dass es einen Referenten gibt, der aber erst noch gefunden werden muss.

Innerhalb von propertyElt werden der Ressource der Aussage die Eigenschaften und deren Werte zugewiesen. Jedes propertyElt fügt der Ressource eine neue Kante im Modell und den entsprechenden Wert hinzu. Zu beachten ist die Möglichkeit der Schachtelung innerhalb von propertyElt, denn der Wert der Eigenschaft kann wiederum eine Description sein. PropertyElt entspricht also dem Prädikat inklusive dem dazugehörigen Wert.

Prädikate müssen mit einem Schema assoziiert sein. Das geschieht durch die Kennzeichnung eines Elementes mit einem namespace Präfix, welches das Prädikat eindeutig mit einem RDF Schema assoziiert.

Die Aussage aus Abb.2

Christos Bräunle ist der Urheber der Ressource http://www.66-99.ch
würde in RDF etwa folgendermassen aussehen:

```
<rdf:RDF>
     <rdf:Description about="http://www.66-99.ch">
          <s:Creator>Christos Bräunle</s:Creator>
     </rdf:Description>
</rdf:RDF>
```

In diesem Beispiel referiert‘*s*‘ auf einen bestimmten namespace, der vom Autor dieser Aussage noch z.B wie folgt spezifiziert werden muss:

```
xmlns:s="http//:description.org/schema/"
```

Die namespace Deklaration kann innerhalb der `Description` gemacht werden, kann aber auch auf einen einzelnen `propertyElt` Ausdruck angewendet werden.

Was aber ist ein namespace und wozu braucht es ihn?

3.4 RDF Schema und XML namspace

Wenn ich eine Aussage mache, so weise ich einer Ressource eine Eigenschaft und den entsprechenden Wert dieser Eigenschaft zu. Der Name der Prädikation sagt aber nichts über dessen Eigenschaft aus. Wenn ich ein Prädikat wie `Creator` verwende, hat ein Interpreter noch keinerlei semantische Informationen über dessen Eigenschaften. Wir haben gesehen, dass in einer Ontologie solche semantischen Merkmale spezifiziert werden können.

RDF Schema (RDFS) bietet nun die Werkzeuge, die die Beschreibung von Ressourcen und Prädikationen erlauben. Indem RDFS in einem namespace definiert ist, hat das Schema ein URI, mit dem es für alle verwendbar wird, denn ein namespace ist im Prinzip einfach eine Internetseite. Ich brauche nur mein RDF Dokument mit diesem namespace zu verlinken, um die Relationen, die dort definiert sind, verwenden zu können.

Das grundlegenste Konzept bei der Definition einer Ontologie in RDF ist die Einteilung von Ressourcen in Klassen. Ist eine Ressource Element einer Klasse, so sagt man, dass sie eine

Instanz dieser Klasse ist. Um eine Ressource einer Klasse zuzuweisen, bietet RDFS das Prädikat `rdf:type`.
Klassen wiederum können hierarchisch organisiert werden. Eine Klasse `hund` könnte Unterklasse der Klasse `säugetier` sein, die ihrerseits wiederum Unterklasse von `tier` ist. Um solche Hierarchien zu modellieren, bietet RDFS das Prädikat `rdfs:subClassOf`.

Prädikate werde dadurch definiert, dass sie bestimmte Klassen von Ressourcen als Argumente haben können. Mit `rdfs:domain` und `rdfs:range` kann man die Klassen, die auf die Argumente eines Prädikats angewendet werden dürfen, einschränken. So könnte ich das Prädikat `author` so definieren, dass seine `domain` ein Element aus der Klasse `buch` sein muss und seine `range` ein Element aus der Klasse `literal`.

Wie gesagt, ist die Einteilung in Klassen das grundlegendste Bauelement für eine Ontologie in RDF. Dem entsprechend ist auch RDFS nach diesem Bauprinzip organisiert. Abb.3 gibt eine Übersicht von der Einteilung der Ressourcen in die entsprechenden Klassen.

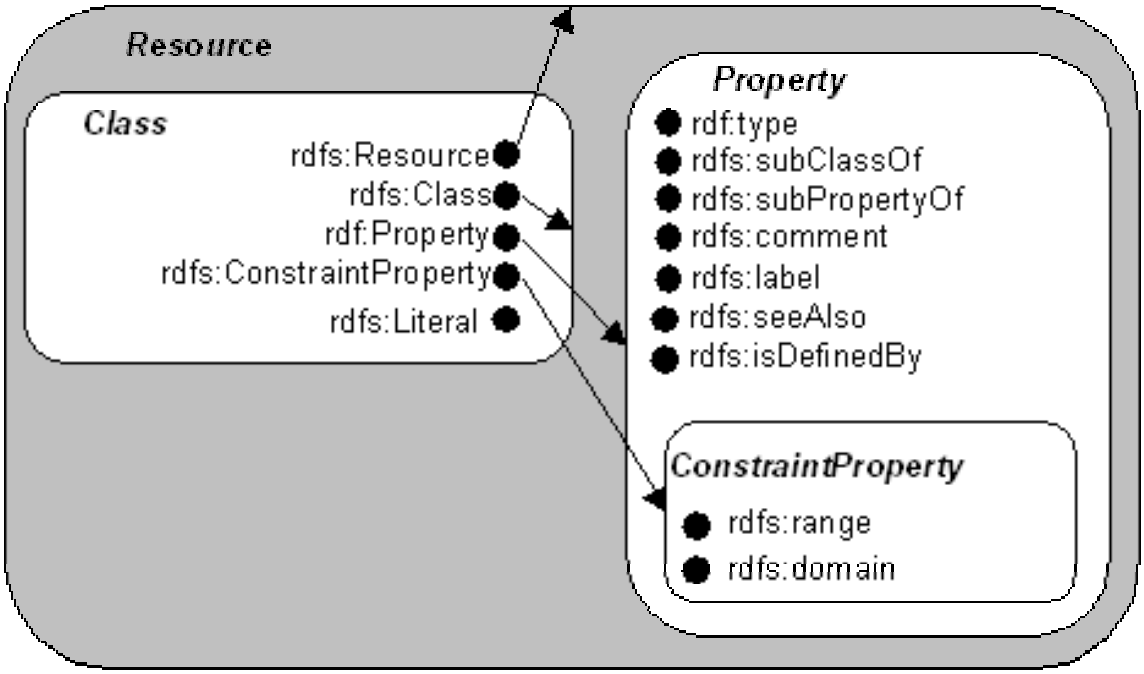

Abb.3

Die Klassen sind als gerundete Rechtecke symbolisiert, die Instanzen dieser Klassen als Punkte. Wie aus der Graphik ersichtlich wird, ist bspw. `rdfs:resource` Instanz der Klasse

Resource. D.h. eine Klasse ist eine Instanz der Klasse selber. Genau gleich verhält es sich mit `rdfs:class`, `rdf:property` und `rdfs:constraintProperty`.
In der Prädikatenlogik darf aber eine Klasse nicht Instanz ihrer selbst sein. Dieser Punkt ist auch schon an RDFS kritisiert worden[11], denn eine solche Zuweisung kann zu einem Halteproblem führen, was die Enthalten-Seins Relation unentscheidbar macht. Von der anderen Seite wird dagegen argumentiert, dass ein solcher Fall zwar theoretisch möglich sei, aber praktisch nie eintrete. Wie auch immer: Eine solche Zuweisung ist in der Prädikatenlogik nicht legal und sollte daher eigentlich vermieden werden.

4. Ein konkretes Beispiel

Ich werde nun an einem einfachen Beispiel zeigen, wie man für einen bestimmten Bereich Aussagen formuliert und diese mit einer Ontologie verknüpft. Dabei werden die in Kapitel drei vorgestellten Konzepte zum tragen kommen. Die grundsätzliche Vorgehensweise ist, dass man eine Anzahl Aussagen definiert und diese mit einer Ontologie verbindet.

Wie gesagt, stellt der namspace von RDFS die Werkzeuge zur Verfügung, um eine Ontologie zu definieren. Dabei muss im Auge behalten werden, dass es sich immer um eine „Miniontologie“ handelt, d.h. eine Ontologie, die nur den für eine Domäne relevanten Bereich abdeckt. Wenn ich bspw. ein Autohändler bin und meine Webseite semantisch anreichern will, so mache ich einerseits einmal eine gewisse Anzahl von Aussagen über meine Produkte. Diese Aussagen sind im Quellcode der Seite eingebunden. Um sie semantisch interpretierbar zu machen, kreiere ich zusätzlich einen namespace. In diesem definiere ich eine „Miniontologie“, indem ich die Ressourcen, die ich verwende, in Klassen und Unterklassen einteile, und den von mir verwendeten Prädikaten diejenigen Klassen zuweise, aus denen sie ihre jeweiligen Argumente beziehen können. Diese „Miniontologie“ wird, wie gesagt, mit den Prädikaten aus dem namespace von RDFS definiert.

[11] Oberle 2001 gibt unter Kapitel 2.4 einen Lösungsansatz für dieses Problem. Die Relationen werden auf vier Metaebenen verteilt. Durch diese Aufteilung lassen sich die Prädikate domain, range usw. auf vier verschiedene Layer aufteilen. Zur Definition in einem Layer werden die Prädikate des darüber liegenden Layers verwendet.

4.1 Definition der Ontologie

Beginnen wir zuerst mit der Definition der Ontologie. Ich möchte auf meiner Webseite Aussagen über verschiedene Typen von Fahrzeugen machen. Ich habe Vans, Minivans, Trucks und Passenger Vehicles. Ich möchte Aussagen über ihren Preis, ihre Marke und über die Grösse ihres Kofferraumes machen.
Bei den Fahrzeugtypen handelt es sich um Klassen, denen dann in den Aussagen bestimmte Objekte zugewiesen werden können. Wir haben gesehen, dass Klassen in Ober- und Unterbegriffe eingeteilt werden. Ein möglicher Oberbegriff für meine Fahrzeuge wäre bspw. Motor Vehicle.
In RDF könnte eine solche Einteilung folgendermassen gestaltet werden:

```
<rdf:RDF xml:lang="en"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
```

Mit dem Verweis auf diese beiden namespaces, kann ich die Prädikate und Klassen verwenden, die in RDF und RDFS definiert sind.

```
<rdf:Description ID="MotorVehicle">
     <rdf:type resource="http://www.w3.org/2000/01/rdf-
     schema#Class"/>
```

Mir dem Prädikat `rdf:type` mache ich `MotorVehicle` zu einer Klasse, indem ich sie der Klassendefinition von RDFS zuweise. Da diese Klasse neu ist und daher noch nicht existiert, was heisst, dass sie kein URI hat, muss sie mit einem ID Attribut neu generiert werden.

```
 <rdfs:subClassOf
     rdf:resource="http://www.w3.org/2000/01/rdf-
     schema#Resource"/>
</rdf:Description>
```

Da eine Klasse auch eine Ressource ist, ist auch MotorVehicle eine Ressource. Die Eigenschaft eine Ressource zu sein, wird auch mit `rdfs:subClassOf` realisiert.

```
<rdf:Description ID="PassengerVehicle">
     <rdf:type resource="http://www.w3.org/2000/01/rdf-
     schema#Class"/>
     <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

`PassengerVehicle` ist ebenfalls eine Klasse und Unterklasse von `MotorVehicle`. Da die `subClassOf` Relation transitiv ist, brauche ich nicht mehr auszusagen, dass `PassengerVehicle` auch Unterklasse von `resource` ist.

```
<rdf:Description ID="Truck">
     <rdf:type resource="http://www.w3.org/2000/01/rdf-
     schema#Class"/>
     <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="Van">
     <rdf:type resource="http://www.w3.org/2000/01/rdf-
     schema#Class"/>
     <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="MiniVan">
     <rdf:type resource="http://www.w3.org/2000/01/rdf-
     schema#Class"/>
     <rdfs:subClassOf rdf:resource="#Van"/>
     <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>

</rdf:RDF>
```

Im RDF data model sieht diese Klassifikation dann wie in Abb. 4 aus:

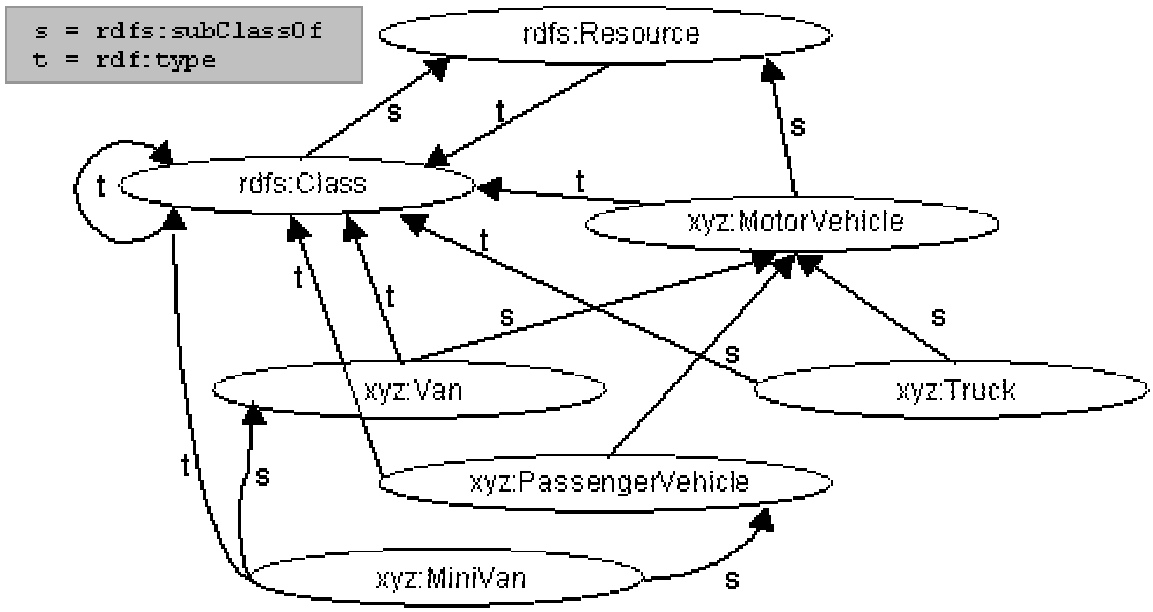

Abb.4

So viel also zur Definition der Begriffe. Ich habe über alle Begriffe ausgesagt, dass sie eine Klasse sind und habe alle diese Klassen der Klasse der Ressourcen untergeordnet und noch eine Feinunterteilung der verschiedenen Klassen gemacht.

Nun wollen wir aber auch Prädikate verwenden, um Aussagen über den Preis, die Marke und den Hubraum des Kofferraumes der verschiedenen Objekte zu machen. Wie schon gesagt, werden Prädikate dadurch definiert, dass ich angebe, welche Klasse von Objekten sie als Argumente haben können. So gilt beim Prädikat `price`, dass das erste Argument aus der Klasse `MotorVehicle` sein muss, während der Wert der Aussage ein Objekt aus der Klasse `Number` sein muss usw.
In RDF würde diese Spezifikation dann wie folgt aussehen:

```
<rdf:RDF xml:lang="en"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
```

Wiederum müssen wir angeben, auf welchen Namespace wir uns bei der Definition unserer Prädikate beziehen.

```
<rdf:Description ID="price">
```

Der Name meines Prädikats für den Preis eines Fahrzeuges.

```
 <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
```

`price` ist ein Prädikat und ist folglich vom `rdf:type Property.`

```
 <rdfs:domain rdf:resource="#MotorVehicle"/>
```

Das erste Argument für das Prädikat `price` muss eine Instanz aus der Klasse `MotorVehicle` sein.

```
 <rdfs:range
rdf:resource="http://www.w3.org/2000/03/example/classes#Number
"/>
```

Der Wert der Aussage muss eine Instanz aus der Klasse `Number` sein.

```
</rdf:Description>
```

```
<rdf:Description ID="bootSize">
```

Das Prädikat zur Angabe der Kofferraumgrösse heisst `bootSize`.

```
 <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-
 ns#Property"/>
 <rdfs:domain rdf:resource="#PassengerVehicle"/>
 <rdfs:domain rdf:resource="#Minivan"/>
```

Dieses Prädikat darf nur Instanzen aus den Klassen `PassengerVehicle` und `Minivan` haben. Die Objekte aus den Klassen `truck` und `van` haben keinen Kofferraum.

```
 <rdfs:range
rdf:resource="http://www.w3.org/2000/03/example/classes#Number
"/>
</rdf:Description>

<rdf:Description ID="brandName">
 <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-
 ns#Property"/>
 <rdfs:domain rdf:resource="#MotorVehicle"/>
 <rdfs:range rdf:resource="#Brand"/>
</rdf:Description>
</rdf:RDF>
```

Visualisiert sieht das wiederum folgendermassen aus:

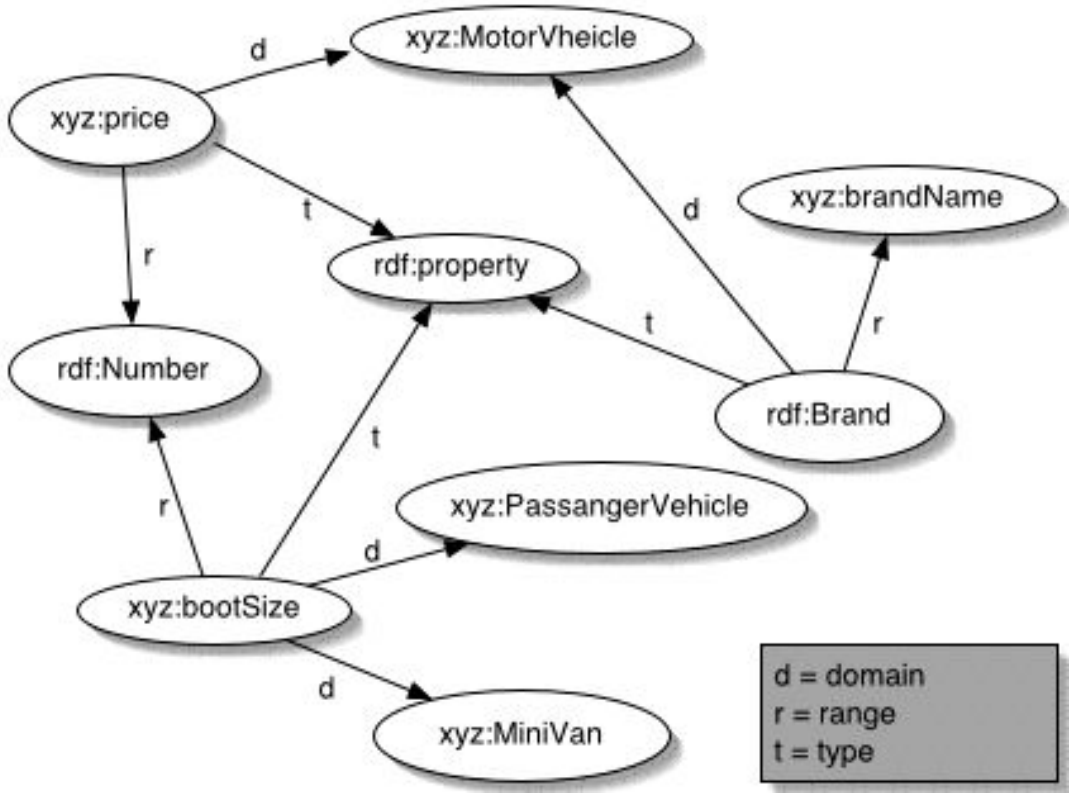

Abb.5

Damit haben wir die Konzepte und Prädikate, die wir verwenden wollen definiert. Aber eine Ontologie ist nutzlos, solange keine Aussagen über die Welt existieren, die sie beschreiben

will. Die Aussagen über die konkreten Preise, Marken und Kofferraumgrössen meines Sortiments, binde ich nun in den Quellcode meiner Webseite ein.

4.2 Formulierung von Aussagen

Meine kleine Ontologie ist doch schon so weit, dass ich einfache Aussagen machen kann. Nehmen wir an, ich habe für alle Wagen, die ich im Sortiment habe eine eigene URL und ich habe für jeden Wagen eine Seite auf meiner Home Page. Damit kann ich dieses URL als Name für den Wagen brauchen den ich genauer beschreiben will. Die Beschreibung meines Mercedes Vito, könnte dann z.B. so Aussehen:

```
<rdf:RDF>
     xmlns:xyz="http//:mycars/schema# "
```

Der namspace meiner Ontologie für Fahrzeuge. Damit werden die zuvor definierten Klassen und Prädikate verwendbar.

```
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

     <rdf:Description about="http://www.mycars.ch/vito.html">
          <rdf:type resource="http//:mycars/schema#MiniVan"/>
          <xyz:#brandName>Mercedes</xyz:#brandName>
          <xyz:#bootSize>80</xyz:#bootSize>
          <xyz:#price>25000</xyz:#price>
     </rdf:Description>
</rdf:RDF>
```

Mit diesen Metadaten kann nun ein Besucher meiner Seite die so verankerten Informationen direkt anfragen. Wenn einmal eine Ontologie definiert ist, die den Bedürfnissen vieler Autohändler entspricht, ist es für sie auch möglich, diese Ontologie untereinander zu teilen. Da die Definition der Ontologie unter einem separaten URL angelegt ist, können auch verschiedene Interessenten diese benutzen. Hat sie erst eine gewisse Akzeptanz erreicht, kann man auch davon ausgehen, dass die Aussagen der verschiedenen Anbieter untereinander konsistent sind.

5. Probleme und Kritikpunkte

Ich habe in meiner Arbeit vor allem den technischen Aspekt einer Wissensrepräsentation für das WWW angesprochen. Ich habe gezeigt, wie man eine Ontologie definieren kann und wie man Aussagen macht. Die Realisation einer Ontologie in einem namspace ist dabei von entscheidender Bedeutung, auch im Bezug auf die Idee, viele verschiedene Ontologien zu entwickeln, die immer nur einen begrenzten Bereich des WWW abdecken. Ich möchte nun aber doch noch einige prinzipielle Probleme ansprechen, die, wie ich glaube, entscheidend für die Entwicklung des SW sind.
Die Vision von Tim Berners Lee, ein durchweg maschinenverständliches WWW zu entwickeln, in dem die Suche nach bestimmten Inhalten völlig der Maschine übergeben werden kann, ist sicherlich noch weit von ihrer Realisation entfernt. Die Literatur die vom W3C publiziert wird, befasst sich mehrheitlich mit syntaktischen und formal-logischen Problemen. Es wäre aber wünschenswert, wenn mehr über prinzipielle Probleme, bei der Entwicklung eines SW gesprochen würde. Der einzige Aspekt, der diesbezüglich einige Beachtung gefunden hat, ist die Tatsache, dass im WWW Jeder Alles sagen kann, und daher ein Konzept für den Umgang mit falscher Information gefunden werden muss.

Doch sind das Probleme, die ein funktionierendes SW voraussetzten. So weit sind wir aber leider noch lange nicht. Bevor solche Probleme gelöst werden können, gilt es noch eine Reihe anderer Schwierigkeiten aus dem Weg zu räumen. So ist die Idee von verschiedenen kleinen Ontologien ein sehr überzeugender Gedanke, doch leider weiss noch kein Mensch, wie diese Ontologien miteinander verbunden werden sollen. Aber selbst das ist nicht das vordergründigste Problem.

Das Erstellen einer Ontologie ist nämlich in komplexeren Fällen nicht so trivial, wie es mein kleines Beispiel vielleicht suggerierte. Es stellt sich die Frage, von wem solche Ontologien definiert werden sollen. Bei einer manuellen Entwicklung der Ontologie, muss diese Aufgabe wahrscheinlich von Spezialisten übernommen werden, denn gerade bei der Definition von Ober- und Unterbegriffshierarchien ist die Konsistenz dieser Hierarchien von grosser Bedeutung. Aber auch die Annotation von Webseiten ist nicht ganz einfach. Selbst wenn dazu Hilfsprogramme entwickelt werden, ist es fraglich, ob das Definieren von Aussagen dem „Laien“ überlassen werden kann. Das Problem der Konsistenz ist auch hier wieder entscheidend.

Selbst bei Fachleuten, die solche Annotationen durchführen, ergeben sich häufig Unterschiede. Ist „Garfield“ nun eine Katze, eine Figur oder ein Cartoon? Wenn sich die Aussagen auf den verschiedenen Webseiten nicht decken, kann das System nicht alle Antworten finden.

Mehr Konsistenz liesse sich erreichen, wenn man auch die Annotation von Webseiten und das Erstellen der Ontologie automatisieren würde. Man könnte meinen, dass im Web ja schon einiges an Hierarchien vorhanden ist. So bietet Yahoo als Suchhilfe sog. directories an. Das sind Stichworte die einander Untergeordnet sind. Doch leider sind diese directories für den Computer von keinem Nutzen. Das Problem liegt wiederum bei der Konsistenz. Die verschiedenen Begriffe stehen nämlich nicht in einer eindeutigen Relation zueinander. Man findet dort Hierarchien wie:

Gesundheit > Arbeitsplatz > Mobbing
Oder:
Umwelt und Natur > Energie > Veranstaltungen

Die Beziehungen dieser Begriffe zueinander machen höchstens für den Menschen einen Sinn, können aber von einer Maschine nicht interpretiert werden. So ist *Arbeitsplatz* wohl kaum eine Art von *Gesundheit* und *Veranstaltungen* sind keine Art von *Energie*.

Aber auch das Erstellen von Aussagen, also die Annotation von Webseiten, stellt sehr hohe Anforderungen an ein System, dass dies automatisch bewerkstelligen will. Selbst wenn es sich um einen eng umrissenen Bereich handelt, ist es schwierig alle Informationen aus einem Dokument zu extrahieren. Will man z.B. Webseiten annotieren, in denen ein Vortrag angekündigt wird, so möchte man die Informationen zu Ort, Datum, Zeit, Thema und Referenten wissen. Es ist aber für ein System selbst in so einem eng eingegrenzten Bereich schwierig, all diese Informationen automatisch zu extrahieren. Selbst das Auffinden des Datums ist nicht immer einfach. So gibt es keine einheitliche Schreibweise für das Datum. Es existieren verschiedene Variationen wie: 09.08.2003 oder 9. Aug. 2003 usw. Oder es wird eine Zeitspanne angegeben wie 9. – 12. August 2003. In den USA werden Monat, Tag und Jahr in umgekehrter Reihenfolge geschrieben, also 2003.08.09.
Wenn man andererseits einmal ein Datum gefunden hat, stellt sich auch noch die Frage, ob dieses Datum auch dasjenige des Vortrags ist und nicht etwa das Geburtsdatum des

Referenten. Es genügt also nicht nur ein Datum zu finden, man muss auch noch den Kontext erkennen, in dem das Datum genannt ist.
Natürlich sind solche Aufgaben durchaus Lösbar und auch wenn solche Systeme nicht immer alle Informationen finden, so kann man durchaus beachtliche Erfolge damit erzielen.

Ist der Bereich aber nicht so genau definiert, wird die Aufgabe noch viel komplexer. Gerade aber in diesem Bereich, wo es um natürlichsprachliche Texte geht, hat sich in der CL schon ein grosses Wissen angesammelt, das auch für die Entwicklung das SW nützlich sein könnte, weil im WWW der grösste Teil der Informationen in natürlicher Sprache vorhanden ist.

6. Fazit

Ich habe in dieser Arbeit zuerst die Motivation für ein SW erläutert. Der Hauptgrund liegt in der ungenügenden Strukturiertheit des WWW, das eine effiziente Suche nach seinen Inhalten erschwert. Das Problem liegt hauptsächlich in der html Syntax, die nur geringe Möglichkeiten bietet, die Inhalte von Webseiten mit Metadaten anzureichern.
Mit der RDF Syntax hat das W3C ein Werkzeug entwickelt, das es erlaubt, Aussagen über den Inhalt einer Webseite zu machen und gleichzeitig diese Metadaten in einer Ontologie zu strukturieren. Indem ich auf die Syntax eingegangen bin und den XML namespace vorgestellt habe, hat sich auch das Bauprinzip des SW herauskristallisiert. Zentral ist die Idee von vielen, kleinen, domänenspezifischen Ontologien, die jeweils in einem namespace definiert werden und so für alle zugänglich sind.
Natürlich hat ein so grosses Projekt auch seine Mängel. Einige Probleme sind bereits erkannt worden und es existieren auch Lösungsansätze dazu. Man muss auch bedenken, dass das SW noch in seinen Anfängen steckt, ist es doch erst einige Jahre alt. Andererseits beschränkt sich der Problemkreis nicht nur auf das WWW, sondern auf alle grösseren Datenbanken, die hauptsächlich aus narürlichsprachlichen Daten bestehen. Grade hier hat sich in der CL bereits ein beträchtliches Wissen angesammelt und es wäre wünschenswert, wenn davon auch das SW profitieren könnte.

7. Abbildungsverzeichnis

8. Literatur

[Berners-Lee 2001] Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web*, Scientific American, 2001
http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2

[Lassila 1999] Ora Lassila, Ralph R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*,1999
http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[Oberle 2001] Daniel Oberle, Raphael Volz, *Implementierung eines Sichtenmechanismus für ontologiebasierte Metadaten*, 2001
http://www.aifb.uni-karlsruhe.de/WBS/dob/pubs/KAON-Views.pdf

[Brikley 2000] Dan Brikley, *Resource Description Framework (RDF) Schema Specification 1.0*, 2000
http://www.w3.org/TR/2000/Cr-rdf-schema-20000327/

[Berners-Lee 1998] Tim Berners-Lee, *Semantic Web Road map*, 1998
http://www.w3.org/DesignIssues/Semantic.html

[Berners-Lee 2002] Tim Berners-Lee, *The Semantic Web*, 2002
http://www.w3.org/2002/Talks/04-sweb

[Hayes 2002] Patrik Hayes, *RDF Model Theory*, 2002
http://www.w3.org/TR/rdf-mt/

[Brickley 2002] Dan Brickley, *RDF Vocabulary Description Language 1.0: RDF Schema*, 2002
http://www.w3.org/TR/rdf-schema/

[Palmer 2001] Sean B.Palmer, *The Semantic Web: An Introduction*, 2001
http://www.infomesh.net/2001/swintro/