Alexander Maedche, Boris Motik

Repräsentations- und Anfragesprachen für Ontologien – eine Übersicht

Semantik spielt in vielen Anwendungsgebieten eine wichtige Rolle. Eines der Anwendungsgebiete von semantischen Modellen ist die Informationsintegration, in welcher heterogene Informationsquellen zusammengeführt werden. Dieser Beitrag gibt eine Einführung und Übersicht über den aktuellen Stand im Bereich der Repräsentations- und Anfragesprachen für semantische Modelle und stellt den Karlsruher Ansatz KAON zur Ontologierepräsentation und -anfrage vor.

1 Einführung

Die Unterstützung expliziter Modellierung von Information mittels semantischer Modelle ist ein Thema, das derzeitig unter unterschiedlichen Gesichtspunkten in verschiedenen Bereichen der Informatik aufgegriffen wird. Im Bereich der künstlichen Intelligenz und im Speziellen in der »Wissensakquisition und -repräsentation« verfolgt man die Vision eines Semantic Web [Berners-Lee et al. 2001], welches die im WWW verfügbare Information nicht nur für Menschen, sondern auch für Maschinen verständlich machen soll. Im Bereich der Datenbank- und Informationssysteme tritt verstärkt das Thema der Informationsintegration, in welchem heterogene Informationsquellen zusammengeführt werden, in den Vordergrund [Ludäscher et al. 2001]. Semantische Modelle können dabei sowohl zur Quellenbeschreibung als auch als integriertes und übergreifendes Modell dienen, um Quellenheterogenität zu überbrücken.

Obwohl der Nutzen semantischer Modelle allgemein anerkannt wird, mangelt es an einer akzeptierten und standardisierten Repräsentations- und Anfragesprache für semantische Modelle. Im Gegenteil, wer Techniken und Werkzeuge zum Aufbau, zur Repräsentation und zur Anfrage semantischer Modelle nutzen will, wird konfrontiert mit einer Vielfalt von Varianten von Repräsentations- und Anfragesprachen.

Dieser Beitrag gibt eine Einführung und Übersicht über den aktuellen Stand im Bereich der Repräsentations- und Anfragesprachen für semantische Modelle und stellt den Karlsruher Ansatz KAON zur hybriden Ontologierepräsentation und -anfrage vor. Neben dieser Einführung und Übersicht wird kurz die Informationsintegration als ein klassisches Anwendungsgebiet von semantischen Modellen diskutiert.

2 Was ist Semantik?

Dieser Abschnitt beschreibt die wünschenswerten Eigenschaften eines Modells, um die Bezeichnung »semantisches Modell« tragen zu können. Die zugrunde liegende Idee aller semantischen Modelle ist die Fähigkeit, eine eindeutige Bedeutung jedem Informationselement explizit zuordnen zu können. Um dieses Ziel zu erreichen, sollte ein semantisches Modell den folgenden drei Anforderungen gerecht werden:

- **Gemeinsame Begrifflichkeit.** Die Bedeutung von Information kann nur eindeutig unterschieden werden, falls alle Akteure eine gemeinsame Sprache sprechen. Das realisiert man typischerweise, indem zu jedem Ding eines bestimmten Anwendungsbereiches ein eindeutiges Symbol zugewiesen wird. Diese eindeutige Abbildung schafft dann die Vorraussetzung, durch symbolische Manipulation »Wissen« zu manipulieren. Dieses Vorgehen ist jedoch nur erfolgreich, wenn der Anwendungsbereich sehr klar abgegrenzt ist. Andernfalls wird es schwierig, die real existierenden Dinge jeweils eindeutig zu identifizieren. Diese Problematik kann man auf unterschiedliche Weise lösen. Typischerweise entwickelt man für jeden Anwendungsbereich eine spezielle Begrifflichkeit, unabhängig von allen anderen Anwendungsbereichen. Um Anwendungsbereiche zu vereinigen, muss man Übereinstimmungen bzw. Überlappungen zwischen den jeweiligen Dingen aus unterschiedlichen Anwendungsbereichen explizit repräsentieren [Maedche et al. 2002].
- **Beziehungen zwischen Dingen.** Im Allgemeinen ist es nicht ausreichend, nur jedem Ding ein Symbol zuzuweisen, da eine explizite Semantik erst durch das Verstehen der Beziehungen zwischen Dingen entsteht. Aus diesem Grund bieten alle semantischen Modelle Modellierungsprimitive zur Beschreibung der jeweiligen Rolle eines Symbols. Diese Modellierungsprimitive folgen in der Regel dem objektorientierten Paradigma.
- **Mathematisch fundierte Semantik.** Wesentlicher Vorteil ausdrucksstarker semantischer Modelle ist, dass ein freies Kombinieren der zur Verfügung gestellten Modellierungsprimitiven möglich ist. Es ist dabei jedoch sehr wichtig, die Semantik dieser Primitive eindeutig und mathematisch korrekt zu definieren, da nur dadurch eine eindeutige Interpretation eines jedes einzelnen Konstrukts ermöglicht wird. Der Mechanismus zur Definition einer mathematisch fundierten Semantik wird auch als Modelltheorie [Fitting 1996] bezeichnet.

Wir bezeichnen Modelle, die alle drei Eigenschaften besitzen, als Ontologien. Viele Modellierungsansätze weisen die ersten beiden Eigenschaften auf, während die dritte Eigenschaft, nämlich die auf einer Modelltheorie basierende Semantik, typischerweise nur in semantischen Modellen zu finden ist. Im Folgenden erklären wir daher dem Leser etwas detaillierter, was mit dieser sehr wichtigen Eigenschaft gemeint ist. Semantische Modelle werden normalerweise in einer symbolischen Sprache dargestellt. Diese Sprache definiert die Syntax des Modells. Auf der linken Seite von Abbildung 1 wird eine einfache Ontologie mittels einer grafischen Syntax dargestellt. Die Beispielontologie *Ontologie* enthält den Begriff *Person*, der als eine Menge aller Personen zu verstehen ist. Für jede Person kann das Attribut *Alter* spezifiziert werden, mit dem man das Alter der Person definieren kann. Der Begriff *Autor* ist ein Unterbegriff des Begriffs *Person*, der nur die Personen, die ein Dokument geschrieben haben, enthält. Zu jedem Autor ist es möglich, die Dokumente, die er geschrieben

hat, mittels der *hat-geschrieben*-Beziehung zuzuordnen. Die inverse Beziehung, zwischen Dokumenten und Autoren, ist durch die *hat-Autor*-Beziehung dargestellt. *Peter* ist eine Instanz vom Begriff *Person*, und *Alex* ist eine Instanz vom Begriff *Autor*. Da *Alex* ein *Autor* ist, kann man explizit sagen, dass Alex *Artikel1* geschrieben hat. Analog dazu kann man das Alter beider Personen repräsentieren.

Die Beispielontologie modelliert einen Ausschnitt der realen Welt, ähnlich wie viele objektorientierte Modelle. Da die Ontologie ein semantisches Modell darstellt, bieten sich zudem Möglichkeiten an, nicht explizit dargestellte Informationen automatisch ableiten zu können. Beispielsweise enthält die Ontologie die explizite Information, dass *Alex Artikel1* geschrieben hat und dass *hat-geschrieben* und *hat-Autor* inverse Beziehungen sind. Aus dieser Information lässt sich ableiten, dass eine *hat-Autor*-Beziehung zwischen *Artikel1* und *Alex* gilt.

Um solche Ableitungen automatisch durchführen zu können, muss die Bedeutung der Symbole mathematisch präzise formuliert sein. Dies erfolgt mittels der Verwendung einer so genannten Interpretationsfunktion, die jedem Symbol ein Ding einer abstrakten Interpretationsdomäne zuweist. Dabei werden Symbole abhängig von ihrer jeweiligen Rolle unterschiedlich interpretiert. Beispielsweise werden Instanzen als individuelle Dinge der Interpretationsdomäne, Begriffe als Mengen von Dingen der Interpretationsdomäne und Beziehungen als Paare von Dingen interpretiert.

Dieses Vorgehen ist exemplarisch für eine einfache Beispielontologie in Abbildung 1 dargestellt. Die Interpretationsdomäne enthält Dinge der realen Welt, wie z.B. eine konkrete Person namens Alex, einen konkreten Artikel sowie das Faktum, dass Alex Artikel1 geschrieben hat. Die Zuordnung von Dingen zu Symbolen folgt auf eine recht intuitive Weise, z.B. wird dem Symbol *Alex* die eigentliche Person Alex zugeordnet. Die Beziehungen zwischen unterschiedlichen Symbolen stellen die Einschränkungen an die Interpretationsfunktion fest, die von jeder Interpretation eingehalten werden müssen. So muss die Interpretation eines Unterbegriffs immer eine Teilmenge der Interpretation des Oberbegriffs sein. In unserem Beispiel bedeutet das konkret, dass die Interpretation von *Person* die Interpretation von *Autor* beinhalten muss und somit alle Autoren auch Personen sind. Ein weiteres Beispiel sind die inversen Beziehungen, die als inverse Relationen interpretiert werden müssen. Unsere Beispielontologie enthält das Faktum, dass *Alex Artikel1 geschrieben-hat* und dass *hat-geschrieben* und *hat-Autor* inverse Beziehungen sind. Das bedeutet, dass die Interpretation von *hat-geschrieben* und *hat-Autor* invers zueinander sind, und daraus lässt sich schließen, dass die Beziehung *hat-Autor* zwischen *Artikel1* und *Alex* gilt.

Es muss dabei betont werden, dass man sich für eine Ontologie beliebig viele Interpretationen vorstellen kann. Man sollte ebenfalls auch nicht den Fehler machen anzunehmen, dass der Name des Symbols die Auswahl des Elementes der Interpretationsdomäne definiert. So zum Beispiel wäre es auch absolut korrekt, das Symbol *Alex* mit der Nummer 1 zu interpretieren. Für eine Ontologie muss man immer alle möglichen Interpretationen berücksichtigen. Wenn etwas in allen möglichen Interpretationen der Ontologie gilt, dann kann dies als allgemein gültig betrachtet werden. Dabei enthält die Ontologie die Axiome, die die Struktur der möglichen Welten einschränken. Diese Axiome sollten dabei so definiert werden, dass sie die beabsichtigte Welt möglichst korrekt beschreiben. In der Realität ist es jedoch unmöglich, eine Welt vollkommen genau zu beschreiben. In diesem Sinne sind Ontologien immer nur Modelle der realen Welt und deswegen modelliert eine Ontologie immer mehrere Welten parallel. Schließlich sind die Fakten, die aus einer Ontologie folgen, genau diese, die in allen möglichen Welten gelten.

Genau in diesem Punkt unterscheidet sich die formale Semantik von einfachem Kodieren von Information. Praktisch betrachtet, könnte auch jedes Ding mit einem Symbol repräsentiert werden und zu jedem Symbol eine flache Liste von bestimmten Eigenschaften zugeordnet werden. So könnte man das Faktum, dass *Alex* ein *Autor* ist, mit der *hat-Typ=Autor*-Eigenschaft kodieren. In einem solchen Modell ist die Information über den Typ des Dinges zwar vorhanden, ist jedoch nur enkodiert und nicht semantisch dargestellt. Um ableiten zu können, dass für *Alex* auch die *hat-Typ=Person*-Eigenschaft gilt, muss man die *hat-Typ*-Eigenschaft speziell behandeln (auf jeden Fall unterschiedlich von z.B. der *hat-Alter*-Eigenschaft). Wie bereits dieses sehr einfache Beispiel zeigt, führt das Fehlen eines klaren mathematischen Modells dazu, dass korrektes Schlussfolgern schwierig oder gar unmöglich wird.

Dass diese gewünschten mathematischen Eigenschaften in praktischen Fällen wichtig sind, kann man auch anhand des »Barbier-Paradoxon« von *Bertrand Russell* sehen, welches wir im Folgenden kurz betrachten wollen. Nehmen wir an, es gibt in einer Stadt einen Barbier, der einer bestimmten Regel folgt: »Er schneidet die Haare von allen Personen, die sich selbst nicht die Haare schneiden.« Eine entsprechende Beispielontologie ist in Abbildung 2 dargestellt. Alex schneidet sich selbst die Haare und Peter nicht – deswegen schneidet gemäß der Regel der Barbier die Haare von Peter. Die dabei entstehende Frage ist jedoch, wer schneidet die Haare des Barbiers? Es stellt sich heraus, dass auf diese Frage keine Antwort gegeben werden kann. Wenn wir die Annahme machen, dass er sich selbst die

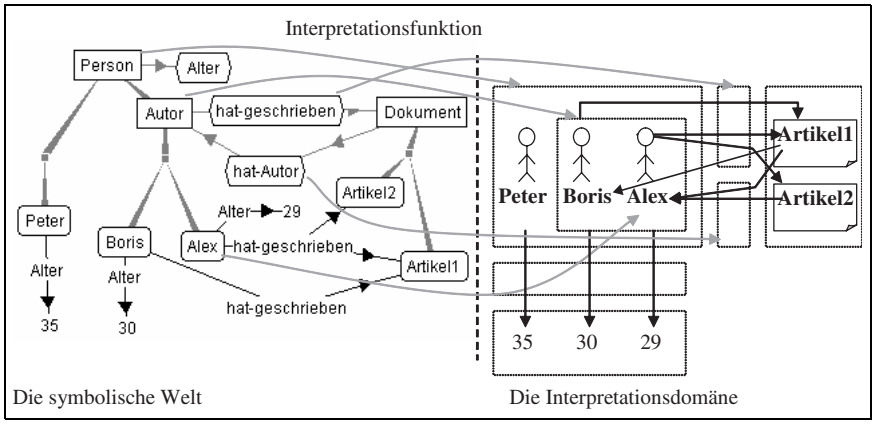

Abb. 1: Beispiel einer Ontologie und einer möglichen Interpretation

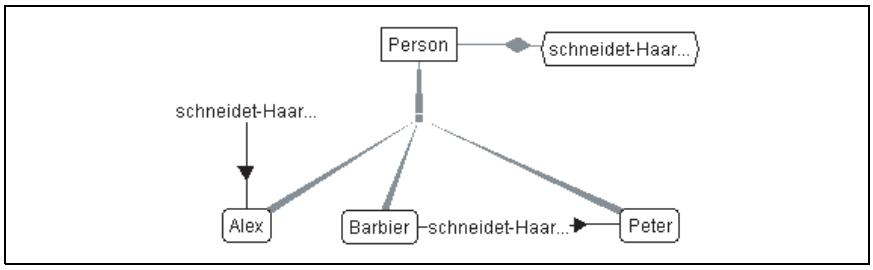

Abb. 2: Das »Barbier-Paradoxon«

Haare schneidet, dann sollte er es laut seiner Regel nicht tun, da er nur die Haare der Leute schneidet, die sich selbst die Haare nicht schneiden. Analog dazu, wenn wir die Annahme machen, dass er sich selbst die Haare nicht schneidet, dann sollte er es gemäß der Regel eigentlich tun. Auf Basis der gegebenen Information ist es also nicht möglich, eine eindeutige Antwort auf obige Frage zu geben.

Es ist nicht trivial, ohne Unterstützung eines wohlfundierten mathematischen Modells ähnliche komplexe Situationen zu beschreiben. Die Verwendung einer Modelltheorie hilft uns hier: Gemäß unserer Regel soll die Interpretationsfunktion für die *schneidet-Haare-von*-Beziehung eine Menge S sein, die unter anderem die folgende Eigenschaft aufweisen soll:

$$\forall x\,(x,x) \notin S \Rightarrow (Barbier,x) \in S$$

Auf dieser Basis lässt sich das Paradoxon leicht aufspüren, da es aus der mathematischen Mengentheorie bekannt ist, dass eine solche Menge nicht existiert. Aus den mathematischen Eigenschaften der Modelltheorie ist es auch möglich, die Bedingungen, unter welchen ähnliche Situationen nicht vorkommen, abzuleiten. Unser Problem taucht deswegen auf, weil die Definition der *schneidet-Haare-von*-Beziehung auf sich selbst durch Negation verweist. Falls derartige Definitionen in der Ontologie nicht existieren, kann bewiesen werden [Abiteboul et al. 1995], dass das Modell immer korrekt ist.

3 Die Rolle von Ontologien in der Informationsintegration

Dieser Abschnitt motiviert kurz, wie Ontologien als semantische Modelle im Gebiet der Informationsintegration verwendet werden können. Die Notwendigkeit der Informationsintegration entsteht dadurch, dass typischerweise unterschiedliche Informationssysteme das gleiche oder ähnliche Anwendungsgebiete aus unterschiedlichen Sichten betrachten. Dies führt dazu, dass sich die Datenmodelle dieser Systeme teilweise überlappen. Trotz dieser Überlappungen können die gemeinsamen Teile der Modelle meistens nicht eins-zu-eins abgebildet werden, da die Art, wie die Information im jeweiligen Modell strukturiert und repräsentiert ist, durch die Anforderungen an die jeweiligen Systeme stark beeinflusst wurde. Um diese Heterogenität zu überbrücken, ist es sehr wichtig, die semantischen Beziehungen zwischen Entitäten innerhalb und zwischen den Modellen möglichst exakt zu repräsentieren. Da Ontologien diese Beziehungen explizit modellieren können, sind sie besonders geeignet für diese Aufgabe. Wir beziehen uns im Folgenden auf einen typischen Informationsintegrationsansatz, der in [Lud äscher et al. 2001] ausführlich beschrieben wurde.

In einem ersten Schritt verwendet man in diesem Ansatz Ontologien, um das Datenmodell jedes zur Integration vorgegebenen Systems zu beschreiben. In einem zweiten Schritt wird dann eine Ontologie als die integrierte Sicht auf das Integrationssystem erzeugt. Beide Schritte stellen aufwendige Modellierungsprozesse dar. Beim ersten Schritt, der Remodellierung der existierenden Systeme mittels einer Ontologie, lassen sich folgende wichtige Aspekte von semantischer Modellierung erkennen:

- Auf Basis eines formalen Modells können die erzeugten Ontologien automatisch auf Konsistenz geprüft und potenzielle Fehler entdeckt werden. Beispielsweise lässt sich herausfinden, dass die Definition zweier Begriffe widersprüchlich ist.
- Zusätzlich ist es möglich, mittels Ontologien allgemeines Domänenwissen zu repräsentieren (z.B. die Transitivität einer Beziehung). Dieses explizit repräsentierte Wissen kann für die Anfrageverarbeitung verwendet werden.

Nachdem die Datenmodelle der existierenden Systeme als Ontologien modelliert wurden und die integrierte Ontologie erzeugt wurde, müssen die semantischen Korrespondenzen zwischen Symbolen der integrierten Ontologie und der lokalen Ontologien erzeugt werden. Dies ist typischerweise die schwierigste Aufgabe, da die Begriffe in einzelnen Modellen nicht eins-zu-eins mit der integrierten Ontologie abgebildet werden können. Um diese Aufgabe durchzuführen, ist es entscheidend, die intendierte Bedeutung jedes Symbols der jeweiligen Ontologie exakt zu repräsentieren. Hier ist wieder die Darstellung des Modells in Form einer Ontologie sehr hilfreich: Die Beziehungen im Modell sind explizit dargestellt und die Semantik der Symbole eindeutig spezifiziert. Eine weitere Frage stellt sich, wie die Korrespondenzen zwischen den Modellen konkret repräsentiert werden. Typischerweise unterscheidet man die beiden folgenden Ansätze: Entweder werden die Elemente des integrierten Modells als Anfragen über Elemente des lokalen Modells spezifiziert (sog. *global-as-view*-Ansatz) oder umgekehrt (sog. *local-as-view*-Ansatz). Falls die jeweiligen Datenmodelle als semantische Modelle in Form von Ontologien explizit ausgedrückt wurden, wird dies zusätzlich vereinfacht, da die Korrespondenzen als ganz normale Beziehungen zwischen Begriffen der Ontologien repräsentiert werden können. Nehmen wir beispielsweise an, dass die integrierte Ontologie den Begriff *Unterkunft*, eine lokale Ontologie den Begriff *Hotel* und eine weitere lokale Ontologie den Begriff *Apartment* enthält. Die semantischen Korrespondenzen zwischen diesen beiden Begriffen werden spezifiziert, indem man die Begriffe *Hotel* und *Apartment* als Unterbegriffe vom Begriff *Unterkunft* definiert. Analog lässt sich mittels dem *global-as-view*-Ansatz definieren, dass alle Instanzen des Begriffs *Unterkunft* als eine Vereinigung von Instanzen von *Hotel* und *Apartment* berechnet werden.

4 Repräsentationssprachen

In diesem Abschnitt beschreiben wir existierende Möglichkeiten und Ansätze zur Repräsentation von (semantischen) Modellen.

4.1 Das relationale und das Entity-Relationship-Modell

Das am meisten verwendete Modell für die Verwaltung von großen Datenmengen ist das relationale Modell [Abiteboul et al. 1995]. Dieses Modell wurde in vielen kommerziellen Produkten erfolgreich umgesetzt und hat allgemeine Verbreitung gefunden. Es ist jedoch bekannt, dass das relationale Modell in Bezug auf semantische Modellierung einige Schwächen aufweist. Hauptproblem dabei ist, dass die unterschiedlichen Konstrukte des Modells nur indirekt die realen Dinge des jeweiligen Anwendungsbereiches darstellen. Das relationale Modell fokussiert im Wesentlichen auf Konstrukte, die einen effizienten Zugriff auf die Daten ermöglichen, aber mittels derer die Dinge des jeweiligen Anwendungsbereiches erst durch eine zusätzliche Abstraktionsebene dargestellt werden können. So bietet das relationale Modell keine Konstrukte zur direkten Modellierung von Beziehungen zwischen Dingen eines Anwendungsbereiches an. Sicherlich kann man Joins als Implementierungsmechanismus verwenden, dies löst jedoch die eigentliche Modellierungsaufgabe nur indirekt. Außerdem unterstützt das relationale Modell in seiner ursprünglichen Form keine Konstrukte zur Modellierung von Objekthierarchien. Stattdessen sind die Benutzer des Modells gezwungen, eine Objekthierarchie mit existierenden Konstrukten zu simulieren. Dieses Problem wird durch den Mangel von deduktiven Inferenzen, wie z.B. das Berechnen der transitiven Hülle der Objekthierarchie, weiter verstärkt[1]. Der Mangel an deduktiven Inferenzen verhindert auch die explizite Darstellung der Semantik von Objektbeziehungen. In der realen Anwendung bedeutet das normalerweise, dass die Semantik des relationalen Modells meistens nicht im Modell selbst enthalten, sondern oberhalb des Modells in Form der Anwendungslogik realisiert ist.

Um die Schwächen des relationalen Modells zu kompensieren, wurde das Entity-Relationship-Modell entwickelt. Es ist das am meisten verwendet semantische Modell. Das ER-Modell bietet spezielle Konstrukte, die für die Modellierung eines Anwendungsbereiches verwendet werden können, wie z.B. »Entitäten«, die Elemente der realen Welt darstellen, und »Beziehungen«, die die Beziehungen zwischen den Entitäten modellieren. Oberhalb dieses Basismodells wurden zahlreiche Erweiterungen in der Literatur diskutiert. Zum Beispiel wurden in [Gogolla & Hohenstein 1991] eine formale Definition der Semantik des Modells sowie Erweiterungen für das explizite Modellieren von Entitätshierarchien vorgestellt.

Das größte Problem des ER-Modells liegt jedoch darin, dass zur Ausführung des Modells eine Transformation in ein ausführbares logisches Modell (fast ausschließlich ins relationale Modell) notwendig ist. Nach dieser Transformation kann dann das semantische Modell nicht mehr direkt manipuliert werden. Das hat mehrere Nachteile, wobei der schwerwiegendste die Divergenz zwischen konzeptuellem und logischem Modell ist. Des Weiteren bietet das ER-Modell keine deduktiven Primitive an und teilt die gleichen Probleme mit dem relationalen Modell bezüglich der Möglichkeiten zur expliziten Darstellung von Semantik.

4.2 Das objektorientierte Modell

Die Entwicklung des objektorientierten Modells wurde motiviert von der Idee, ein allgemeines Modellierungsparadigma anzubieten, welches gleichzeitig für das konzeptuelle und logische Modellieren verwendet werden kann. Die Ziele des OO-Modells gehen über das reine Modellieren hinaus, so kann beispielsweise die Anwendungslogik mittels Methoden repräsentiert und damit eng mit dem Datenmodell integriert werden. Mehrere unterschiedliche OO-Modelle sind in den letzten Jahren vorgeschlagen worden, wobei sich das von der Object Data Management Group (ODMG) [Cattell et al. 2000] entwickelte Modell als Referenz durchgesetzt hat.

Obwohl sich das OO-Modell prinzipiell für das Modellieren von Ontologien verwenden lässt, sind die Ziele des OO-Modells und der existierenden Ontologiesprachen unterschiedlich. Dies lässt sich an einem einfachen Beispiel erklären: Beide Modelle besitzen das Konzept der Klasse. In OO-Modellen dient eine Klasse zur Typisierung des Objekts, wobei der Typ eines Objekts beim Erzeugen festgestellt wird und sich nicht nachträglich ändern lässt. Hätte man die Beispielontologien aus Abbildung 1 mittels eines OO-Modells realisiert, wäre es nicht möglich gewesen, Peter, nachdem er einen Artikel geschrieben hat, als Autor neu zu »klassifizieren«. Dies macht den Unterschied klar. In Ontologien werden Klassen zur Klassifikation verwendet, wobei die Zugehörigkeit eines Objekts zu einer Klasse bedeutet, dass das Objekt bestimmte Eigenschaften dieser Klasse aufweist. Generell kann die Klasse des Objekts geändert werden, falls sich die Eigenschaften des Objekts ändern.

Analog zum relationalen und zum ER-Modell bietet das OO-Modell keine deduktiven Primitive. Daher müssen die einfachen semantischen Eigenschaften, wie Transitivität einer Eigenschaft, manuell implementiert werden.

4.3 RDF(S)

In [Berners-Lee et al. 2001] wurde die Vision eines Semantic Web beschrieben, in der online verfügbare Information nicht nur für Menschen, sondern auch für Maschinen verständlich gemacht werden soll. Auf dieser Basis könnten Maschinen die großen Mengen vorhandener Daten mit einem höheren Automatisierungsgrad bearbeiten und damit dem Menschen bei der Bewältigung der Informationsflut helfen. Eine zentrale Voraussetzung für die Umsetzung dieser Vision ist die Beschreibung von Web-Ressourcen mit formaler Semantik. Als erster Schritt in diese Richtung wurden vom World Wide Web Consortium (W3C) zwei Sprachen zur Metadatenrepräsentation entwickelt und als Standards propagiert. Das Resource Description Framework (RDF) [Lassila & Swick] bildet dabei das Fundament und stellt die Syntax und die Semantik für die Definition einfacher Metadatenmodelle. Das Vokabular von RDF besteht im Wesentlichen aus drei zentralen Elementen:

1. In der SQL:1999-Version bietet das relationale Modell eine begrenzte Möglichkeit für rekursive Anfragen und Sichten, die bestimmte deduktive Inferenzen erleichtern. Diese Möglichkeit ist aber nur auf sog. lineare Rekursion begrenzt und reicht im allgemeinen Fall nicht aus.

- »Ressourcen« sind Objekte, über die etwas gesagt werden kann. Alle Ressourcen können durch einen eindeutigen Bezeichner (Uniform Resource Identifier – URI) identifiziert werden.
- »Literale« sind Werte (wie z.B. Zeichenketten) zur Beschreibung der unterschiedlichen Eigenschaften von Ressourcen.
- »Eigenschaften« sind benannte Kanten zwischen Ressourcen und Literalen, die die Eigenschaften von Ressourcen spezifizieren. Eine Eigenschaft kann auf ein Literal oder auf eine Ressource verweisen.

Das grundlegende RDF-Modell ist damit ein graphbasiertes Datenmodell, das seinen Ursprung im Object Exchange Model (OEM) [Papakonstantinou et al. 1995] – einem Modell für semistrukturierte Daten – findet. Ein sehr einfaches RDF-Modell ist in der vom W3C definierten grafischen Syntax in Abbildung 3 dargestellt. Im Beispiel ist die Ressource #Alex mittels der Eigenschaft #Name mit dem Literal »Alex« verknüpft (was die Tatsache darstellt, dass der Name der Person, die durch die Ressource #Alex modelliert wird, gleich Alex ist) und mittels der Eigenschaft #hat-geschrieben mit der Ressource #Artikel1 (was die Tatsache darstellt, dass #Alex den #Artikel1 geschrieben hat).

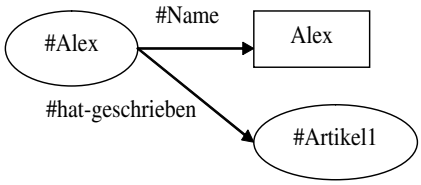

Abb. 3: Einfaches Beispiel eines RDF-Modells

RDF ist wie bereits erwähnt ein einfaches graphbasiertes Modell und bietet keine Möglichkeiten zur weiteren Strukturierung der Information an. Aus diesem Grunde wurde oberhalb von RDF die Sprache RDF Schema (RDFS) [Brickley & Guha] definiert. RDFS bietet die Möglichkeit, Klassen zu deklarieren, unterschiedliche Instanzen den Klassen zuzuordnen und mögliche Beziehungen zwischen Klassen zu definieren. RDFS ist innerhalb RDF realisiert, indem die Modellierungsprimitive als spezielle Beziehungen mit ganz bestimmten Eigenschaften definiert wurden. Um beispielsweise auszudrücken, dass die Ressource #Person eine Klasse aller Personen darstellen soll, definiert man die Beziehung rdf:type zwischen #Person und rdfs:Class (rdf:type ist eine Beziehung, die eine bestimmte Semantik hat; die Semantik von rdf:type wird außerhalb des RDF-Modells spezifiziert).

RDFS ist eine sehr einfache Ontologierepräsentationssprache und bietet keine Möglichkeiten zur komplexeren Modellierung. In diesem Sinne ist es in RDFS auch nicht möglich, das »Barbier-Paradoxon« korrekt zu modellieren. Des Weiteren ist zu betonen, dass von ihrer Historie her sowohl für RDF als auch RDFS zu Beginn keine saubere Semantik definiert war. Dieses Problem wurde mittlerweile korrigiert (die offizielle Semantik heißt RDF-MT [Hayes]), jedoch bleiben trotz der formalen Spezifikation der Semantik der Sprachen einige Probleme bestehen. Zentrales Hauptproblem der Sprachen ist, dass die jeweiligen Primitive sowohl auf der Ontologiemodellierungs- als auch auf der Instanzebene verwendet werden können. Um diese Situation zu bereinigen, wurde die RDFS Finite Model Architecture (RDFS-FA) [Pan & Horrocks 2001] vorgeschlagen. Diese Variante von RDFS teilt die unterschiedlichen Beziehungen in separate Ebenen. So werden die Beziehungen zwischen Instanzen zur Instanzebene, die Beziehungen zwischen Klassen zur Ontologieebene, die Beziehungen zwischen Konstrukten der RDFS-Sprache zur Sprachebene und die Beziehungen für die Definition der Sprache zur Metaebene zugeordnet.

4.4 Topic Maps

Topic Maps [Pepper & Moore] sind ein Standard, der von TopicMaps.Org Authoring Group (AG) mit dem Zweck der Annotierung von Internetressourcen entwickelt wurde. Das zentrale Objekt von Topic Maps ist das »Topic«, das als Proxy für ein Subjekt (ein Objekt der realen Welt) betrachtet werden kann. Topics kann man verschiedenste Informationen zuordnen, wie beispielsweise den Namen oder Assoziationen mit anderen Topics. Der Standard enthält eine Menge bekannter Assoziationen, die die Einordnung von Topics in Klassenhierarchien ermöglichen. Der Topic-Maps-Standard ähnelt stark dem RDF(S)-Vorschlag vom W3C. Leider ist die Menge der zur Verfügung stehenden Primitive begrenzt – einfache semantische Konzepte wie symmetrische oder transitive Assoziationen lassen sich nicht ausdrücken. Wesentliches Problem der Topic Maps ist, dass das Datenmodell keine formale modelltheoretische Semantik bietet und deswegen nicht für deduktive Inferenzen geeignet ist.

4.5 F-Logik

F-Logik ist eine Ontologiemodellierungssprache, die ihren Ursprung in objektorientierten deduktiven Datenbanken hat [Kifer et al. 1995]. Die Sprache bietet die traditionellen Konstrukte der objektorientierten Modelle zur Datenstrukturierung, wie Klassen, Instanzen und Beziehungen, für die eine mathematisch saubere Modelltheorie definiert ist. Es ist möglich, die Klassen als Instanzen von anderen Klassen zu betrachten. Dies erfolgt, indem alle Klassen und Instanzen in eine partielle Ordnung gebracht werden. Damit hat man in F-Logik die Möglichkeit, die Klassennamen als normale Objekte zu betrachten (z.B. kann man über Klassennamen quantifizieren). In der Regel kann man diese Eigenschaft auch folgendermaßen bezeichnen: F-Logik hat die Syntax der Logik der zweiten Ordnung, aber die Semantik der Logik der ersten Ordnung. In F-Logik ist es des Weiteren auch möglich, Methoden einer Klasse als logische Inferenzregeln zu spezifizieren. Damit kann die Ontologie nicht nur strukturelle Information enthalten, sondern auch die Logik zur Interaktion zwischen Objekten steuern. F-Logik bietet im Vergleich zu den vorangestellten Ansätzen eine größere Ausdrucksmächtigkeit. Im Folgenden geben wir die Beispielontologie aus Abbildung 1 in F-Logik-Syntax an. Die Ontologie besteht aus vier unterschiedlichen Teilen. Zuerst wird die Begriffshierarchie spezifiziert, wobei die Ober-/Unterbegriff-Beziehung mit dem Zeichen »::« spezifiziert wird. In einem zweiten Schritt folgt die Spezifikation der möglichen Beziehungen für jeden Begriff. Drittens werden die Instanzen und ihre Eigenschaften spezifiziert, wobei das Zeichen »:« den Instanznamen vom Begriffnamen trennt. In einem letzten Schritt werden die Eigenschaften der Begriffe mittels F-Logik-Regeln spezifiziert.

```
// Begriffshierarchie
Autor::Person.
Person::#DEFAULT_ROOT_CONCEPT.
Document::#DEFAULT_ROOT_CONCEPT.

// Beziehungen
Autor["hat-geschrieben"=>>Dokument].
Person[Alter=>>a#Literal].
Dokument["hat-Autor"=>>Autor].

// Instanzen
Artikel:Dokument.
Alex:Autor.
Alex[Alter->>29.0;
   "hat-geschrieben"->>Artikel1;
   "hat-geschrieben"->>Artikel2].
Boris:Autor
Boris[Alter->>30.0;
   "hat-geschrieben"->>Artikel1]
Peter:Person.
Peter[Alter->>35.0].

// Eigenschaften der Beziehungen
FORALL X,Y  X["hat-Autor"->>Y] <-> Y["hat-geschrieben"->>X].
```

4.6 Beschreibungslogiken

Beschreibungslogiken [Borgida 1992] bilden eine Sprachfamilie der Ontologiemodellierung, die ihre Wurzeln in der KL-ONE-Sprache [Brachmann & Schmolze 1985] haben. Vor kurzem wurde auch vom W3C die Web Ontology Language (OWL) [Patel-Schneider et al. 2002] – eine ausdrucksmächtige Beschreibungslogikvariante – als Standard für Web-Ontologien propagiert. Ontologien in Beschreibungslogiken bestehen aus Begriffen und Beziehungen, ähnlich wie bei allen bisher untersuchten Ansätzen. Die Beschreibungslogiken differenzieren sich jedoch dadurch, dass die Extension eines Begriffs nicht explizit in der Ontologie spezifiziert werden muss. Begriffe der Ontologie können die notwendigen und/oder hinreichenden Bedingungen für die Mitgliedschaft in der Extension vorschreiben. Begriffe werden durch sog. »Begriffausdrücke« spezifiziert, die beliebig kombiniert werden können, um komplexere Ausdrücke zu erzeugen. Die Anzahl und die Art der zur Verfügung stehenden Primitive hängen von der eigentlichen Sprache und ihrer Ausdrucksstärke ab. Da die Komplexität und die Entscheidbarkeit der Inferenzprozeduren von den verfügbaren Primitiven abhängen, stellt jede Beschreibungslogik einen Kompromiss zwischen der Ausdrucksmächtigkeit und der Komplexität dar.

ALC [Schmidt-Schauß & Smolka 1991] ist eine einfache Beschreibungslogik, die die Basisprimitive fast aller anderen Sprachen enthält. Die Primitive dieser Sprache und die umgangssprachliche Interpretation sind in Tabelle 1 dargestellt.

Eine einfache Beispielontologie in ALC könnte wie folgt aussehen:

- Alle Personen haben einen Namen:
  Person := ∃ *Name.Zeichenkette*
- Autoren sind Personen, die mindestens ein Dokument geschrieben haben:
  Autor := *Person* ∩ ∃ *hat-geschrieben.Dokument*
- Interessante Dokumente sind Dokumente, die ausschließlich von Professoren geschrieben wurden:
  InteressantesDokument := *Dokument* ∩ ∀ *hat-Autor.Professor*

Dabei ist es besonders wichtig, zu verstehen, dass diese Beschreibungen als Bedingungen gelten. Beispielsweise ist es nicht nötig, die Extension des Begriffs *InteressantesDokument* explizit zu spezifizieren: Jedes konkrete Dokument, das nur Professoren als Autoren hat, wird automatisch als eine Instanz dieses Begriffs klassifiziert. Analog dazu ist jede Person, die mindestens ein Dokument geschrieben hat, automatisch ein *Autor*. Dies war in allen bisher dargestellten Modellen nicht möglich, da man typischerweise alle Instanzen explizit zu den Begriffen zuordnen musste.

Eine weitere Eigenschaft der Beschreibungslogiken ist das automatische Erzeugen einer entsprechenden Begriffshierarchie. Zum Beispiel ist es offensichtlich, dass alle Instanzen des Begriffs *Autor* immer eine Instanz des Begriffs *Person* sein müssen. In anderen Worten, der Begriff *Person* subsumiert den Begriff *Autor*. In den meisten Beschreibungslogiken ist diese Prozedur entscheidbar – für jede zwei Begriffausdrücke kann es immer entschieden werden, ob ein Begriff den anderen subsumiert. Damit kann man immer automatisch alle Begriffe in eine partielle Ordnung stellen.

Die Beschreibungslogiken werden typischerweise in Fällen, in denen die Begriffsstruktur sehr komplex ist, angewendet. Dabei finden die besonderen Eigenschaften, wie z.B. das automatische Berechnen der Begriffshierarchie, ihren besonderen Nutzen. Im Gegensatz dazu ist die Navigation in beschreibungslogikbasierten Modellen sowie beliebiges Filtern von Werten schwierig beziehungsweise gar nicht möglich.

5 Anfragesprachen

Neben der formalen Ontologierepräsentationssprache stellt die Anfragesprache einen wichtigen Faktor für die erfolgreiche Anwendung dar. Generell sind Anfragesprachen mit der jeweiligen Repräsentationssprache eng verwandt, da sie ja auf dem entsprechenden Modell aufbauen.

Bevor wir Anfragesprachen für die jeweiligen semantischen Modelle aus Abschnitt 4 vorstellen, diskutieren wir kurz

C ∩ D	Die Menge aller Elemente aus C und D.
C ∪ D	Die Menge aller Elemente aus C oder D.
¬ C	Die Menge aller Elemente, die nicht in C sind.
∀ R.C	Die Menge aller Elemente, die mit Beziehung R nur mit Elementen aus C verknüpft sind.
∃ R.C	Die Menge aller Elemente, die mit Beziehung R mit mindestens einem Element aus C verknüpft sind.

Tab. 1: Primitive und die Semantik der ALC-Sprache

welche Eigenschaften eine Anfragesprache aufweisen sollte. Eine Anfrage kann als Funktion verstanden werden, welche auf ein Modell angewendet wird, um eine Menge von Ergebnissen zu berechnen. Der Wertebereich dieser Funktion ist das entsprechende semantische Modell. Der Bildbereich der Funktion kann, aber muss nicht dasselbe Modell sein. Wir behaupten, dass es im Allgemeinen und bei semantischen Modellen im Besonderen von großem Vorteil ist, wenn der Wertebereich und der Bildbereich der Anfragen gleich sind. In diesen Fällen kann man eine Anfrage als eine Transformation des Modells verstehen. Das Ergebnis der Anfrage ist wieder ein konsistentes semantisches Modell, auf das man dann weitere Anfragen anwenden kann. Damit können Anfragen beliebig verkettet werden.

5.2 Relationale Anfragesprachen

Für das relationale Modell sind mehrere Anfragesprachen entwickelt worden. In der Praxis hat sich Structured Query Language (SQL) [Abiteboul et al. 1995] als die erfolgreichste und wichtigste Anfragesprache etabliert. SQL basiert auf der relationalen Algebra. Sie besteht aus Operationen, die auf mehreren Relationen (sprich Datenbanktabellen) verwendet werden können und eine neue Relation erzeugen. Da das Ergebnis jeder Operation wieder eine Relation ist, bleibt das Ergebnis einer Anfrage immer innerhalb des relationalen Modells, und damit können einfache Anfragen beliebig für den Aufbau komplexerer Anfragen kombiniert werden.

Größter Nachteil der relationalen Anfragesprachen ist, dass das relationale Modell, wie in Abschnitt 4 schon diskutiert, keine explizite Semantik aufweist. Das betrifft auch die Ergebnisse der relationalen Anfragen – das Ergebnis jeder Anfrage ist eine einfache Tabelle von Werten. Die semantischen Beziehungen zwischen den Werten sind im Ergebnis nicht explizit dargestellt.

5.2 Anfragesprachen für objektorientierte Modelle

Objektorientierte Modelle verfügen typischerweise über eine mächtige Anfragesprache die das Manipulieren und Wiederabrufen der Objekte ermöglicht. Die Object Query Language (OQL) [Cattell et al. 2000] ist Teil des ODMG-Standards und die wohl am weitesten entwickelte Anfragesprache für objektorientierte Modelle. OQL hat ihre Wurzeln im typisierten Lambdakalkül. Die Sprache besteht aus verschiedenen Operatoren, die auf Objekte eines bestimmten Typs angewendet werden können und dabei Objekte eines anderen, neuen Typs erzeugen. Um die Mengen von Objekten bearbeiten zu können, verfügt das ODMG-Modell über eine umfassende Menge von Kollektionsdatentypen, wie z.B. Mengen, Multimengen oder Listen. Bestimmte Operatoren der OQL nehmen solche Kollektionen als Parameter und erzeugen neue Kollektionen als Ergebnis. Obwohl OQL eine ausdrucksmächtige Sprache ist, ermöglicht sie keine rekursiven Anfragen. Somit ist es auch nicht möglich, deduktive Inferenzen mittels OQL einfach zu realisieren.

5.3 Anfragesprachen für RDFS

Seitdem RDF und RDFS vom W3C zur Metadatenbeschreibung im WWW vorgeschlagen wurden, sind zahlreiche Anfragesprachen für diese Modelle entwickelt worden. Die RDF Query Language (RQL) [Karvounarakis et al. 2002] hat sich als eine wichtige Anfragesprache für RDF/RDFS etabliert. RQL ist eine funktionale Anfragesprache, die stark von objektorientierten Anfragesprachen, wie z.B. OQL [Cattell et al. 2000], beeinflusst wurde. Um RDF mit funktionalen Primitiven bearbeiten zu können, wird oberhalb vom grundlegenden RDF-Modell ein Typsystem definiert. Zu allen Objekten des RDF-Modells wird der entsprechende Typ explizit zugewiesen. Jedes RQL-Primitiv kann auf Objekte des entsprechenden Typs angewendet werden und erzeugt ein neues Objekt eines bestimmten Typs. Beispielsweise gibt das Anwenden des Primitivs zur Aufzählung von Instanzen eines Begriffs eine Sammlung der zu dem Begriff gehörenden Instanzen als Ergebnis zurück.

Um komplexere Anfragen stellen zu können, verfügt RQL über Tupel – ein bestimmter Datentyp, der die sequenzielle Reihenfolge von Werten darstellt. Viele Anfragen liefern als Ergebnis eine Menge Tupel, so sieht z.B. die Anfrage an die Beispielontologie aus Abbildung 1, die alle Autoren mit ihren Artikeln selektiert, in RQL folgendermaßen aus:

```
SELECT X, Y
FROM Autor{X}.hat-geschrieben{Y}
```

Das Ergebnis dieser Anfrage ist eine Menge Tupel von zwei Werten. Ein mögliches Ergebnis ist in Tabelle 2 dargestellt.

X	Y
Alex	Artikel1
Alex	Artikel2
Boris	Artikel1

Tab. 2: Das Ergebnis einer RQL-Anfrage

Dieses einfache Beispiel zeigt auch eine der größten Schwächen der RQL-Sprache: Im Ergebnis wurde die Semantik des ursprünglichen Modells zerstört. Die Autoren sind im Originalmodell explizit mit jeweiligen Artikeln verknüpft. Falls mehrere Autoren mehrere Artikel geschrieben haben, wird dies durch eine n:m-Beziehung zwischen Autoren und Artikeln dargestellt. Die Möglichkeit, die Elemente des Modells zu verknüpfen, liegt allen semantischen Modellen zugrunde. Die obige RQL-Anfrage hat diese Verknüpfung zerstört – das Ergebnis weist die Beziehung zwischen Autoren und Artikeln nicht auf. Im Gegenteil, das Ergebnis enthält eine Menge Tupel, die diese Beziehung simulieren. Das Faktum, dass Alex zwei Artikel geschrieben hat, ist nicht durch eine 1:2-Beziehung dargestellt. Um auf diese Information zugreifen zu können, muss man zuerst die Ergebnisse nach X gruppieren und dann die Anzahl von wiederholten Y-Werten berechnen.

5.4 Anfragen in F-Logik

Anfragen in F-Logik sind sehr ähnlich zu Anfragen in klassischen logikbasierten Programmiersprachen wie z.B. Prolog. Die Anfrage ist dabei eine Regel, die »falsch« im Regelkopf hat und freie Variablen enthält. Eine Anfrage zu beantworten bedeutet dabei herauszufinden, welche Werte von Variablen die Anfrageregel und die Ontologie unerfüllbar machen.

Beispielsweise kann die Anfrage nach Autoren und ihren zugehörigen Artikeln an die Ontologie aus Abbildung 1 in F-Logik wie folgt repräsentiert werden:

```
FORALL X,Y
<- X:Autor["hat-geschrieben"->>Y]
```

Es wird schnell klar, dass F-Logik-Anfragen auf Ontologien angewendet werden können, aber als Ergebnis eine Relation (eine Menge von Tupeln) erzeugen. Damit lassen sich die Anfragen in F-Logik nicht einfach kombinieren – das Ergebnis einer Anfrage kann nicht als Eingabe für andere Anfragen benutzt werden. Damit leiden die Anfragen in F-Logik unter denselben Problemen wie RQL-Anfragen: Die Semantik des Modells wird durch die Anfragen zerstört.

5.5 Beschreibungslogiken

Beschreibungslogiken bieten einen sehr interessanten Ansatz für die Definition semantischer Anfragen. Die Möglichkeit, Ontologien anzufragen, ist im Kern der Modellierungssprache eingebaut, da man definierte Begriffe als gespeicherte Anfragen verstehen kann. Ein Begriffausdruck schreibt die Bedingungen vor, die für die Mitgliedschaft im jeweiligen Begriff erfüllt werden müssen. Das Interessante an diesem Einsatz ist, dass Anfragen gar nicht von anderen Modellierungsprimitiven unterschieden werden können. Damit ist automatisch gesichert, dass die Ergebnisse einer Anfrage als Eingabe für weitere Anfragen verwendet werden können. Das Problem der Zerstörung der Semantik gibt es bei Beschreibungslogiken deshalb auch nicht. Das Ergebnis einer Anfrage ist ein Begriff, der eine wohldefinierte semantische Interpretation hat. Anfragen in einer Beschreibungslogik sind jedoch ziemlich unterschiedlich von allen bisher vorgestellten Anfragesprachen. Beispielsweise kann man alle Autoren und ihre zugehörigen Artikel nicht auf einmal in einer Anfrage erhalten. Dies kann nur in zwei Schritten erfolgen. Im ersten Schritt muss man alle Autoren anfragen. Dies geschieht durch eine sehr einfache Anfrage, die alle Instanzen des Begriffs *Autor* aufzählt. Danach kann man für jeden Autor die entsprechende Menge der Artikel anfragen. Auf den ersten Blick scheint ein solcher Anfrageansatz ausdrucksschwächer zu sein als andere Anfragesprachen, was auch teilweise stimmt. Der zugrunde liegende Zweck jeder Anfrage ist das Aufrufen von Daten aus einem Modell. Damit das Ergebnis der Anfrage innerhalb des Modells bleibt, bieten unterschiedliche Anfragesprachen die Möglichkeit, diese Ergebnisse umzustrukturieren. Dieses Umstrukturieren führt jedoch oft dazu, dass die ursprüngliche Semantik der Daten verloren geht. Die Beschreibungslogiken bieten daher nur die Möglichkeit, bestimmte Daten wiederaufzurufen, die dann als Einstiegspunkt für die Navigation im Modell dienen. In unserem einfachen Beispiel kann man zeigen, wie die Primitive gut mit der Semantik des Modells harmonieren: Um alle Autoren und alle Artikel zu selektieren, verwendet man die Primitive in der natürlichen Reihenfolge:

```
I := Menge aller Instanzen von Autor
für jedes i ∈ I
   A := Menge aller Instanzen die
        durch hat-geschrieben
        mit i verknüpft sind
   für jedes a ∈ A
        bearbeite a
   Ende
Ende
```

Die Manipulation eines semantischen Modells erfolgt so auf natürliche Art und Weise. Problematisch dabei ist jedoch, dass Beschreibungslogiken sich nur auf die Modellierung von Daten und nicht auf die Manipulation von Datenmengen konzentrieren. Deswegen sind die Filterungsmöglichkeiten, die mit Beschreibungslogiken gemacht werden können, sehr beschränkt. So wäre es in den meisten Beschreibungslogiken nicht möglich, alle Autoren, die älter als 35 Jahre sind, anzufragen. Die Einführung solcher Primitive in Beschreibungslogiken würde dazu führen, dass die Berechnung der Subsumption von Begriffen nicht mehr entscheidbar ist.

6 Hybride Ontologierepräsentation und -anfrage – der Karlsruher Ansatz

In diesem Abschnitt stellen wir unseren Ansatz zur hybriden Ontologierepräsentation und -anfrage vor [Maedche et al. 2003]. Beim Entwurf dieses Ansatzes haben wir versucht, die gewünschten Eigenschaften existierender Ontologierepräsentationssprachen zu kombinieren, um die jeweiligen negativen Eigenschaften zu kompensieren. Dieser Ansatz ist auch in einem Ontologiewerkzeug umgesetzt worden und als Open Source Software verfügbar[1].

1. http://kaon.semanticweb.org

6.1 Repräsentation von KAON-Ontologien

Unsere Ontologiesprache folgt den bisher präsentieren Ansätzen und bietet Möglichkeiten zur Modellierung von Begriffen, Beziehungen und Instanzen. Zusätzlich weist unser Ansatz bestimmte Besonderheiten auf, die wir als notwendig zur Realisierung konkreter Anwendungen betrachten: Zum einen kann die Semantik der Beziehungen weiter spezifiziert werden. So ist es möglich, eine Eigenschaft als symmetrisch oder als transitiv zu bezeichnen. Dies bietet eine größere Freiheit beim Modellieren. Beispielsweise beinhaltet eine typische Ontologie über Dokumente eine Menge Themen, die man zum jeweiligen Dokument zuordnen kann. Typischerweise sind die Themen in eine partielle Ordnung strukturiert – manche Themen (wie z.B. *Programmieren*) sind allgemeiner als andere Themen (wie z.B. *Java*) und dadurch entsteht eine Hierarchie von Themen. Da man die Begriffe in einer Hierarchie anordnen kann, ist es auf den ersten Blick offensichtlich, die Themen als Begriffe zu modellieren. Damit ergibt sich aber das folgende Problem: Wenn *Programmieren* ein Begriff ist, was sind dann seine Instanzen? Prinzipiell gibt es keine natürlichen Instanzen dieses Begriffs, was darauf hindeutet, dass eigentlich *Programmieren* eine Instanz des Begriffs *Thema* sein sollte. Die partielle Ordnung zwischen Themen kann man mittels transitiver Beziehung *ist-Unterthema-von* modellieren. Damit ist das Modell semantisch korrekt.

Des Weiteren zwingen wir den Modellierer nicht, die Welt in Begriffe und Instanzen streng zu teilen. Wir haben mehrere Beispiele gefunden, in welchen eine derartige Modellierung nötig ist. Nehmen wir an, dass wir die Beziehung zwischen Spezies und einer bestimmten Spezies – Affen und konkreten Affen, wie Nkima – beschreiben möchten. Um dies zu modellieren, kann man *Spezies* als ein Begriff und *Nkima* als eine Instanz modellieren. Dabei ist es nicht klar, wie man das Symbol *Affe* modellieren soll. Auf der einen Seite ist der Affe eine bestimmte Spezies und es wäre korrekt, *Affe* als eine Instanz von *Spezies* zu modellieren. Auf der anderen Seite wäre es korrekt, *Affe* als ein Begriff zu modellieren, der eine Menge aller Affen repräsentiert. Dann könnte

man sagen, dass *Nkima* eine Instanz von *Affe* ist. Also, der Begriff *Affe* scheint die Eigenschaften eines Begriffs und einer Instanz gleichzeitig aufzuweisen.

Dieses Problem kann mit der Einführung der Prädikatenlogik der zweiten Ordnung gelöst werden. In der Prädikatenlogik der zweiten Ordnung kann man Variablen anstelle von Prädikatennamen verwenden. Dies ermöglicht Fragen zu stellen, wie z.B.: »Welche *Spezies* hat *Nkima* als Instanz?« Damit werden aber mehrere neue Probleme geschaffen, da die Prädikatenlogik der zweiten Ordnung nur sehr schwer bearbeitet werden kann und viele der existierenden Algorithmen unentscheidbar sind. Um solche Probleme zu vermeiden, bleiben wir in der Prädikatenlogik der ersten Ordnung, aber mit einer Syntax der zweiten Ordnung. Jedes Symbol kann damit als eine Instanz, ein Begriff oder eine Beziehung interpretiert werden. Die Interpretation hängt von dem jeweiligen Kontext ab, wobei zu jedem Zeitpunkt die Interpretationssicht geändert werden kann. Eine solche Ontologie ist in Abbildung 4 dargestellt.

Das Beispiel kann folgendermaßen interpretiert werden: *Affe* ist eine *Spezies*. Für diese Spezies ist es bekannt, dass ihr Habitat der Urwald ist. Gleichzeitig kann *Affe* als ein Begriff interpretiert werden. *Nkima* ist ein konkreter Affe, der fünf Jahre alt ist.

Der dritte Unterschied zwischen unserem Ontologiemodellierungsansatz und den Beschreibungslogiken ist, dass bei uns Domänen und Ranges als Konsistenzbedingungen interpretiert werden. In Beschreibungslogiken sind Domänen und Ranges Inferenzregeln. Falls unser Beispiel als eine reine Beschreibungslogikontologie interpretiert werden würde, könnte man die folgende Schlussfolgerung ziehen: Falls irgendeine Instanz das Attribut *hat-Alter* aufweist und da die Domäne von *hat-Alter* der Begriff *Affe* ist, kann schlussgefolgert werden, dass diese Instanz eine Instanz des Begriffs *Affe* sein muss. Dies ist wesentlich anders als bei üblichen objektorientierten Modellen: Da *Affe* in der Domäne von *hat-Alter* ist, bedeutet das, dass man das *hat-Alter*-Attribut nur für Instanzen von *Affe* verwenden darf. Aufgrund unserer Erfahrung konnten wir feststellen, dass die meisten Benutzer die Konsistenzbedingungssemantik für die Domänen und Ranges erwarten, und deswegen verfolgen wir diesen Ansatz.

6.2 Anfrage von KAON-Ontologien

Um auf Ontologien zugreifen zu können, bietet KAON eine Anfragesprache, die stark vom Beschreibungslogikparadigma beeinflusst wurde. Die Aufzählung von Begriffen und die Navigation sind dabei zwei komplementäre Primitive, die einen effizienten und intuitiven Zugriff auf Ontologien ermöglichen. Dabei wurde das grundlegende Paradigma von Beschreibungslogiken mit zusätzlichen Fähigkeiten ausgestattet, wie z.B. die Möglichkeit, Schemaanfragen zu formulieren oder Mechanismen zum besseren Selektieren von Daten. Wie schon diskutiert, harmoniert unserer Ansicht nach das Aufteilen von Datenzugriffprimitiven in Anfragesprache und Modellnavigationsprimitive gut mit semantischen Modellen. Um auf die Daten zugreifen zu können, verwendet der Benutzer die vorhandenen Primitive in genau der Reihenfolge, die mit der Struktur des Modells übereinstimmt. Dieses Vorgehen hat jedoch einen Nachteil: Die Reihenfolge der Datenzugriffe ist vom Benutzer bestimmt und damit geht die Möglichkeit der Anfrageoptimierung verloren. Um beispielsweise alle Autoren und die von ihnen geschriebenen Dokumente aus einer Ontologie zu lesen, wird erst eine Anfrage nach allen Autoren gestellt und dann in einer verschachtelten Schleife für jeden Autor eine Anfrage nach Dokumenten erzeugt. Im obigen Beispiel kann man beobachten, dass bei n Autoren insgesamt n+1 Anfragen erzeugt werden. Ein derartiger Datenzugriff führt somit zu erheblichen Performance-Problemen. Dagegen können die notwendigen Daten aus relationalen Datenbanken durch nur einen Join gelesen werden. Da alle notwendigen Daten nur mit einer Anfrage geladen werden, hat die Datenbank die Möglichkeit, den optimalen Datenzugriffsplan zu erzeugen. Diese Eigenschaft wird im Allgemeinen als der entscheidende Vorteil von relationalen Datenbanken zu hierarchischen Datenbanken betrachtet.

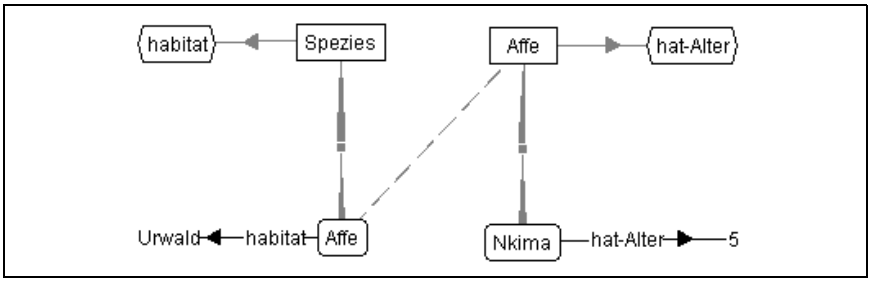

Abb. 4: Das Affenbeispiel

Diese zwei Anforderungen scheinen inkompatibel zu sein: Auf der einen Seite soll der Zugriffspfad der Struktur des Modells folgen, während auf der anderen Seite man aus Performanzgründen so viele Daten wie möglich auf einmal anfragen können soll. In KAON haben wir versucht, einen Mittelweg zu finden. Mit jeder Hauptanfrage ist es möglich, zusätzliche Anfragen zu spezifizieren. Dabei sind nur die Ergebnisse der Hauptanfrage direkt zugreifbar. Die zusätzlichen Anfragen bestimmen die Menge der Informationen, die in der näheren Zukunft auch gebraucht werden. Diese Informationen gehören nicht zu dem Anfrageergebnis und sind erst durch Navigationsprimitive zugreifbar. Zum Beispiel kann man alle Autoren und alle geschriebenen Dokumente in KAON folgendermaßen effizient laden:

```
Autor
[hat-geschrieben IN:1 this]
```

Die erste Zeile der Anfrage ist die Hauptanfrage, deren Ergebnisse die Ergebnisse der gesamten Anfrage darstellen. Die zweite Zeile ist eine zusätzliche Anfrage, die spezifiziert, dass *hat-geschrieben* auch von Interesse ist (`IN:1 this` dient dazu, dass nur diese *hat-geschrieben*-Beziehungen geladen werden, bei denen sich das erste Element in der Ergebnismenge der Hauptanfrage befindet). Der Anfrageprozessor kann die zusätzlichen Anfragen komplett ignorieren – das System wird zwar langsamer, aber korrekt funktionieren. Ein fortgeschrittener Anfrageprozessor wird diese Anfrage mittels Query-Unnesting-Strategien auswerten, d.h., die zusätzliche Anfrage kann als in die Hauptanfrage verschachtelt betrachtet werden. Um diese zwei Anfragen

zu bearbeiten, verwendet der Anfrageprozessor Query Unnesting. Wenn die Ergebnisse der Anfrage berechnet sind, muss der Anfrageprozessor die Ergebnisse wieder korrekt verschachteln.

Die zweite wichtige Eigenschaft von KAON-Anfragen ist die Möglichkeit, das »Schema« anzufragen. Zum Beispiel kann man alle Unterbegriffe eines Begriffs mit folgender Anfrage auslesen:

```
SUBCONCEPTS(!#Autor!)
```

Diese Anfrage weist einige Sonderheiten auf. Zum einen ist es eine Anforderung, dass die Anfrage als Ergebnis eine Menge von Begriffen und nicht eine Menge von Instanzen zurückliefern soll. Zum anderen ist `!#Autor!` eine Konstante, die eine Instanz *Autor* spezifiziert, und es ist nicht klar, warum man die Unterbegriffe einer Instanz auslesen soll. Da unsere Anfragesprache durch Beschreibungslogiken motiviert wurde, folgen wir strikt dem Prinzip, dass das Ergebnis einer Anfrage immer eine Menge von Instanzen sein muss. Um die Begriffe als Ergebnis liefern zu können, verwendet die obige Anfrage einen Trick: Das Ergebnis der Anfrage enthält nicht die eigentlichen Begriffe, sondern die Instanzen mit den gleichen Namen wie die entsprechenden Begriffe. Ähnlich ist das Argument der Anfrage nicht der Begriff, sondern die Instanz mit dem gleichen Namen wie der eigentliche Begriff. Die Funktion `SUBCONCEPTS` macht dann Folgendes: Sie selektiert eine oder mehrere Instanzen, interpretiert sie als Begriffe, stellt alle Unterbegriffe fest, interpretiert sie als Instanzen und gibt dann diese Menge endgültig zurück. Damit können Schemaanfragen leicht verkettet werden. Beispielsweise kann man alle Begriffe, die sich zwei Ebenen unter dem Begriff *Autor* befinden, folgendermaßen anfragen: (`SUBCONCEPTS^` ist die Funktion, die nur die direkten Unterbegriffe berechnet):

```
SUBCONCEPTS^(SUBCONCEPTS^(!#Autor!))
```

Bei der Realisierung der Anfragesprache haben wir versucht, den Ansätzen deduktiver Datenbanken weitgehend zu folgen. Die dahinter liegende Idee dabei war, auf den bestehenden Arbeiten zur effizienten Evaluierung von rekursiven Anfragen aufzubauen. Die komplette Ontologie mit zugehörenden Instanzen wird in einer relationalen Datenbank gespeichert. So ist es möglich, das System skalierbar zu machen. Im Falle einer größeren Ontologie (mit z.B. 10^6 Instanzen) wäre es verheerend, den kompletten Datensatz im Hauptspeicher zu verwalten. Die Anfragen in der Beschreibungslogiksyntax werden als Erstes in Datalog übersetzt. Die Grundlagen für die Übersetzung nach Prädikatenlogik der ersten Ordnung sind in [Borgida 1996] gegeben. Dabei wurde die Übersetzung nach Datalog weiter ausgebaut. Das entstehende Datalog-Programm wird dann mittels Magic Sets Transformation [Beeri & Ramakrishnan 1987] ausgeführt. Die Magic Sets Transformation sorgt dafür, dass nur die relevanten Daten aus der Datenbank gelesen werden müssen. Dabei fügt die Magic Sets Transformation für jedes berechnete Prädikat zu dem Programm ein magisches Prädikat, in dem die Menge für die Anfrage relevanten Fakten berechnet werden soll, hinzu. Die Regeln des Originalprogramms werden mit magischen Prädikaten angereichert, um zu vermeiden, dass die Daten, die für die Anfrage nicht relevant sind, bearbeitet werden müssen. Die Magic Sets Transformation hat sich als eine Methode erwiesen, die allgemein und vergleichsweise einfach realisiert werden kann.

Für den Fall, dass das Datalog-Programm Negation enthält, muss besonders vorgegangen werden. Es ist bekannt, dass wenn das Originalprogramm stratifiziert ist (das bedeutet, dass die rekursiven Abhängigkeiten zwischen Prädikate nicht durch die Negation gehen), das transformierte magische Programm nicht stratifiziert sein muss. In diesem Fall verwenden wir die Ergebnisse aus [Kemp et al. 1991], um das wohlfundierte Modell des Programms mittels eines alternierenden Fixpunktoperators zu berechnen.

7 Fazit

Obwohl der Nutzen semantischer Modelle allgemein anerkannt wird, mangelt es jedoch an einer allgemein akzeptierten und standardisierten Repräsentations- und Anfragesprache für semantische Modelle. In diesem Artikel haben wir dem Leser eine Einführung und Übersicht über den aktuellen Stand im Bereich Repräsentations- und Anfragesprachen für Ontologien gegeben. Des Weiteren wurde kurz die Rolle von Ontologien in der Informationsintegration diskutiert. Wir haben dabei argumentiert, dass Ontologien sowohl bei der Quellenbeschreibung als auch bei der Modellierung des integrierten Modells klare Vorteile bringen. Schließlich wurde auch noch der Karlsruher Ansatz zur hybriden Ontologierepräsentation und -anfrage vorgestellt. Beim Entwurf dieses Ansatzes haben wir versucht, die gewünschten Eigenschaften existierender Ontologierepräsentationssprachen zu kombinieren, um die jeweiligen negativen Eigenschaften zu kompensieren.

8 Literatur

[Abiteboul et al. 1995] *Abiteboul, S.; Hull, R.; Vianu, V.:* Foundations of Databases. Addison-Wesley, 1995, ISBN: 0-201-53771-0.

[Beeri & Ramakrishnan 1987] *Beeri, C.; Ramakrishnan, R.:* On the power of magic. In: Proc. ACM SIGACTSIGMOD-SIGART Symposium on Principles of Database Systems, pp. 269–284, 1987.

[Berners-Lee et al. 2001] *Berners-Lee, T.; Hendler, J.; Lassila, O.:* The Semantic Web. Scientific American, May 2001.

[Borgida 1992] *Borgida, A.:* Description logics are not just for the FLIGHTLESS-BIRDS: A new look at the utility and foundations of description logics. DCS-TR-295, Department of Computer Science, Rutgers University, 1992.

[Borgida 1996] *Borgida, A.:* On the relative expressiveness of description logics and predicate logics. Artificial Intelligence, Vol. 82(1-2), pp. 353-367, 1996.

[Brachman & Schmolze 1985] *Brachman, R. J.; Schmolze, J.:* An Overview of the KL-ONE Knowledge Representation System. In: Cognitive Science, Vol. 9, 171-216, 1985.

[Brickley & Guha] *Brickley, D.; Guha, R. V.:* RDF Vocabulary Description Language 1.0: RDF Schema. *http://www.w3.org/TR/rdf-schema/*

[Cattell et al. 2000] *Cattell, R. G. G.; Barry, D. K.; Catell, R.; Berler, M.; Eastman, J.; Jordan, D.; Russell, C.; Schadow, O.; Stanienda, T.; Velez, F.:* The Object Data Standard: ODMG 3.0. Morgan Kaufmann, 2000, ISBN 1558606475.

[Fitting 1996] *Fitting, M.: First*-Order Logic and Automated Theorem Proving. 2nd Edition, Springer-Verlag, 1996, ISBN: 0-387-94593-8.

[Gogolla & Hohenstein 1991] *Gogolla, M.; Hohenstein, U.:* Towards a Semantic View of an Extended Entity-Relationship Model. In: ACM Transactions on Database Systems (TODS), Volume 16, pp. 369–416, ACM Press, New York, NY, USA, 1991.

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

[Karvounarakis et al. 2002] *Karvounarakis, G.; Alexaki, S.; Christophides, V.; Plexousakis, D.; Scholl, M.:* RQL: A Declarative Query Language for RDF. In: Proceedings of the 11th International Conference on the WWW, Hawaii, 2002.

[Kemp et al. 1991] *Kemp, D. B.; Srivastava, D.; Stuckey, P. J.:* Magic Sets and Bottom-Up Evaluation of Well-Founded Models. In: Proceedings of the 1991 International Symposium on Logic Programming, pp. 337–354, MIT Press, San Diego, USA, 1991.

[Kifer et al. 1995] *Kifer, M.; Lausen, G.; Wu, J.:* Logical Foundations of Object-Oriented and Frame-Based Languages. In: Journal of the ACM, Vol. 42, pp. 741-843, July 1995.

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

[Ludäscher et al. 2001] *Ludäscher, B.; Gupta, A.; Martone, M. E.:* Model-Based Mediation with Domain Maps. In: Proceedings of the 17th Intl. Conference on Data Engineering (ICDE), Heidelberg, Germany, IEEE Computer Society, April 2001.

[Maedche et al. 2002] *Maedche, A.; Motik, B.; Silva, N.; Volz, R.:* MAFRA – Mapping Distributed Ontologies in the Semantic Web. In: Proceedings of the 13th European Conference on Knowledge Engineering and Management, EKAW-2002, Springer-Verlag, LNAI, Madrid, Spain, 2002.

[Maedche et al. 2003] *Maedche, A.; Motik, B.; Stojanovic, L.:* Managing Multiple and Distributed Ontologies in the Semantic Web. To appear in: VLDB Journal, Special Issue on the Semantic Web, Springer-Verlag, 2003.

[Pan & Horrocks 2001] *Pan, Jeff Z.; Horrocks, I.:* Metamodeling Architecture of Web Ontology Languages. In: Proceedings of the First Semantic Web Working Symposium (SWWS01), pp. 131-149, July 2001

[Papakonstantinou et al. 1995] *Papakonstantinou, Y.; Garcia-Molina, H.; Widom, J.:* Object Exchange across Heterogeneous Information Sources. In: Proceedings of the 11th Conference on Data Engineering, pp. 251-260, IEEE Computer Society, Taipei, Taiwan, 1995.

[Patel-Schneider et al. 2002] *Patel-Schneider, P. F.; Hayes, P.; Horrocks, I.; van Harmelen, F.:* Web Ontology Language (OWL) Abstract Syntax and Semantics, 2002, *http://www.w3.org/TR/owl-semantics/*

[Pepper & Moore] *Pepper, S.; Moore, G. (Eds.):* XML Topic Maps (XTM) 1.0, *http://www.topicmaps.org/xtm/1.0/*

[Schmidt-Schauß & Smolka 1991] *Schmidt-Schauß, M.; Smolka, G.:* Attributive concept descriptions with Complements. In: Artificial Intelligence, Vol. 43, pp. 1-26, 1991.

Alexander Maedche ist seit 2001 Leiter des Forschungsbereichs Wissensmanagement (WIM) am Forschungszentrum Informatik (FZI) der Universität Karlsruhe. Seine Forschungsinteressen liegen in den Bereichen der konzeptuellen und semantischen Datenmodellierung sowie des Maschinellen Lernens. Alexander Maedche erhielt im Jahr 1999 sein Diplom im Bereich Wirtschaftsingenieurswesen (Informatik / Operations Research) an der Universität Karlsruhe und promovierte im Jahr 2001 in Angewandter Informatik ebenfalls an der Universität Karlsruhe.

Boris Motik arbeitet als wissenschaftlicher Mitarbeiter im Bereich »Wissensmanagement« (WIM) am Forschungszentrum Informatik an der Universität Karlsruhe. Seine Forschungsinteressen beinhalten konzeptuelle Modellierungssprachen-, konzeptuelle Anfragesprachen sowie Transformationen zwischen konzeptuellen Modellen. Boris Motik hat sechs Jahre Erfahrung bei unterschiedlichen Industrieprojekten in Themen wie z.B. E-Business oder Multimediasysteme gesammelt. Von der Fakultät für Elektrotechnik und Informatik an der Universität Zagreb, Kroatien erhielt er im Jahre 1996 ein Diplom in Elektrotechnik sowie 1999 einen Masters in Informatik.

Alexander Maedche
Boris Motik
Universität Karlsruhe
FZI – Forschungszentrum Informatik
Haid-und-Neu-Straße 10-14
76131 Karlsruhe
{maedche, motik}@fzi.de
http.//www.fzi.de/wim