Ein Ansatz für eine integrative Ontologie-Beschreibungssprache

von Dipl.-Inform. Andreas Billig
aus Berlin

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Professor Dr. Hans-Ulrich Heiß
Gutachter: Professor Dr. Herbert Weber
Gutachter: Professor Dr. Bernd Mahr

Tag der wissenschaftlichen Aussprache: 5. Oktober 2007

Berlin 2008
D83

FÜR MUTTER UND VATER

Zusammenfassung

Moderne Ontologie-Beschreibungssprachen basieren auf den klassischen Ansätzen F-Logic und Description Logic. Obwohl all diese Ansätze gemeinsame Charakteristika besitzen, verfügen sie jedoch über teilweise sehr unterschiedliche Konstruktmengen, welche bezüglich folgender Aspekte untersucht werden können: logische Höherstufigkeit, Konstrukte zur Kontextualisierung, Konstrukte zur Kategorisierung und Konstrukte für Regeln. Der erste Aspekt bezieht sich auf die Möglichkeit der Sprache, alle Basiselemente der Modellierung als eigenständige Objekte (first class citizens) zu behandeln. Erst eine derartige Gleichbehandlung läßt es zu, den gesamten Elementraum der Modellierung den Anfrage- und Inferenzkomponenten zur Verfügung zu stellen. Konstrukte zur Kontextualisierung gestatten es, Begriffe oder auch Beziehungssetzungen zwischen diesen in einen Kontext zu stellen. Eines der speziellen Konstrukte hierfür ist die sogenannte taxonomische Kontextualisierung, welches die Auflösung mehrfacher Generalisierung erlaubt. Der dritte Aspekt betrifft die Fähigkeit der Sprache, taxonomische Kontextualisierungen den Attributen von Instanzen zuzuordnen. Schließlich bezieht sich der letzte Aspekt auf die Angabe von Regeln über den gesamten Modellraum, welches sowohl die Definition einer maßgeschneiderten Semantik als auch die Definition von Anwendungsregeln vergleichbar mit denen der klassischen Wissensrepräsentation erlaubt. Die vorliegende Arbeit untersucht die relevanten Ontologie-Beschreibungssprachen bezüglich der aufgeführten Aspekte. Jede der untersuchten Sprachen unterstützt nicht vollständig die geforderte Konstruktmenge. Aufbauend auf diesen Untersuchungen wird hier ein Ansatz vorgestellt, der

- eine Integration der bestehenden Ansätze zur Unterstützung der geforderten Konstruktmenge
- eine Erweiterung um die taxonomische Kontextualisierung und Kategorisierung
- eine Integration von höherstufigen Regeln sowie
- eine Definition der axiomatischen Semantik der resultierenden Ontologie-Beschreibungssprache

vornimmt.

Abstract

Modern ontology languages are based on the two classic main directions of F-Logic and description logics. Although these languages share common characteristics they partly contain very different construct sets which can be examined related to the following aspects: logical higher-orderness, constructs for contextualization, constructs for categorization, and constructs for rules. The first aspect concerns the facility of the language to treat all basic elements as so-called first class citizens. The equal treatment of these elements allows for querying and inferencing over the complete modelling space as a whole. Constructs for contextualization enable to set terms as well as relationships in a context. One of the special context constructs is the so-called taxonomic contextualization which allows for disambiguating taxonomies containing multiple inheritance. The third aspect concerns the facility of the ontology language to relate taxonomic contextualizations to attributes of artefact individuals. The last aspect deals with constructs for defining rules over the whole model element space which play an important role to define customizable semantics of ontology constructs as well as to specify application rules similarly to those of knowledge representation tasks. The thesis studies all relevant ontology languages concerning the aspects described above. Every studied language supports only parts of the whole required construct set. Based on these studies the thesis developed an approach which aimed at

- integrating existing languages in order to support the required construct set
- the extension by taxonomic contextualization and categorization
- the integration of higher-order rules
- the definition of the axiomatic semantics of the resulting language

Danksagung

Mein Dank gilt zuvorderst meinen beiden Gutachtern Prof. Bernd Mahr und Prof. Herbert Weber. Bernd Mahr lehrte mich auf seine unnachahmliche Art und Weise das wissenschaftliche Denken als eine mathematisch- und philosophisch-logische Disziplin, welches eine unabdingbare Voraussetzung für die Erreichung der Ziele dieser Arbeit war. Seine Gabe, komplexe Probleme auf essentielle Fragestellungen zu reduzieren und sie als Forschungsfragen zu formulieren, wird mir weiterhin ein Vorbild sein. Herbert Weber bot mir die ausserordentliche Unterstützung, diese Art der wissenschaftlichen Auseinandersetzung im Rahmen seines Instituts an pragmatischen Zielsetzungen zu brechen. Seine Gabe, die Rolle eines richtungweisenden Gestalters auszufüllen und seine geistige Kraft, haben mich überdurchschnittlich angespornt. Prof. Kurt Sandkuhl hat mich in der Rolle eines klugen, beständigen und wegweisenden Mentors begleitet. Ohne ihn wäre diese Arbeit nicht möglich gewesen – so hat er doch dank seines immerwährenden Forschergeistes mich in grossem Masse motiviert und den Weg bereitet. Bei Dr. Ralf Kutsche möchte ich mich für die besondere Unterstützung bedanken, meine Brücke zur Wissenschaft aufrechterhalten zu haben, was ohne sein Zutun ein schweres Unterfangen gewesen wäre. Auch seine fachliche Herangehensweise war derart, dass er mir eine gedankliche Heimat bot. Dr. Andreas Leicher und Dr. Susanne Busse haben mit zu dem Gelingen der Arbeit beigetragen. Die fachlichen Diskussionen und die wissenschaftliche Zusammenarbeit mit ihnen haben die Arbeit besonders positiv beeinflusst. Eine hervorzuhebene Rolle haben die Kollegen vom Fraunhofer ISST eingenommen, die mir gestatteten, meine wissenschaftlichen Fragestellungen in zumeist anwendungsorientierten Projekten beizubehalten. Die zahlreichen Diskussionen und die Ermutigung, die ich erfahren habe, waren von grossem Wert. Zuletzt, aber von seiner Bedeutung her unter den vordersten, möchte ich meinem Freundeskreis[1] und meinen Eltern danken. Deren nie versiegte Anteilname war von unschätzbarem Wert.

[1] Danke Anne, danke Markus, für das ergiebige Korrekturlesen und die fruchtbaren Anregungen.

Inhalt

1 Einleitung

1.1 Hintergrund und Zielsetzung der Arbeit

Die letzten Jahre der Entwicklung in der Informatik, insbesondere das Entstehen der weltweiten Vernetzung, haben in hohem Maße Möglichkeiten zur Bereitstellung und Nutzung von Informationen aus den unterschiedlichsten Domänen geschaffen. Auf technologischer Seite hat – zur Unterstützung der neuen Vorhaben – eine Wiederbelebung und Weiterentwicklung einer Vielzahl von Ansätzen aus dem Bereich der ontologiebasierten Modellierung stattgefunden.

Die Ansätze der ontologiebasierten Modellierung lassen sich in zwei Hauptrichtungen unterteilen. Zum einen in die Ansätze, die mit der Welt der klassischen konzeptuellen Modellierung vergleichbar sind, wie sie im Bereich der Datenbanken und objektorientierten Modellierung vorkommt, und zum anderen in die Ansätze, die in der Formalisierung von terminologischen Systemen ihren Ursprung haben. Die Ansätze der ersten Richtung folgen der *frame-basierten* Modellierung, welche ihre Fundierung in der sogenannten *F-Logic* erfahren haben. Die Ansätze der zweiten Richtung basieren auf einer Logik zur Beschreibung von Begriffssystemen, der sogenannten *Description Logic*.

Die einzelnen Sprachen verfügen, wenngleich sie gemeinsame Ziele haben, über zum Teil sehr unterschiedliche Konstrukte. Für den durch die Konstruktmenge definierten Sprachumfang der einzelnen Ansätze lassen sich die folgenden Aspekte aufführen.

- syntaktische Höherstufigkeit
- Konstrukte zur Kontextualisierung und Modularisierung
- Konstrukte zur Kategorisierung
- Konstrukte zur Angabe von Regeln

- Entscheidbarkeit der Konstrukte

Der erste Aspekt bezieht sich auf die Möglichkeit der Sprache, die Basiselemente zur Modellierung als eigenständige Objekte (*first class citizens*) zu behandeln. Dazu gehören nicht nur Klassen, sondern auch deren Attribute und Beziehungen untereinander. Eine Gleichbehandlung dieser Elemente zusammen mit den Instanzen (und den Instanzen von Instanzen) eröffnet die Möglichkeit, den gesamten Elementraum der Modellierung den Anfrage- und Inferenzkomponenten zur Verfügung zu stellen.

Die syntaktische Höherstufigkeit ist nicht nur von Interesse im Zusammenhang mit der Metamodellierung, sondern auch von Interesse bei der unterschiedlichen Verwendung der Termini. Ein Terminus kann die Rolle einer mit Attributen ausgestatteten Klasse annehmen, bei der die Extensionsbildung vorrangiges Ziel ist, oder die Rolle eines Individuums, der dem Attribut einer Klasse zugewiesen wird. Ohne eine Höherstufigkeit ist man nicht in der Lage, Termini zugleich in der Klassenrolle als auch in der Individuenrolle zuzulassen.

Konstrukte zur Kontextualisierung gestatten es, Begriffe oder auch Beziehungssetzungen zwischen diesen in einen Kontext zu stellen. Neben der klassischen Begriffskontextualisierung zur Auflösung von Homonymen ist die Kontextualisierung von Begriffen zur Auflösung mehrfacher Generalisierung hervorzuheben, genauer: ist ein Element Subklasse oder Instanz von mehreren Klassen, so kann eine Superklasse zur Kontextualisierung herangezogen werden (taxonomische Kontextbildung). Die Kontextualisierung von Beziehungssetzungen dient im wesentlichen zur Relativierung von Aussagen: gleiche Aussagen müssen nicht in allen Kontexten gelten. Bei der Modularisierung von Ontologien handelt es sich um diese einfache Art der Kontextualisierung von Beziehungssetzungen. Ist die Kontextualisierung von Beziehungssetzungen parametrisch, so sind damit Transformationen ausdrückbar.

Kategorisierungen stellen eine Zuordnung von Termini zu Individuen dar. Meist beschränkt sich die Auswahl der Termini auf alle diejenigen, die sich in Spezialisierungsbeziehung zu einer ausgewählten Wurzel einer Taxonomie befinden. Nicht nur einzelne Termini, sondern auch die Kontextualisierung dieser Termini können zur Kategorisierung herangezogen werden. Eine Erweiterung der klassischen Signaturkonstrukte um die taxonomische Kategorisierung ist dafür erforderlich.

Der Einsatz von logischen Regeln sorgt für eine präzise Definition von logischen Abhängigkeiten zwischen den Elementen von Ontologien. Die Nutzung von Regeln ist dabei auf zwei unterschiedlichen Ebenen zu beobachten. Zum einen läßt sich die Semantik der abstrakteren ontologiebildenden Konstrukte mit Hilfe von Regeln in präziser Art und Weise beschreiben.

Ein Beispiel dafür wäre die Optionalität eines Klassenattributs. Zum anderen kann die Formulierung von Regeln auf der Anwendungsebene erfolgen, wie es bei dem praktischen Einsatz von Systemen zur Wissensrepräsentation zahlreich vorkommt.

Der letzte Aspekt hat mit der prinzipiellen Anwendbarkeit einer Ontologie-Beschreibungssprache zu tun. Im Gegensatz zur Programmierung, sei es imperativ, funktional oder logisch, bei der ein Anwender der Sprache gegebenenfalls für die Termination einer Anwendung sorgen muß, ist es bei der Modellierung eher notwendig, diese Eigenschaft für aller Sätze der Sprache zuzusichern, da von ihrem Wesen her die konzeptuelle und damit auch die ontologiebasierte Modellierung deklarativen Charakter besitzt.

Jeder dieser Ansätze zur ontologiebasierten Modellierung, seien es Ansätze innerhalb der Richtungen oder die Ansätze, welche ein Zusammenführen der beiden Richtungen vornehmen, verfügen nicht vollständig über den eben beschriebenen Sprachumfang und erfüllen nicht alle sich daraus ergebenden Eigenschaften. Die Zielsetzung der Arbeit ist es nun, eine

- eine Integration der bestehenden Ansätze zur Unterstützung der grundlegenden Konstruktmenge
- darauf aufbauend die Ausstattung des Integrationsansatzes mit den oben beschriebenen, erweiterten Sprachkonstrukten und
- die Definition der formalen Semantik des Integrationsansatzes

vorzunehmen.

1.2 Gliederung der Arbeit

Kapitel 2 erörtert die einzelnen Vertreter der beiden Hauptrichtungen, welche sich wiederum in zwei Gruppen unterteilen lassen. Die erste Gruppe umfaßt die beiden Basissprachen, welche die zwei verschiedenen Hauptrichtungen konstituieren, nämlich *F-Logic* und *Description Logic* (DL). Neben der Einführung der grundlegenden Konstrukte wird die Semantik der Basissprachen erläutert. Dabei wird sowohl die modelltheoretisch orientierte Semantik umrissen als auch die axiomatische Semantik dargelegt. Die zweite Gruppe, deren Syntax und Semantik erörtert wird, besteht aus den Web-orientierten Sprachen. Dazu gehören der Metadatenstandard *Resource Description Framework* (RDF), die *Ontology Web Language* (OWL), das eine

Adaption der DL für das *Semantic Web* darstellt und die entsprechenden Erweiterungen um Regeln. Ferner zählt dazu TRIPLE, eine Transformations- und Regelsprache für RDF.

Kapitel 3 stellt die Anforderungen an eine integrative Ontologie-Beschreibungssprache auf. Diese Anforderungen beziehen sich auf den oben angesprochenen Sprachumfang und einiger Erweiterungen aus Sicht der praktischen Anwendung. Ein Vergleich der dort geforderten Konstrukte mit den in Kapitel 2 vorgestellten Konstrukten legt die Grundlage für die Konzeption der in Kapitel 4 vorgestellten Ontologie-Beschreibungssprache $\mathcal{I}$. Schließlich werden die Anforderungen an eine Axiomatisierungssprache, auf die $\mathcal{I}$ zum Zwecke der Definition der formalen Semantik in Kapitel 4 abgebildet wird, aufgestellt.

Kapitel 4 stellt die sich aus den Anforderungen ergebende Ontologie-Beschreibungssprache $\mathcal{I}$ vor. Die syntaktischen Komponenten bestehen im wesentlichen aus einer an F-Logic und TRIPLE angelehnten Regelsprache mit entsprechenden Signaturkonstrukten. Diese Regelsprache bildet die Grundlage für die syntaktisch abstraktere Form der sogenannten Ontologiedeklaration, mit denen Klassen im herkömmlichen objektorientierten Stil definiert werden können. Den Abschluß bzgl. der syntaktischen Komponenten bilden Einbettungen von DL-Programmen und Kollektionen von Ontologiedeklarationen, mit deren Hilfe die Gesamtontologie einer Anwendung spezifiziert wird. Schließlich wird in Kapitel 4 eine Definition der formalen Semantik von $\mathcal{I}$ vorgenommen, die in zwei Schritten erfolgt: die Abbildung von Ontologiedeklarationen und DL-Einbettungen auf die Regelsprache und deren anschließende Abbildung auf die Axiomatisierungssprache. Zum Abschluß des Kapitels wird ein Vergleich mit den bestehenden Ansätzen vorgenommen.

1.3 Typographische Konventionen

Die nachstehende Tabelle zeigt die unterschiedlichen typographischen Elemente, die in der vorliegenden Arbeit verwendet werden.

Kategorie	*Beispiel*
normaler Text	Verwendung unterschiedlicher Typographie ...
mathematische/ metasprachliche Symbole	$\hat{r}(x,c) := \{y \in \alpha(c) \mid (x,y) \in \alpha(r)\}$

formalsprachliche Elemente	***select*** *x* ***such that*** *x− value −> 99*
definierte Begriffe	Unter einer *Modellentität* versteht man ... (Bem.: Normalerweise wird nur beim ersten Auftreten des zu definierenden Begriffs die kursive Schreibweise verwendet. Handelt es sich um einen englisch-sprachlichen Begriff, der nicht übersetzt wird, so wird er meist durchgängig kursiv geschrieben)
Aussagen und Redewendungen (‚...')	Der angebotene Kurs hat als Thema ‚Alexander der Große als Feldherr'

2 Beschreibungssprachen für Ontologien

2.1 Basissprachen

2.1.1 F-Logic

Die prädikatenlogisch basierte Sprache *Frame-Logic* (kurz *F-Logic*) wurde von Kifer, Lausen und Wu eingeführt [KL89, KLW95]. Ihr Ziel war es, eine prädikatenlogische Fundierung der objektorientierten Programmierung und eine Integration von logischen und objektorientierten Konstrukten vorzunehmen. Ihr Name zeigt gleichzeitig eine starke Orientierung an den Konzepten für frame-basierte Sprachen an, wenngleich vorwiegend die Terminologie aus der objektorientierten Welt herangezogen wurde.

Die üblichen Basisbausteine zur Bildung von Modellen – wie Klassen, die Instanz-und Generalisierungs/Spezialisierungsbeziehung, Attribute und Signaturen – sind durch spezielle Konstrukte gegeben, welche direkt in die Sprache der Prädikatenlogik erster Stufe eingebettet werden können. Eine der hervorzuhebenden Eigenschaften ist die Behandlung der Basisbausteine als eigenständige Objekte (*first class citizens*), über die quantifiziert werden kann und die damit als Anfrageelemente zur Verfügung stehen. Damit ist es auf einfache Art möglich, sich sowohl auf Klassen(-Hierarchien) als auch auf den Instanzraum innerhalb einer Anfrage zu beziehen. Beispielsweise wird in [SM01, Seite 5] folgendes ausgeführt: *„... one may ask for questions like 'show me the concept taxonomy including only those concepts for which you have some news in the last week' ...“*. Diese Fähigkeit von F-Logic, also im Kern die Quantifizierung über Klassen, Attribute und Signaturelemente, macht sie aus syntaktischer Sicht zu

einer Sprache zweiter Stufe, die sich jedoch auf einfache Art und Weise auf die Logik erster Stufe abbilden läßt, was in Abschnitt 2.1.1.6 dargelegt wird. Im weiteren wird ein Überblick über die Sprache F-Logic gegeben.[1]

2.1.1.1 Basisbausteine

Es sei das folgende Alphabet vorausgesetzt, gegeben durch eine Menge von *Objektkonstruktoren* $\mathbf{F}$ und *Prädikatensymbolen* $\mathbf{P}$, einer unendlichen Menge von *Variablen* $\mathbf{V}$, den Symbolen

(i) :, ::, @, –>, –≫, =>, =≫, •->, •-≫, einschließlich den gängigen Klammer- und Interpunktionssymbolen,

(ii) und den logischen *Konnektoren* ***and***, ***or***, ***not***, ← und *Quantoren* ***forall*** und ***exists***.

Die Elemente der Menge $\mathbf{F}$ entsprechen den *Funktionssymbolen* aus der Prädikatenlogik, womit bezüglich der Termbildung alle in der Prädikatenlogik aufgestellten Definition übernommen werden können (siehe z.B. [EMC$^+$01, EFT86]). Objektkonstruktoren besitzen eine *Stelligkeit*, wobei 0-stellige Objektkonstruktoren auch *Konstanten* genannt werden. Die Menge der Terme über $\mathbf{F}$ und $\mathbf{V}$ wird mit $T_{\mathbf{F}}(\mathbf{V})$ bezeichnet, die variablenfreien mit $T_{\mathbf{F}}$. Letztere entspricht genau dem sogenannten *Herbrand Universum* [Llo87].

Elemente aus $T_{\mathbf{F}}(\mathbf{V})$ werden *id-Terme* genannt, Elemente aus $T_{\mathbf{F}}$ *Grund-id-Terme*. Grund-id-Terme stellen eindeutige Bezeichner für Objekte dar, worunter sowohl Klassen und deren Instanzen als auch einfache Datentypen mit deren Instanzen fallen. Gemäß der F-Logic-Terminologie stellt der Begriff Objekt innerhalb dieses Kapitels den Oberbegriff für Instanzen, Klassen und einfache Datentypen dar. Beispiele für id-Terme sind

99, *Lehrgang*, *lehrgang4563* resp. *zentral(berlin)* ,

welche die Zahl neunundneunzig, die Klasse für Lehrgänge, einen bestimmten Lehrgang resp. den zentralen Ort in Berlin bezeichnen könnten. Für zwei id-Terme a und b werden die Formen $a : b$ und $a :: b$ *is-a-assertions* genannt, wobei erstere die Instanzbeziehung und letztere die Spezialisierungsbeziehung symbolisieren soll. Beispiele hierfür sind

[1] Die speziellen nicht-prädikatenlogischen Begriffe werden in dieser Arbeit überwiegend direkt aus dem Englischen übernommen und nicht übersetzt. Lediglich für die wohlbekannten prädikatenlogischen Begriffe wird zumeist die deutsche Form gewählt.

lehrgang4563 : Lehrgang, *zentral(berlin) : Ort* und
Dozent :: Person .

2.1.1.2 Konstruktion von Objekten und Signaturen

Mit Hilfe von *is-a-assertions* wird einer der Grundbausteine von Ontologien, nämlich Taxonomien, gebildet. Neben diesen ausgezeichneten Elementen zur Beziehungssetzung von Objekten ist die Definition frei benennbarer Objektbeziehungen ein weiteres übliches Modellierungskonstrukt. Sie werden hier als *object molecules* bezeichnet. Ein *object molecule* hat die Form $o[e_1; \ldots; e_n]$, wobei o ein id-Term und e_i eine *method expression* ist ($n \geq 0$). *Method expressions* untergliedern sich in die weiter unten beschriebenen *signature expressions* sowie die *data expressions* mit den folgenden zwei Formen, wobei im weiteren innerhalb dieses Abschnitts $o, m, s_1, \ldots, s_k, s, t_1, \ldots, t_l$ id-Terme sind und $k, l \geq 0$:

- *scalar data expression:* $m@s_1, \ldots, s_k \rightarrow s$
- *set-valued data expression:* $m@s_1, \ldots, s_k \twoheadrightarrow \{t_1, \ldots, t_l\}$.

Für eine *data expression* d drückt die Form $o[d]$ aus, daß die Anwendung der *Methode* m auf das *host-Objekt* o mit den *Argumenten* $s_1, \ldots, s_k$ das *Resultat* s beziehungsweise $\{t_1, \ldots, t_l\}$ im Fall der *set-valued data expression* liefert. Ist $k = 0$, so nennt man m auch *Attribut* und das Symbol @ kann weggelassen werden.

```
klaus : Person. maria : ClubMitglied. maria : Dozent.
theater : Bereich. fußball : Bereich.
klaus [ alter -> 33; hobby ->> {theater, fußball}].
maria [ alter -> 27; hobby ->> {theater}].

e : ClubExkursion.
e [ treffpunkt -> zentral(Berlin); motto -> 'Historische Theater in Berlin';
    führung -> maria ].
```

Fragment 2.1: *Data Expressions* in F-Logic

Beispielsweise soll Fragment 2.1 den folgenden Sachverhalt darlegen: die dreiunddreißigjährige Person Klaus hat als Hobbys Theater und Fußball. Maria, die Mitglied eines Theaterclubs

und Dozentin ist, interessiert sich für Theater und führt eine Club-Exkursion unter dem Motto 'Historische Theater in Berlin'.

Das vorangegangene Beispiel legt lediglich fest, daß die Objekte *klaus* und *maria* Instanzen von *Person* sind, nicht aber, inwieweit das ihre Eigenschaften bestimmt. Es ist nicht angegeben, daß das Attribut *führung* zu *ClubExkursion* gehören und seine Zielklasse eine Person sein sollte, die sowohl Clubmitlied als auch Dozent ist. Wie üblich wird dies mit Hilfe von den Klassennamen zugeordneten Signaturen festgelegt, die in F-Logic *signature expressions* genannt werden und die folgenden zwei Spezialisierungen besitzen.

- *scalar signature expression:* $m@s_1,\ldots,s_k \Rightarrow (t_1,\ldots,t_l)$
- *set-valued signature expression:* $m@s_1,\ldots,s_k \Rightarrow\!\!> (t_1,\ldots,t_l)$

Für eine *signature expression* e bedeutet die Form $o[e]$, daß die Anwendung von m auf eine Instanz von o mit einer Argumentliste entsprechend den Klassen $s_1,\ldots,s_k$ ein Objekt als Resultat liefern muß, das Instanz jeder Klasse t_i ist.[2] Im Falle der *set-valued signature expression* muß letzteres für jedes Element der Resultatsmenge gelten. Ist $l = 1$, so können die Klammern weggelassen werden. Fragment 2.2 zeigt die entsprechenden Ausdrücke für das Fragment 2.1.

```
Clubmitglied :: Person. Dozent :: Person.
Person [ alter => Integer; hobby =>> Bereich ].

ClubExkursion [ treffpunkt => Ort; motto => String;
                führung => (ClubMitglied, Dozent) ].
```

Fragment 2.2: *Signature Expressions* in F-Logic

Signature expressions lassen sich als typ-bezogene Randbedingungen (*type constraints*) ansehen und zwar genau in dem Sinne, wie Java-Klassendefinitionen *type constraints* sind für Objektnetze oder XML Schemata für XML-Dokumente. Letzterer Vergleich trifft in besonderem Maße zu: denn die gesamte Informationsmenge besteht lediglich aus (wohlgeformten) XML-Dokumenten resp. *object molecules*. Einigen Elementen kommt die besondere Rolle zu, als *type constraints* (XML-Schema resp. *signature expression*) für andere zu dienen. Wie eingangs erwähnt und in [SM01] hervorgehoben, wird damit eine einheitliche Behandlung von

[2] Es sei bemerkt, daß die Angabe von i Ausdrücken mit jeweils t_i als Zielklasse semantisch äquivalent ist zu der die Klassenkonjunktion *(. . .)* verwendenden Form.

Klassen, Eigenschaften und Instanzen unterstützt, die in [KLW95] wie folgt beschrieben ist: „This, for instance, implies that their names [Anm.: Klassen, Attribute und Methoden] can be returned as query answers. In this way, schema information is turned into data and can be manipulated in the same language“.

2.1.1.3 Initialwerte

F-Logic stellt neben den Konstrukten zur Bildung von Objekten und Signaturen Konstrukte zur Angabe von Initialwerten zur Verfügung. Zu diesem Zwecke werden *data expressions* um die zwei Formen

- *inheritable scalar data expression:* $m@s_1,\ldots,s_k \bullet\!\!\rightarrow s$
- *inheritable set-valued data expression:* $m@s_1,\ldots,s_k \bullet\!\!\rightarrow\!\!> \{t_1,\ldots,t_l\}$

erweitert [3]. Mit ihrer Hilfe werden initiale Wertzuweisungen für Methodenaufrufe bereits auf Klassenebene vorgenommen, die bei der Instantiierung oder Spezialisierung überschrieben werden können. Dieses Konstrukt bewirkt beispielsweise, daß die Menge von *object molecules* $M = \{a[b \bullet\!\!\rightarrow c], d:a, e::a\}$ implizit erweitert ist um $\{d[b \rightarrow c], e[b \bullet\!\!\rightarrow c]\}$. Dies wäre aber nicht der Fall, wenn $d[b \rightarrow p]$ und $e[b \bullet\!\!\rightarrow q]$ in M enthalten, also d resp. e überschrieben worden wären. Analoges gilt für die *inheritable set-valued data expression*. Unser Beispiel aus Fragment 2.2 könnte nun um *Person [alter •-> 33]* erweitert werden. Jede Instanz von *Person* hätte dann das initiale Alter von dreiunddreißig und jede Spezialisierung von *Person* würde diesen Initialwert übernehmen, falls keine Überschreibung wie im Fall der Instanz *klaus* stattfindet.

Is-a-assertions und *object molecules* werden als *F-molecule* bezeichnet. Bild 2.1 zeigt die Hierarchie der bisher eingeführten Konstrukte. Während also mit *is-a-assertions* Instanz-und Generalisierungs-/Spezialisierungsbeziehungen angegeben werden, dienen *non-inheritable data expressions* zur Spezifikation von Objekten über Attributwerte und Methodenauswertungen und *inheritable data expressions* zur Definition von überschreibbaren Initialwerten. *Signature expressions* geben die Randbedingungen für die Objektbildung vor. Dabei gibt es jeweils eine Variante für skalare Objekte und eine für Mengen.

[3] Dementsprechend werden die auf Seite 10 eingeführten *data expressions* auch als *non-inheritable* bezeichnet.

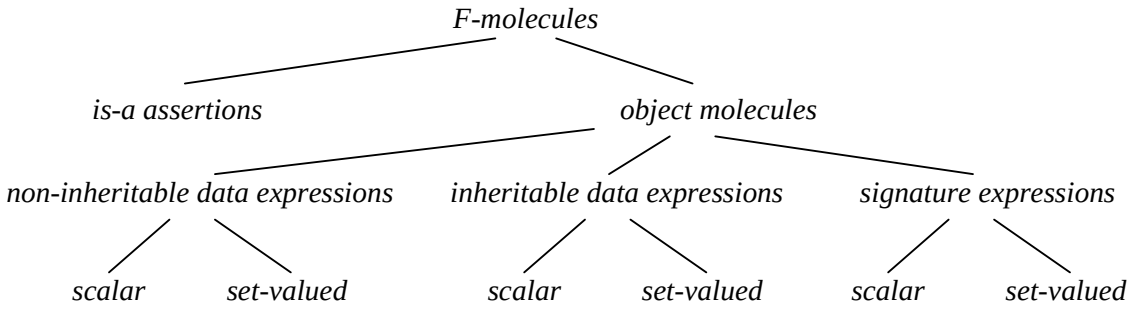

Bild 2.1: Hierarchie der *F-molecules*

2.1.1.4 Prädikatenlogische Formeln, abgeleitete Objektbeziehungen und Signaturen

Neben den Konstrukten zur Bildung von Objekten und deren Beziehungen gestattet F-Logic die Bildung von atomaren Formeln. Eine über $T_{\mathbf{F}}(\mathbf{V})$ und $\mathbf{P}$ bildbare atomare Formel wird auch als *P-molecule* bezeichnet. Wie üblich wird die Erweiterung zu einer prädikatenlogischen Formel, hier *F-formula* genannt, folgendermaßen definiert:[4]

- *F-molecules* und *P-molecules* sind *F-formulas*.
- Sind G und H *F-formulas*, so auch $(G \leftarrow H)$, $(G$ ***and*** $H)$, $(G$ ***or*** $H)$ und $($***not*** $G)$.
- Ist $v \in \mathbf{V}$ und G eine *F-formula*, so auch $($***exists*** $v\, G)$ und $($***forall*** $v\, G)$.

Damit ist man nun in der Lage, *Regeln* zur Deduktion neuer Zusammenhänge aus bestehenden Objektbeziehungen abzuleiten. Beispielsweise könnte die Interessengemeinschaft zweier Personen bezüglich eines Bereichs mit den Formeln

forall *b, p1, p2, p3* *gleichInteressiert(b, p1, p3)* $\leftarrow$
gleichInteressiert(b, p1, p2) ***and*** *gleichInteressiert(b, p2, p3).*
forall *b, p1, p2* *gleichInteressiert(b, p1, p2)* $\leftarrow$ *gleichInteressiert(b, p2, p1).*

[4] Es gelten die gängigen Regeln zur Klammerung und Priorität. Die (redundante) Aufnahme des Implikationssymbols $\leftarrow$ trägt der Tatsache Rechnung, daß in der Anwendung der Fokus meist auf *Regelmengen*, d.h. Mengen von Implikationen, liegt.

```
forall b, p1, p2  gleichInteressiert(b, p1, p2) ←
                    p1 [ hobby ↠ {b} ]  and  p2 [ hobby ↠ {b} ].
```

ausgedrückt werden, die besagen, daß die Gleich-Interessiertheit eine transitive und symmetrische Relation ist und sich aus den gemeinsamen Hobbys ergibt. Damit wäre *gleichInteressiert(theater, klaus, maria)* aus Fragment 2.1 ableitbar. Eine gemischte Verwendung von Prädikaten und *data expressions* zur Ableitung von neuen *data expressions* erweist sich als sinnvoll für die Definition von abgeleiteten Attributen: die Angabe von

```
    hans [ hobby ↠ {theater} ].
(i) forall b   Person [ gleichInteressierte(b) ⇒ Person ]  ←  b : Bereich.
    forall b, p1, p2   p1 [ gleichInteressierte(b) ↠ p2 ]  ←  gleichInteressiert(b, p1, p2).
```

zusammen mit Fragment 2.1 und der Definition des Prädikats *gleichInteressiert* führt nämlich zu der abgeleiteten Attributzuweisung

```
klaus [ gleichInteressierte(theater) ↠ {maria, hans} ].
```

Dieses Beispiel zeigt zugleich die Möglichkeit, zusammengesetzte Attribute zu definieren. Denn Regel (i) definiert ein zusammengesetztes Attribut *gleichInteressierte(b)* für jeden Bereich *b*. Es sei bemerkt, daß *x[y↠{u,v}]* äquivalent ist zu *x[y↠{u}] and x[y↠{v}]* , daß also ↠ zu lesen ist als ‚hat als Element(e)‘. [5]

Aufgrund der erwähnten Behandlung von Klassen und Signaturen als eigenständige Objekte und deren Verwendung in Formeln ist man in der Lage, *type constraints* mit Hilfe von Formelmengen zu definieren, das heißt eine vollständig dynamische und damit flexible Typisierung zu erreichen. Als Beispiel hierfür läßt sich eine Veränderung des Attributs *gleichInteressierte* aufführen. Bisher ist als Zielklasse lediglich *Person* angegeben. Sinnvoll wäre aber die Einschränkung auf Personen mit gleicher Interessenlage: es soll ausgedrückt werden, daß diese Beziehung nur mit Personen eingegangen werden soll, die an diesem Bereich auch interessiert sind. Die Formulierung in F-Logic könnte wie folgt lauten.

```
forall p, b   p : Interessierte(b)  ←  p [ hobby ↠ {b} ].
forall b      Interessierte(b) [ gleichInteressierte(b) ⇒ Interessierte(b) ]  ←  b : Bereich.
```

[5] Außerdem sind Schachtelungen zugelassen, das heißt die Form *x[p ⇝ y[q ⇝ z]]* steht als Abkürzung für *x[p ⇝ y] and y[q ⇝ z]*, wobei ⇝ für eine des bisher eingeführten Beziehungsarten steht.

Die erste Formel definiert für jedes *b* die Klasse *Interessierte(b)*, deren Instanzen diejenigen Personen sind, die das gemeinsame Interesse *b* haben. Die zweite Formel definiert eine durch diese Klassen induzierte Familie von Signaturen.[6] Damit wird erreicht, daß

klaus [gleichInteressierte(fußball) ↠ {maria}]

nicht signaturkonform ist, da zwar *klaus : Interessierte(fußball)*, aber nicht *maria : Interessierte(fußball)* gilt.

F-Logic verfügt hiermit über das Ausdrucksmittel, Ziel- und Quellklassen innerhalb von Signaturen dynamisch über die Angabe von Regelmengen zu bestimmen und dient damit als Vorbild für die dynamische Typisierung von Attributen als Teil der in dieser Arbeit vorgestellten Beschreibungssprache (vergl. Anforderung aus Abschnitt 3.1.6).

2.1.1.5 F-Logic-Programme

Analog zur logischen Programmierung [Llo87] bezeichnet man hier eine Menge von *F-formulas* als F-Logic-Programm. Zur Spezifikation einer Domäne für eine Anwendung ist es jedoch angebracht, diese Menge zu untergliedern. Die in [KLW95] vorgeschlagene Untergliederung (siehe Fragment 2.3), für die eine Einschränkung auf Regeln voraussetzt ist, deren Konklusion nur aus einem *P*- oder *F-molecule* bestehen darf, postuliert drei Untermengen:

a) Die *is-a-Hierarchie*, die nur aus Regeln besteht, deren Konklusion eine *is-a-assertion* ist.

b) Den *Signaturdeklarationsteil*, der nur aus Regeln besteht, deren Konklusion eine *signature expression* ist.

c) Die *Objektbasis*, deren Regelkonklusionen *P-molecules* oder *data expressions* sind.

Man beachte, daß der Instanzraum entgegen der T/A-Box-Aufteilung in a) enthalten ist, weshalb die Bezeichnung Objektbasis für c) mißverständlich ist. Eine ähnliche und bezüglich c) sinnvollere Dreiteilung ist in [Dec02, Kap. 7] zu finden. Dort werden die Teile a)-c) *Class*

[6] Dieses Beispiel zeigt zugleich die Möglichkeit, Typkonstruktoren zu definieren. Sonst gängige Typkonstruktoren sind List(), Stack(), etc. .

```
// is-a-Hierarchie:
Clubmitglied :: Person.
Dozent :: Person.                          theater : Bereich.
klaus : Person.                            fußball : Bereich.
maria : ClubMitglied. maria : Dozent.      exkurs : ClubExkursion.
hans : Person.

forall p, b   p : Interessierte(b) ← p [ hobby →≫ {b} ].

// Signaturdeklarationsteil:
Person           [ alter ⇒ Integer; hobby ⇒≫ Bereich].
ClubExkursion [ treffpunkt ⇒ Ort; motto ⇒ String; führung ⇒ (ClubMitglied, Dozent) ].

forall b   Interessierte(b) [ gleichInteressierte(b) ⇒≫ Interessierte(b) ]  ← b : Bereich.

//Objektbasis:
klaus   [ alter → 33; hobby →≫ {theater, fußball}].
maria   [ alter → 27; hobby →≫ {theater}].
hans    [ alter → 30; hobby →≫ {fußball, theater}].
exkurs [ treffpunkt → zentral(Berlin); motto → 'Historische Theater in Berlin'; führung → maria ].

forall b, p1, p2, p3   gleichInteressiert(b, p1, p3) ←
                          gleichInteressiert(b, p1, p2)  and  gleichInteressiert(b, p2, p3).
forall b, p   gleichInteressiert(b, p, p).
forall b, p1, p2   gleichInteressiert(b, p1, p2) ← gleichInteressiert(b, p2, p1).

forall b, p1, p2   gleichInteressiert(b, p1, p2) ←
                     p1 [ hobby →≫ {b} ]  and  p2 [ hobby →≫ {b} ].
```

Fragment 2.3: Dreiteilung eines F-Logic-Programms

Hierarchy, *Attribute Declarations* resp. *Axioms Declarations* genannt, wobei der Instanzraum nicht enthalten ist.

2.1.1.6 Semantik und Wohlgetyptheit

Für F-Logic ist eine modelltheoretische Semantik definiert. Sie stellt im wesentlichen eine Erweiterung der klassischen Semantik der FOL dar. Wie üblich wird die Semantik als eine Interpretation I definiert (genannt *F-Struktur*), bestehend aus einer Trägermenge D und den Abbildungen $I_{\mathbf{F}}$ resp. $I_{\mathbf{P}}$, welche den Objektkonstruktoren aus $\mathbf{F}$ und den Prädikatensymbolen

aus $\mathbf{P}$ Funktionen resp. Relationen über D entsprechend den Stelligkeiten zuordnen, wobei $I_{\mathbf{F}}$ wie üblich auf id-Terme fortgesetzt wird.

Gemäß einem der Ziele von Kifer, Lausen und Wu, nämlich eine prädikatenlogische Fundierung der objektorientierten Programmierung, erfolgt eine besondere Behandlung der *is-a-assertions* und der Methodenausdrücke (*method expressions*, siehe Abschnitt 2.1.1.2). Das Instantiierungskonstrukt $:$ und das Spezialisierungskonstrukt $::$ hat als Gegenstück auf semantischer Seite die speziellen Relationen $\in_D$ resp. $\preceq_D$ über D. Eine F-Struktur $I = (D, \preceq_D, \in_D, I_{\mathbf{F}}, I_{\mathbf{P}})$ hat lediglich die folgenden Bedingungen zu erfüllen:

(1) (i) $\preceq_D$ ist eine Halbordnung.[7]

(ii) $a \in_D b$, falls $a \in_D c$ und $c \preceq_D b$.

[KLW95] verzichtet auf eine weitergehende Einschränkung dieser Relationen, um für unterschiedliche Anwendungen offen zu sein. Der wesentliche Unterschied zu der üblichen (eingeschränkteren) direkten Semantik ist der Verzicht auf die Abbildung des Instantiierungs- und Spezialisierungskonstrukts auf die Element- resp. Teilmengenbeziehung. Dies ermöglicht – unter Umgehung von Paradoxien – die Zyklizität von $\in_D$. Diese Indirektion spielt bei der Wissensrepräsentation [HM01] und insbesondere beim *Resource Description Framework* eine wichtige Rolle. Eine Erläuterung dieser Problematik wird im Rahmen der RDF-Semantik in Abschnitt 2.2.1.8 gegeben.

Die zweite Besonderheit der F-Logic-Semantik ist die zusätzliche Interpretation von Methodenausdrücken. Prinzipiell kann jeder id-Term als Methode verwendet werden. Daher werden zusätzlich auf semantischer Seite jedem Element aus D, auf die id-Terme mittels $I_{\mathbf{F}}$ abgebildet werden, entsprechende Funktionen über D zugeordnet. Eine detaillierte Charakterisierung ist in [KLW95, Kapitel 5] zu finden.

Eine Alternative zur direkten modelltheoretischen Semantik ist die Beschreibung der Semantik in einer formalen logischen Sprache (hier FOL), was im Rahmen dieser Arbeit als *axiomatische Semantik* bezeichnet wird.[8] Die axiomatische Semantik eines F-Logic-Programs P umfaßt zwei Teile: zum einen eine (hier mit μ bezeichnete) Transformation, welche P auf

[7] In der Originaldefinition wird zunächst die strenge Halbordnung $\prec_D$ für eine Interpretation eingeführt. Bei der späteren Zuordnung zur Spezialisierungsbeziehung wird die Irreflexivität der Intuition entsprechend aufgehoben.

[8] Der Begriff *axiomatische Semantik* wird hier nicht, wie es sonst auch vorkommt, mit dem Hoare-Kalkül verbunden.

prädikatenlogische Formeln abbildet. Zum anderen müssen die gemäß der Modelltheorie geltenden Eigenschaften von Interpretationen durch entsprechende Formeln axiomatisiert werden, sei es direkt oder indirekt über F-Logic-Formeln (mit anschließender Anwendung von μ). Die in [SS04, Kapitel 2] und [Dec02, Kapitel 7] tabellarisch informell verdeutlichte Transformation, dort allerdings nach Hornlogik, kann wie folgt präzisiert werden, wobei wir uns bei den *object molecules* exemplarisch auf *non-inheritable scalar expressions* (siehe Bild 2.1) beschränken:[9]

(2)
- (i) Für id-Terme ist μ die eineindeutige Zuordnung zu prädikatenlogischen Termen.
- (ii) $(x : y)^\mu = instanceOf(x^\mu, y^\mu)$
- (iii) $(x :: y)^\mu = subClassOf(x^\mu, y^\mu)$
- (iv) $o\,[m@s_1, \ldots, s_k \rightarrow s]^\mu = methodAppl(m^\mu, arg(o^\mu, {s_1}^\mu, \ldots, {s_k}^\mu), s^\mu)$
- (v) $o\,[m@s_1, \ldots, s_k \Rightarrow s]^\mu = methodSig(m^\mu, arg(o^\mu, {s_1}^\mu, \ldots, {s_k}^\mu), s^\mu)$
- (vi) Für (die verbleibenden) *F-formulas* ist μ die Fortsetzung auf die Argumente. [10]

Der zweite Teil der axiomatischen Semantik eines F-Logic-Programms P besteht aus den *F-formulas*, welche die in der Modelltheorie geltenden Eigenschaften axiomatisieren. Neben den wie üblich aufgestellten Formeln zur Erfüllung von (1.i)[11] beziehen sich die folgenden auf die Instanzbeziehung und Methodensignaturen.[12]

(3)
- (i) ***forall*** *x,y,z* $x : z \leftarrow x : y$ ***and*** $y :: z$ (*Subklassen-Inklusion*)
- (ii) ***forall*** *o, o', m,* $s_1, \ldots, s_k$, *s*
  $o'\,[\,m\,@\,s_1, \ldots, s_k \Rightarrow s\,] \leftarrow o' :: o$ ***and*** $o\,[\,m\,@\,s_1, \ldots, s_k \Rightarrow s\,]$
  (*structural inheritance*)

[9] Der besseren Lesbarkeit wegen wird $\mu(x)$ mit x^μ notiert. Auf eine gesonderte Betrachtung xyxy yxx yxyx yxyx yxyx yxyx y xy xyxyx yxyyxy von Zielklassen-Konjunktionen kann bei *signature expressions* verzichtet werden (siehe Fußnote 2 auf Seite 11). Man beachte, daß die mit der Anwendung von μ eingeführten Prädikate (z.B. $subClassOf$) spezielle nicht in P vorkommende Prädikate sind.

[10] Damit sind alle mit Hilfe der logischen Konnektoren und Quantoren gebildeten Formeln gemeint. Wie üblich werden die F-Logic-Konnektoren und Quantoren auf $\wedge, \ldots, \exists$ abgebildet.

[11] Damit sind die Axiome zur Reflexivität, Azyklizität und Transitivität von $::$ gemeint.

[12] Es sei bemerkt, daß im Rahmen der Arbeit nur die *structural inheritance* betrachtet wird, welche leicht mit (monotoner) FOL abbildbar ist.

(iii) ***forall*** *o, m, s_1,. . ., s_i, s'_i,. . ., s_k, s*
o [m @ s_1,. . ., s'_i,. . ., s_k ⇒ s] ← s'_i :: s_i ***and*** *o [m @ s_1,. . ., s_i,. . ., s_k ⇒ s]*
(*input restriction*)

(iv) ***forall*** *o, m, s_1,. . ., s_k, s, s'*
o [m @ s_1,. . ., s_k ⇒ s'] ← s :: s' ***and*** *o [m @ s_1,. . ., s_k ⇒ s]*
(*output relaxation*)

Die erste Regel axiomatisiert Eigenschaft (1.ii). Die zweite entspricht der üblichen Vererbung von Attributen und Methoden. Die letzten beiden spiegeln die Tatsache wieder, daß es für jede Methodensignatur bezüglich des Definitionsbereichs entsprechende Spezialisierungen und bezüglich des Wertebereichs entsprechende Generalisierungen gibt, induziert durch die Methodenstelligkeiten in P. Das heißt, daß die letzten drei jeweils eine Familie von Formeln repräsentieren: für jede in einem F-Logic-Programm P vorkommende Methodenstelligkeit k gibt es jeweils eine Formel. In der dritten Regel gibt es für jedes k zusätzlich für jede Argumentposition eine Ausprägung. Die bezüglich der Stelligkeiten aus P aufgelöste Regelmenge (3) sei mit AX_P bezeichnet.

Data expressions müssen gemäß den *signature expressions* gebildet werden, was man auch *Wohlgetyptheit* nennt. Der Begriff der Wohlgetyptheit in F-Logic ist als Eigenschaft der Modelle auf semantischer Seite definiert. Er soll als Vorlage für Algorithmen zur Typprüfung dienen. Details dazu und zur Beweistheorie (Unifikation, Inferenzregeln), welche eine Erweiterung des Prädikatenkalküls um die F-Logic-spezifischen Konstrukte darstellt, sind in [KLW95, Kap. 5 u. 11] zu finden. Dementsprechend läßt sich die Überprüfung, welche Ausdrücke nicht gemäß den Signaturen gebildet sind, innerhalb der axiomatischen Semantik wie folgt formulieren, wobei $\vdash_\mu$ die klassische Ableitung unter Einbeziehung von AX_P und Anwendung von μ bezeichne.[13] Gelte $P \vdash_\mu d$ für eine *data expression* $d = o[m@s_1, \ldots, s_k \rightarrow s]$. d ist *wohlgetypt*, falls folgende Bedingung gilt:

(4) Wenn $P \vdash_\mu c[m@c_1, \ldots, c_k \Rightarrow c]$ und
$P \vdash_\mu \{\, o : c, s_1 : c_1, \ldots, s_k : c_k \,\}$
dann $P \vdash_\mu s : c$.

Daraus folgt insbesondere, daß d auch dann wohlgetypt ist, wenn gar keine entsprechende Signatur vorhanden ist, d.h. die Prämisse aus (4) nie erfüllt wird. Die originale Definition der Wohlgetyptheit in F-Logic erzwingt jedoch, daß zu jeder *data expression* mindestens eine *signature expression* existieren muß, für die Bedingung (4) mit gültiger Prämisse gilt.

[13] Genauer: $P \vdash_\mu F \;:\Leftrightarrow\; P^\mu \cup (AX_P)^\mu \vdash F^\mu$.

Programm P

```
1   ClubExkursion :: Exkursion.
2   exkurs88 : ClubExkursion.
3   maria : Person.

4   Exkursion [ führung => Person ].

5   exkurs88 [ führung -> maria].
```

Fragment 2.4

Das einfache Beispiel aus Fragment 2.4 soll die Wohlgetyptheit verdeutlichen. Neben den dort aufgeführten Typisierungen *maria : Person* und *exkurs88 : ClubExkursion* ist aufgrund von (3.i) die Typisierung *exkurs88 : Exkursion* aus den Zeilen *1* und *2* ableitbar. Zeile *5* ist nun wohlgetypt, da wegen den eben genannten Typisierungen die Bedingung (4) sowohl für

Exkursion [führung $\Rightarrow$ *Person]*

als auch für

ClubExkursion [führung $\Rightarrow$ *Person]* (abgeleitet mit Hilfe von (3.ii))

gilt.

2.1.2 Description Logic

Die frühen netzbasierten Modellierungssprachen verfügten über keine präzise semantische Charakterisierung. Die ersten Charakterisierungen dieser Sprachen als Fragmente der FOL war der Beginn des Bereichs *terminological systems* / *concept languages*, zu dessen Ergebnissen die heutige Sprachfamilie *Description Logic* (DL) gehört und deren umfangreiche Behandlung in [BCM$^+$03] zu finden ist. DL, im deutschsprachigen Raum auch *Beschreibungslogik* oder *Termsubsumptionssprache* genannt [GRS00, Abschnitt 5.4.1], lässt sich durch die folgenden drei Punkte charakterisieren.

- Ausgehend von Klassennamen (*atomic concepts*), Rollennamen (*atomic ro-*

les) und Individuennamen (*individuals*) steht eine relativ kleine Menge von Sprachkonstrukten zur Bildung von komplexen Klassen- und Rollenbeschreibungen (*concept/role descriptions*, *terminological axioms*) sowie zur Bildung der Instanz- resp. Rollenbeziehung zwischen Klassen und Individuen resp. zwischen Individuen zur Verfügung (*assertions*).

- DL erzwingt eine strikte Trennung von Klassen-/Rollenbeschreibungen und dem Instanzraum. Erstere bilden die sogenannte *terminological box* (TBox) während der Instanzraum mit *assertional box* (ABox) bezeichnet wird.
- DL beschränkt sich auf spezielle Inferenzprozeduren, die das Abfragen der Subsumptionsbeziehung (Subklassenbeziehung) innerhalb der TBox und das Abfragen der Instanzbeziehung innerhalb der ABox ermöglichen.

Im weiteren wird ein Überblick über die Sprachfamilie DL auf der Grundlage von [BCM+03, Appendix 1] gegeben. Dafür sei das folgende Alphabet vorausgesetzt: eine Menge von *atomic concepts* $\mathbf{C}$, welche die Elemente ***top*** und ***bottom*** enthält, eine Menge von *atomic roles* $\mathbf{R}$ und eine Menge von *individuals* $\mathbf{O}$.[14]

2.1.2.1 *Concept/role descriptions*

Mit Hilfe einer Menge von *concept/role constructors*, welche die gleiche Rolle wie die Termfunktoren in der FOL spielen, können nun die sogenannten *concept/role descriptions* (auch *concept/role terms* genannt) gebildet werden. Der Umfang der Konstruktormenge bestimmt die DL-Sprachklasse.

Die hier vorgestellte DL-Sprachklasse wird mit Hilfe der Konstruktormenge für Klassen $\mathbf{FC} = \{$***and***, ***or***, ***not***, ***all***, ***some***, ***at-least***, ***at-most***$\}$ und der für Rollen $\mathbf{FR} = \{$***inverse***, ***transitive***$\}$ gebildet.[15]

Die Menge der *concept/role terms* wird hier mit $DL_{\mathbf{FC}}$ resp. $DL_{\mathbf{FR}}$ bezeichnet und ist induktiv definiert durch

(1) $\mathbf{C} \subseteq DL_{\mathbf{FC}}$ und $\mathbf{R} \subseteq DL_{\mathbf{FR}}$.

[14] Beziehungsnamen werden in der DL-Terminologie Rollennamen genannt. Im folgenden wird bei *concepts* auch von Klassen gesprochen.

[15] Diese Klasse wird in der DL-Terminologie mit dem Kürzel $\mathcal{ALCN}^{-1}_{R^+}$ bezeichnet. Zur Bildung der Kürzel und zu Sprachequivalenzen siehe [BCM+03, Abschnitt 2].

(2) Sei $r \in DL_{\mathbf{FR}}$. Dann sind

(i) ***inverse*** (r) (*inverse roles*)

(ii) ***transitive*** (r) (*transitive roles*)

enthalten in $DL_{\mathbf{FR}}$.

(3) Seien $c, d \in DL_{\mathbf{FC}}$, $r \in DL_{\mathbf{FR}}$ und n eine nicht-negative ganze Zahl. Dann sind

(i) ***and*** (c, d) (*intersection*)

(ii) ***or*** (c, d) (*union*)

(iii) ***not*** (c) (*negation*)

(iv) ***all*** (r, c) (*value restriction*)

(v) ***some*** (r, c) (*existential quantification*)

(vi) ***at-least*** (n, r) (*unqualified number restriction*)

(vii) ***at-most*** (n, r) (*unqualified number restriction*)

enthalten in $DL_{\mathbf{FC}}$.

Im folgenden bezeichne, falls nicht anders angegeben, c, d einen *concept term*, r, s einen *role term*, o ein *individual* und n eine nicht-negative ganze Zahl.

Ausgehend von (1) und den Konstruktoren (i) bis (iii) aus (3) werden Klassenausdrücke gebildet.[16] Dabei sollen z.B. die Ausdrücke

and *(ClubMitglied, Dozent)* und ***and*** *(**or** (Informatiker, Elektrotechniker),* ***not*** *(Berliner))*

für die Individuen stehen, die gleichzeitig Mitglieder eines Clubs und Dozenten sind resp. für diejenigen, die Informatiker oder Elektrotechniker, aber keine Berliner sind. Grundlage für Beziehungsdefinitionen sind die Konstrukte (3.iv) bis (3.vii). Eine Besonderheit im Vergleich zur semantischen Datenmodellierung beziehungsweise zur Modellierung mit F-Logic ist es, daß zunächst Beziehungsdefinitionen nicht an Klassen gebunden werden müssen. Zwei Beispiele für die *value restriction* und *existential quantification* sind die Ausdrücke

[16] Wie üblich ist dabei im folgenden für einen Konstruktor $k \in \{$***and***, ***or***$\}$ der Ausdruck $k(x_1, \ldots, x_n)$ eine Kurzform für $k(x_1, k(x_2, \cdots k(x_{n-1}, x_n) \cdots))$.

***all** (führung, Dozent)* und ***some** (hobby, Sportart)* .

Ersterer soll für alle Dinge stehen, dessen Führung ein Dozent übernehmen muß. Genauer: wenn ein Ding eine Führung besitzt, dann muß die Führung von einem Dozenten übernommen werden. Eine Einschränkung, um welche Art Ding es sich handelt, muß an dieser Stelle noch nicht angegeben werden. Zunächst können alle Individuen aus $\mathbf{O}$ in Frage kommen. Der zweite Ausdruck steht für alle Individuen, die als Hobby eine Sportart haben, genauer: alle Individuen, die ein Hobby haben, wobei dieses Hobby eine Sportart sein muß.

Die Konstrukte (3.vi) und (3.vii) dienen der Festlegung der Kardinalitäten von Rollen. Beispielsweise steht der Ausdruck

***and** (**at-least** (1, hobby), **at-most** (2, hobby))*

für diejenigen, die Hobbys haben müssen, aber nicht mehr als zwei. Die in (2) definierten Konstruktoren bieten die Möglichkeit, Rollen auf bestehenden aufzubauen, die z.B. die folgenden Ausdrücke erlauben:

***some** (**transitive** (kind), Mann)* und ***some** (**transitive** (**inverse** (kind)), Dozent)*

Mit dem ersten Ausdruck sind diejenigen gemeint, die einen männlichen Nachfahren haben, während der zweite diejenigen meint, die einen Dozenten als Vorfahren haben.

2.1.2.2 *Terminological axioms*

Die bisher eingeführten Konstrukte bilden die Grundbausteine zum Aufbau von Klassendefinitionen, welche als Formeln zweier spezieller Prädikate verstanden werden können und im Kontext der DL als *terminological axioms* bezeichnet werden:

$x \equiv y$ und	(*concept/role equations*)
$x \sqsubseteq y$	(*concept/role inclusions*),

wobei x, y entweder für zwei Rollen oder zwei Klassen stehen. Die sogenannte *terminological box* (TBox) ist die konjunktive Verknüpfung genau dieser Formeln. Sind c oder r atomar, so spricht man im Falle der Gleichungen von *concept/role definitions* und im Fall von Inklusio-

nen von *primitive concept/role introductions*.[17] Mit der ersteren Form werden üblicherweise Klassen- und Eigenschaftsdefinitionen vorgenommen wie z.B.

elternteil ≡ ***inverse*** *(kind)* und
Person ≡ ***and*** (***all*** *(hobby, Bereich),* ***all*** *(kind, Person))* .

Der erste Ausdruck definiert *elternteil* als inverse Relation von *kind*. Der zweite Ausdruck ist die Definition der Klasse *Person* mit den Attributen *hobby* (Zielklasse *Bereich*) und *kind* (Zielklasse *Person*). Genauer: ein Individuum ist genau dann eine Person, wenn es als Kinder wieder Personen und als Hobby einen Bereich hat. Die letzte Umschreibung macht eines der Hauptziele von DL deutlich: es geht nicht nur um den Entwurf von Klassen allein mit dem Ziel, später Instanzen davon zu bilden und den entsprechenden Attribute Werte zuzuweisen, sondern auch darum, die Klasse von Individuen nachträglich zu bestimmen. Nämlich auf Grund der Äquivalenz sind alle Individuen, deren Kinder Personen und deren Hobbys Bereiche sind, auch Personen.

Concept/role inclusions dienen gemeinhin der Bestimmung der Subklassenbeziehung oder der Subsumptionsbeziehung zwischen Relationen (z.B. *Mann* ⊑ *Person* oder *nutztSchnittstelleVon* ⊑ *istAbhängigVon*).

2.1.2.3 *Assertions*

Aufbauend auf den Klassen- und Rollendefinitionen der TBox können nun Instantiierungen innerhalb der *assertional box* (ABox) angegeben werden. Sie unterteilen sich in Instantiierungen der Klassen und Instantiierungen der Rollen. Mit ersteren werden Individuen aus **O** einem *concept term* zugeordnet und mit letzterem einem *role term*: die Formeln

$o : c$ und	(*concept assertions*)
$(o_1, o_2) : r$	(*role assertions*)
o_1 = o_2	(*equality*)
o_1 != o_2	(*inequality*)

werden demgemäß *assertions* genannt.[18]

[17] Man beachte, daß, wenn allgemeine Inklusionen erlaubt sind, $x \equiv y$ ersetzbar ist durch die Konjunktion von $x \sqsubseteq y$ und $y \sqsubseteq x$.

[18] Im folgenden steht der Ausdruck $(x_1, \ldots, x_n) : y$ als Abkürzung für $x_1 : y, \ldots, x_n : y$.

TBox T :

```
1    Person ≡ and (
2                some (alter, Integer), at-most (1, alter),
3                all (hobby, Bereich),
4                all (kind, Person)).

5    Dozent ⊑ Person.
6    Bereich ⊑ top.
7    and (Bereich, Person) ≡ bottom.
```

ABox A :

```
8    klaus : Person.
9    maria : Dozent.
10   (fußball, theater) : Bereich.
11   ((klaus, 33), (maria, 27)) : alter.
12   ((klaus, fußball), (klaus, theater), (maria, theater)) : hobby.
```

Fragment 2.5: T/A-Box

Im Gegensatz zu der FOL-orientierten F-Logic ist man hier auf variablenfreie atomare Formeln eingeschränkt, die mit den Prädikaten $\sqsubseteq$, $\equiv$, *:*, *=* und *!=* gebildet werden können. Damit ist auch die Angabe von Regeln, wie sie in Fragment 2.3 auf Seite 16 unter anderem zur dynamischen Typisierung genutzt werden, nicht möglich. Fragment 2.5 zeigt ein Beispiel für die T/A-Box-Aufteilung, welches Teile des Fragments 2.3 übernimmt.

2.1.2.4 Semantik

Wie in den Erklärungen der vorangegangenen Beispielen widergespiegelt, wird die formale Semantik von DL üblicherweise über Interpretationen in einer Struktur definiert. Eine Interpretation $I = (D, \alpha)$ besteht aus einer nichtleeren Menge D, genannt *Domain*, und einer Interpretationsfunktion α, welche Elemente aus $\mathbf{C}$ auf Teilmengen von D, Elemente aus $\mathbf{R}$ auf binäre Relationen über D und Elemente aus $\mathbf{O}$ auf Elemente aus D abbildet. Die ausgezeichneten Konstanten und die Konstruktoren sind wie folgt induktiv definiert:

$$
\begin{aligned}
\alpha(\textbf{\textit{top}}) &= D \\
\alpha(\textbf{\textit{bottom}}) &= \emptyset \\
\alpha(\textbf{\textit{and}}\ (c, d)) &= \alpha(c) \cap \alpha(d) \\
\alpha(\textbf{\textit{or}}\ (c, d)) &= \alpha(c) \cup \alpha(d)
\end{aligned}
$$

$$
\begin{array}{lcl}
\alpha(\textbf{\textit{not}}\ (c)) & = & D \setminus \alpha(c) \\
\alpha(\textbf{\textit{all}}\ (r,c)) & = & \{x \in D \mid \text{falls } (x,y) \in \alpha(r) \text{ für } y \in D, \text{ so ist } y \in \alpha(c)\} \\
\alpha(\textbf{\textit{some}}\ (r,c)) & = & \{x \in D \mid \text{es existiert ein } y \in \alpha(c), \text{ so daß } (x,y) \in \alpha(r)\} \\
\alpha(\textbf{\textit{at-least}}\ (n,r)) & = & \{x \in D \mid\ |\hat{r}(x)| \geq n\}\ , \\
\alpha(\textbf{\textit{at-most}}\ (n,r)) & = & \{x \in D \mid\ |\hat{r}(x)| \leq n\}\ , \\
 & & \quad \text{wobei } \hat{r}(x) := \{y \in D \mid (x,y) \in \alpha(r)\} \\
\alpha(\textbf{\textit{inverse}}\ (r)) & = & \{(x,y) \in D \times D \mid (y,x) \in \alpha(r)\} \\
\alpha(\textbf{\textit{transitive}}\ (r)) & = & \bigcup_{n \geq 1} (\alpha(r))^n \\
 & & \quad \text{wobei } R^n \text{ die iterierte Komposition binärer Relationen ist.}
\end{array}
$$

Weiterhin ordnet eine Interpretation die Prädikate $\sqsubseteq$ und $:$ der Teilmengenbeziehung resp. Elementbeziehung zu (zur Redundanz von $\equiv$ siehe Fußnote 17). Die Definitionen der Gültigkeit und Erfüllbarkeit von Formeln geschieht analog zu denen der FOL (siehe auch [EMC+01, EFT86, Kapitel 19 resp. 3]). Demgemäß sind die Formeln $c \sqsubseteq d$, $o : c$ und $(o_1, o_2) : r$ *gültig* in einer Interpretation $I = (D, \alpha)$, falls $\alpha(c) \subseteq \alpha(d)$,[19] $\alpha(o) \in \alpha(c)$ resp. $(\alpha(o_1), \alpha(o_2)) \in \alpha(r)$. Man sagt, daß I ein *Modell* einer Formelmenge F ist (oder auch I *erfüllt* F), falls alle Formeln aus F gültig sind in I. Man schreibt dann $I \models F$. F ist *erfüllbar*, falls mindestens ein Modell für F existiert. Eine Formel(-Menge) F' *folgt* aus F, falls jedes Modell von F' auch ein Modell von F ist (ebenfalls geschrieben als $F' \models F$).[20]

Eine Axiomatisierung der DL auf der Basis der Prädikatenlogik erster Stufe läßt sich direkt über die Abbildung von Klassen auf monadische (Typ-)Prädikate und die Abbildung von Rollen auf dyadische Prädikate herleiten [Bor96]. Nachstehend wird dazu die Transformation μ induktiv definiert, welche an der Definition aus [Vol04, Abschnitt 4.4] angelehnt ist ($A \in \mathbf{C}, R \in \mathbf{R}, \doteq$ wird mit der Identität über D interpretiert).

$$
\begin{array}{lcl}
\mu_x(A) & = & A(x) \\
\mu_{x,y}(R) & = & R(x,y) \\
\mu_x(\textbf{\textit{top}}) & = & (x \doteq x) \\
\mu_x(\textbf{\textit{bottom}}) & = & \neg(x \doteq x) \\
\mu_x(\textbf{\textit{and}}\ (c,d)) & = & \mu_x(c) \wedge \mu_x(d) \\
\mu_x(\textbf{\textit{or}}\ (c,d)) & = & \mu_x(c) \vee \mu_x(d) \\
\mu_x(\textbf{\textit{not}}\ (c)) & = & \neg\mu_x(c) \\
\mu_x(\textbf{\textit{all}}\ (r,c)) & = & \forall y.\ \mu_{x,y}(r) \rightarrow \mu_y(c) \\
\mu_x(\textbf{\textit{some}}\ (r,c)) & = & \exists y.\ \mu_{x,y}(r) \wedge \mu_y(c)
\end{array}
$$

[19] Die analogen Fälle für *role inclusions* werden im weiteren nicht aufgeführt.
[20] Im weiteren werden die im Rahmen der Semantik eingeführten Begriffe in deutsch belassen.

$$\begin{aligned}
\mu_x(\textbf{\textit{at-least}}\ (n,r)) &= \exists y_1,\ldots,y_n.\ \textstyle\bigwedge_i \mu_{x,y_i}(r) \bigwedge_{i\neq j} \neg(y_i \doteq y_j)\\
\mu_x(\textbf{\textit{at-most}}\ (n,r)) &= \forall y_1,\ldots,y_{n+1}.\ \textstyle\bigwedge_i \mu_{x,y_i}(r) \rightarrow \bigvee_{i\neq j}(y_i = y_j)\\
\mu_{x,y}(\textbf{\textit{inverse}}\ (r)) &= \mu_{y,x}(r)\\
\mu_{x,y}(\textbf{\textit{transitive}}\ (r)) &= R^{(r)}(x,y),
\end{aligned}$$

wobei die Regeln $\forall x, y.\ \mu_{x,y}(r) \rightarrow Q^{(r)}(x,y)$ und
$\forall x, y, z.\ Q^{(r)}(x,y) \vee (Q^{(r)}(x,z) \wedge Q^{(r)}(z,y)) \rightarrow R^{(r)}(x,y)$
dem Endergebnis der Transformation, falls nicht vorhanden, hinzugefügt werden müssen.

Bei jedem Transformationsschritt werden noch nicht vorkommende Variablen für diejenigen erzeugt, die auf der rechten Seite der Gleichungen neu eingeführt werden (y, y_i).[21] Der Unterschied zu [Vol04] besteht allerdings in der letzten Transformationsregel (***transitive***). Dort fehlt diese Gleichung, da nur ***inverse*** als Rollenkonstruktor zugelassen ist. Die letzte Transformationsregel erzeugt sozusagen in einem Transformationsschritt entsprechende Hilfsprädikate für die transitive Relationen.

Die Transformation der *terminological axioms* und *assertions* erfolgt nach folgender Vorschrift.

$$\begin{aligned}
\mu(c \sqsubseteq d) &= \forall x.\ \mu_x(c) \rightarrow \mu_x(d)\\
\mu(o : c) &= \mu_o(c)\\
\mu((o_1, o_2) : r) &= \mu_{o_1,o_2}(r)\\
\mu(o_1 \texttt{ = } o_2) &= (o_1 \doteq o_2)\\
\mu(o_1 \texttt{ != } o_2) &= \neg(o_1 \doteq o_2)
\end{aligned}$$

2.1.2.5 Ableitungen

DL-Anfragen sind im wesentlichen diejenigen, die sich mit dem Prädikat $\sqsubseteq$ in Bezug auf die TBox und mit dem Prädikat $:$ in Bezug auf die ABox beschreiben lassen. Mit der ersteren Form wird gefragt, ob für zwei *concept terms* c und d die Subklassenbeziehung innerhalb einer TBox T gilt, was als *subsumption problem* bezeichnet wird:

- **Subsumption:** c wird von d *bezüglich* T *subsumiert*, falls $T \models c \sqsubseteq d$.

[21] Analog zu [Vol04, Abschnitt 4.4] ist die Beschränkung auf zwei Variablen, wie sie in [Bor96, HMS04] vorgenommen wird, für unsere Zwecke nicht notwendig.

Eine der weiteren wesentlichen Fragestellungen ist es, zu überprüfen, ob eine Klasse, die innerhalb der TBox definiert wurde, fehlerfrei ist in dem Sinne, daß sie überhaupt instantiierbar ist (Erfüllbarkeit der Klasse). Des Weiteren ist es von Bedeutung, ob eine ungewollte Redundanz vorliegt und zwei Klassen identisch (Klassenäquivalenz) sind oder ob sie gemeinsame Instanzen halten könnten (Klassendisjunktheit).

All diese Fragestellungen lassen sich auf die Subsumption reduzieren: c ist *erfüllbar bezüglich* T, falls $T \not\models c \sqsubseteq$ ***bottom***. c und d sind *äquivalent* bezüglich T, falls $T \models c \sqsubseteq d$ und $T \models d \sqsubseteq c$. c und d sind *disjunkt* bezüglich T, falls $T \models$ ***and***$(c, d) \sqsubseteq$ ***bottom***. Von praktischer Bedeutung ist die Reduktion der Fragestellungen auf die der Unerfüllbarkeit. Verfügt die Konstruktormenge über die Konjunktion und allgemeine Negation (also nicht nur Negation der gegebenen Klassennamen), so lassen sich alle anderen Fragestellungen auf die Unerfüllbarkeit reduzieren.

Ein Beispiel für die Erfüllbarkeit ist mit Fragment 2.5 gegeben. Dort ist die Klasse *Person* erfüllbar bzgl. der TBox T, da *Person* $\sqsubseteq$ ***bottom*** nicht aus T folgt. Anders ist dies bei Hinzunahme des Ausdruck ***at-least****(2, alter)* innerhalb der Zeile 2. Da nämlich in jedem Modell I von T diese Zeile nur mit der leeren Menge interpretiert werden kann und die Interpretation von *Person* somit auch leer sein muß, ist natürlich *Person* $\sqsubseteq$ ***bottom*** gültig in I, womit $\mathsf{T} \models$ *Person* $\sqsubseteq$ ***bottom*** gilt, d.h. *Person* unerfüllbar bzgl. T ist.

Die wesentliche Frage an eine ABox A ist die der Konsistenz zusammen mit einer TBox T:

- **Konsistenz:** A ist *konsistent bezüglich* T, falls ein Modell von $A \cup T$ existiert.[22]

Weitere Fragestellung an die ABox werden mit dem Instanzprädikat $:$ gebildet. Geprüft werden soll, ob $T \cup A \models o : c$ oder $T \cup A \models (o_1, o_2) : r$ (*instance checking*). Steht anstelle von o eine Variable, so spricht man in der DL-Terminologie vom *retrieval problem*. Steht anstelle von c eine Variable und sollen nur die spezifischsten Klassen geliefert werden, so spricht man vom *realization problem*. Analog läßt sich auch die Abfrage von Rollenbeziehungen parametrisieren.

Ein Beispiel für eine konsistente ABox ist wiederum Fragment 2.5. Nehme man aber an, daß in der Zeile 12 statt *(maria, theater)* das Paar *(maria, hans)* stehen würde. In einer Interpretation

[22] Man beachte, daß die Formulierung „A konsistent bzgl. T“ mehr der Vorgehensweise in DL entspricht und nicht der gegebenen Symmetrie. Ist $T = \emptyset$, so spricht man nur von der Konsistenz von A.

müßte dann zufolge Zeile 3 $\alpha(\textit{hans}) \in \alpha(\textit{Bereich})$ und zufolge 8 $\alpha(\textit{hans}) \in \alpha(\textit{Person})$ gelten, was aber wegen Zeile 7 nicht sein kann. Somit kann kein Modell von $\mathsf{T} \cup \mathsf{A}$ existieren.

Die Instanzprüfungen für die ABox können geradlinig über die Folgerungsbeziehung durchgeführt werden. Würde die Zeile 8 aus Fragment 2.5 fehlen, so könnte man diese ableiten: da für Klaus nicht mehr als ein Alter angegeben ist und all seine Hobbys aus einem Bereich stammen, ist er in jeder entsprechenden Interpretation Element der Menge von Personen ($\mathsf{T} \cup \mathsf{A} \models \alpha(\textit{klaus}) : \alpha(\textit{Person})$).

2.1.2.6 Erweiterungen

Die übliche Erweiterung zu einer ausdrucksstärkeren DL wird durch Hinzunahme neuer Konstruktoren vorgenommen. Ein Beispiel dafür ist die Erweiterung der Konstruktormenge, die als Grundlage für die DAML-OIL dient [BHS03].[23] Die Konstruktormenge wird dort um den sogenannten *nominal*-Konstruktor `{}` erweitert, der es ermöglicht, aus Individuen Klassen zu bilden, was Ausdrücke wie

some *(hobby, {fußball, theater})*

erlaubt.

Als zweite Konstruktoren-Erweiterung von $\mathbf{FC}$ werden die sogenannte *qualified number restrictions* ***at-least*** (n, r, c) und ***at-most*** (n, r, c) eingeführt (siehe Seite 22). Die auf Seite 26 angegebene Definition für α im unqualifizierten Fall wird dabei dahingehend verändert, den Zielbereich von $\alpha(r)$ auf Elemente aus $\alpha(c)$ einzuschränken, genauer: $\hat{r}(x, c) := \{y \in \alpha(c) \mid (x, y) \in \alpha(r)\}$. Beispielsweise würde

at-least *(2, hobby, {fußball, theater, kino})*

für diejenigen stehen, die mindestens zwei Hobbys unter den angegebenen haben.

[23] Bei dieser DL handelt es sich um die Klasse $\mathcal{SHOIQ}$ (zur Bildung der Kürzel siehe [BCM+03, Appendix 1]).

2.2 Web-orientierte Sprachen

2.2.1 Resource Description Framework

Die Objekte des *World Wide Web* (WWW) sind zumeist Dokumente, deren Repräsentation fast ausschließlich Sprachelemente zur Dokumentpräsentation umfassen, also Elemente zur logischen Dokumentstruktur (Absätze, Fußnoten etc.) und zum Layout (Fonts, Farben etc.). Um jedoch das WWW als ein Informationssystem nutzen zu können, wie es in [BLHL01, Las98] als Vision unter dem Begriff *Semantic Web* formuliert wurde, müssen den Dokumenten Metadaten zu ihrer Bedeutung zugeordnet werden. Das *Resource Description Framework* (RDF) ist eine solche Metadatenstandardisierung. Das *World Wide Web Consortium* (W3C) hat dazu eine Reihe von Standardisierungsempfehlungen gegeben, welche die Konzeption und abstrakte Syntax [KC04], die konkrete Syntax in XML-Form [Bec04b], die Schema-Sprache [BG04] und die formale Semantik [Hay04] umfassen.

Wie in [Las98] angeführt, wurde RDF nicht nur von der WWW-Gemeinde selber beeinflußt, sondern maßgeblich von den Bereichen Wissensrepräsentation, Informationsmodellierung und strukturierte Dokumente. Damit verfügt es über alle wesentlichen Konstrukte, die verschiedensten Arten von Information einheitlich zu repräsentieren. Neben der Repräsentation klassischer Metadaten für elektronische Dokumente (z.B. *Dublin Core*) und der Verwendung zur klassenbasierten Modellierung können mit RDF ganz allgemein Begriffsnetze repräsentiert werden, wie sie für Thesauri und Glossare relevant sind [WSWS01]. Während XML als eine Sprache zur Darstellung hierarchischer Strukturen entworfen wurde, erhebt RDF den Anspruch, eine Art 'lingua franca' für Metadaten beliebiger Art zu sein. Um dieser Allgemeinheit gerecht zu werden, ist das Kernmodell von RDF graphenbasiert. Im weiteren wird ein Überblick über das in [KC04, Hay04] definierte Kernmodell gegeben.

Da RDF ursprünglich als Sprache zur Beschreibung von Web-Ressourcen intendiert war, ist einer der Basisbausteine der sogenannte *Uniform Resource Identifier* (URI, [BLFIM98]). Ein URI steht nicht nur für Ressourcen des Web, sondern dient auch der Denotation der Bestandteile von Ontologien wie Klassen, Eigenschaften und Instanzen ([KC04, Abschnitt 2.2.3]: „URI references are used for naming all things in RDF"). Die zweiten Basisbausteine sind *Literale*. Zum einen können dies *einfache Literale* sein, welche Zeichenketten sind, die nicht wie URIs ‚Dinge der Welt' bezeichnen sondern die für sich stehen. Die zweite Form, *getypte Literale*, sind Paare (z, xsd), wobei z eine Zeichenkette und xsd ein *XML-Datatype* ist [BPM04]. Die dritten Basisbausteine sind die sogenannten *blank nodes*, auf die später näher eingegangen wird.

2.2.1.1 RDF-Graphen

Das bisher Erläuterte präzisierend seien als Alphabet die folgenden paarweise disjunkten Mengen vorausgesetzt: eine Menge von URIs $\mathbf{U}$, eine Menge von Literalen $\mathbf{L}$ und eine nichtendliche Menge von *blank nodes* $\mathbf{B}$. Ein Element aus $\mathbf{U} \cup \mathbf{L}$ heißt *RDF-Name* und eine Menge von RDF-Namen wird *Vokabular* genannt. Ein *RDF-Tripel* (oder auch *RDF-statement*) ist ein Tripel (s, p, o) mit $s \in \mathbf{U} \cup \mathbf{B}$, $p \in \mathbf{U}$ und $o \in \mathbf{U} \cup \mathbf{B} \cup \mathbf{L}$, wobei s, p und o als *Subjekt*, *Prädikat* (oder *property*) resp. *Objekt* bezeichnet wird. Eine Menge solcher Tripel wird *RDF-Graph* genannt.[24] In älteren Versionen des Standards wird auch von *RDF-Modellen* gesprochen. Kommen in einem Graph keine *blank nodes* vor, so spricht man von einem *Grundgraphen*.

Der Standard enthält die Definition der konkreten Syntax für RDF als XML-Form [Bec04b]. Die konkrete Syntax von RDF-Graphen könnte aber auf mehrere Arten angegeben werden (z.B. mit *N-Triple* aus [GB04]). Die XML-Basierung unterstützt jedoch auf einfache Art und Weise eine Integration von Metadaten in vorhandene Web-Ressourcen.

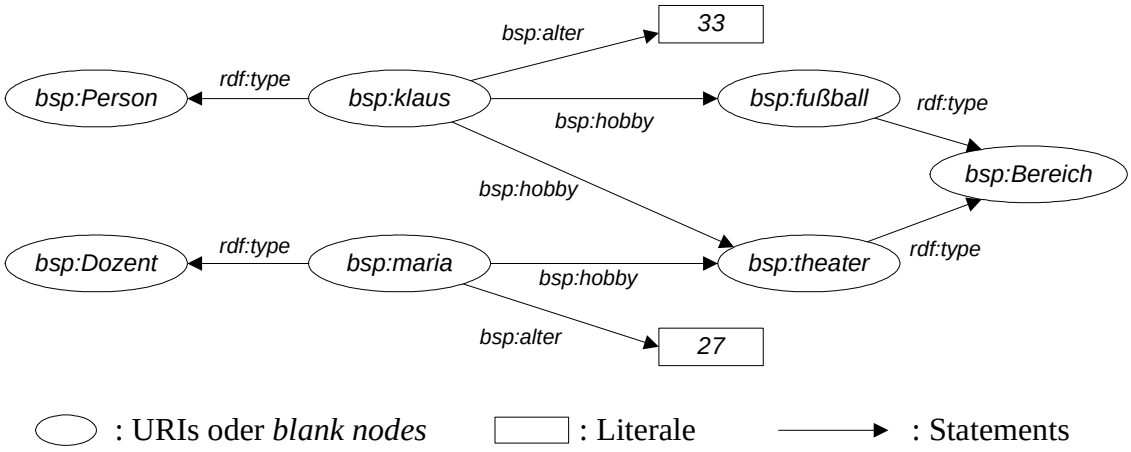

Bild 2.2: RDF-Graph G

Mit Hilfe von RDF-Graphen werden nun benannte Begriffsbeziehungen angegeben. Beispielsweise zeigt Bild 2.2 einen RDF-Graphen, der den gleichen Zusammenhang wie ein Ausschnitt aus dem F-Logic-Fragment 2.1 (Seite 10) beschreiben soll. Man beachte, daß hier nicht die

[24] Die Parallele zur üblichen Graphendefinition ist offensichtlich. Man beachte jedoch, daß ein einfacher kantenmarkierter Graph, wie in [Vol04, Abschnitt 3.3.2] angegeben, nicht ausreicht, da es zwischen zwei Knoten mehrere Kanten mit unterschiedlicher *property* geben kann.

RDF-Repräsentation von *F-Logic data expressions* gemeint ist, sondern RDF direkt zur Modellierung genutzt wird. Zur Partitionierung der gesamten RDF-Namensmenge ist das Konzept des *Namensraums* von XML übernommen worden.[25] In dem Beispiel aus Bild 2.2 sind zwei Namensräume angegeben. Zum einen der für das Anwendungsbeispiel (*bsp:...*) und zum anderen der für das RDF-Standard-Vokabular (*rdf:...*), für die spätere Präzisierung mit V_{RDF} bezeichnet. Das wesentliche Element dieses Vokabulars ist *rdf:type*, womit die Instantiierungsbeziehung gesetzt wird. Einige weitere Elemente sind *rdf:List*, *rdf:first*, *rdf:rest* und *rdf:nil*, womit klassische Listen wie aus der funktionalen Programmierung angegeben werden können oder die Elemente *rdf:Seq*, *rdf:Bag* und *rdf:Alt*, womit klassische Container wie Vektoren, Multimengen und Alternativen ausdrückbar sind.[26]

Bezüglich der Klasse-Instanz-Relation gehört RDF, so wie auch F-Logic, DL und XML, zu denjenigen Sprachen, die erlauben, beliebige Beziehungssetzungen vorzunehmen, ohne vorher den Raum der möglichen Beziehungssetzungen festlegen zu müssen. In XML sind dies einfach nur wohlgeformte statt valide Dokumente, in DL ist dies nur eine ABox und eine leere TBox (siehe Abschnitt 2.1.2.3), in F-Logic sind dies nur *data expressions* ohne entsprechende *signature expressions* (siehe Abschnitt 2.1.1.2) und in RDF sind dies einfach RDF-Graphen. Die Constraints, welche Beziehungssetzungen erlaubt sind, werden dann mit Hilfe von speziellen Konstrukten formuliert. Die *RDF Vocabulary Description Language RDF-Schema* (RDFS, [BG04]), welche im weiteren erläutert wird, wurde für diese Zwecke definiert.

2.2.1.2 RDF-Schemata

RDF-Schemata werden selbst wieder als RDF-Graphen angegeben, basierend auf V_{RDF} und einem speziellen Vokabular V_{RDFS}. Unter Verwendung dieser Vokabulare werden Schemata für spezifische Anwendungen definiert, welche auf der gleichen Ebene wie Modelle in UML angesiedelt sind. Die wesentlichen Elemente sind nachstehend aufgeführt, wobei Elemente für Container, Datentypen und für die Reifikation der Klarheit wegen weggelassen werden.

- *rdfs:Resource*, *rdfs:Class*, *rdfs:Literal*, *rdf:Property* und *rdf:type*: Elemente zur Definition von Klassen, Eigenschaften und Instantiierungen.

- *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain* und *rdfs:range*: Elemente zur

[25] In der Form $x : y$ wird x Namensraum und y *lokaler Name* genannt.

[26] Details zur Nutzung dieser Konstruktoren und zur Reifikation, auf die hier nicht näher eingegangen wird, sind in [MM04, Abschnitt 4] zu finden.

Definition der Subklassenbeziehung, der Subsumptionsbeziehung für Eigenschaften und der Quell- und Zielklasse von Eigenschaften.

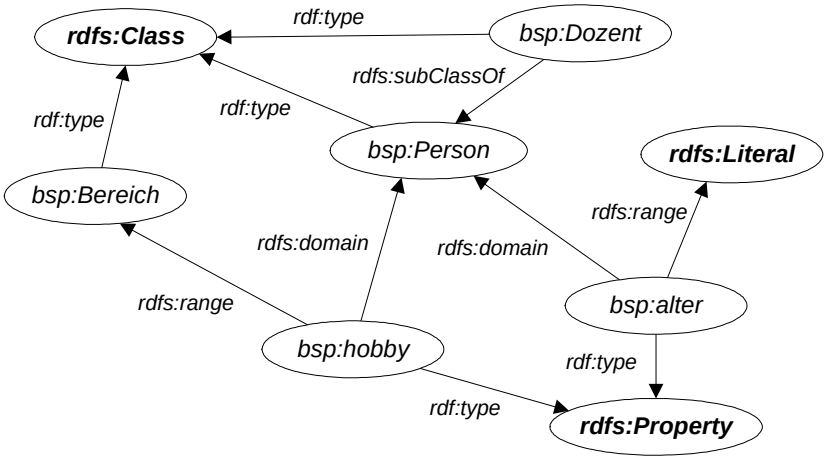

Bild 2.3: RDF-Schema S

Bild 2.3 zeigt ein solches Schema zu dem Graphen aus Bild 2.2. Das Schema definiert die Anwendungsklassen *Person*, deren Subklasse *Dozent* und die Klasse *Bereich* sowie die Eigenschaften *alter* und *hobby* mit entsprechenden Wertebereichen. Auf die Interpretation des Schemas zur Validierung wird in 2.2.1.5 näher eingegangen.

2.2.1.3 Semantik

Die formale Semantik von RDF wird in [Hay04] modelltheoretisch definiert, wobei wie üblich eine Interpretation in einer Struktur der Ausgangspunkt ist. Zunächst definiert der Standard eine sogenannte *einfache* Interpretation. Dazu bezeichne V_U die Menge der URIs, V_{SL} die Menge der einfachen und V_{TL} die Menge der getypten Literale aus einem Vokabular V. Eine *(einfache) Interpretation* $I = (R, P, ext, r, l)$ *eines Vokabulars* V besteht aus[27]

- einer nicht-leeren Menge $R \supseteq V_{SL}$, auch *Universum* oder *Ressourcenmenge* von I genannt,

[27] Im weiteren bezeichne $\mathbb{P}$ den Potenzmengenoperator. Der Einfachheit wegen wird die in der originalen Definition aufgeführte Menge $LV \subseteq R$ hier nicht berücksichtigt.

- einer Menge P (*Properties* genannt),
- einer Abbildung $ext : P \longrightarrow \mathbb{P}(R \times R)$,
- einer Abbildung $r : V_U \longrightarrow R \cup P$ und
- einer Abbildung $l : V_{TL} \longrightarrow R$.

In Bild 2.4 (vergl. auch [Hay04, Bild 1]) sind die Bestandteile einer Interpretation dargestellt. Im weiteren bezeichnen R, P, ext, r und l immer diese Mengen bzw. Abbildungen, falls nicht anders angegeben.

Auf syntaktischer Seite hat man zunächst, erstmal ohne Berücksichtigung von Tripeln, das gegebene Vokabular bestehend aus Literalen und URIs. Einfache Literale werden, da sie semantisch für sich stehen, in die Ressourcenmenge eingebettet. Getypte Literale werden auf die Ressourcenmenge und URIs auf den gesamten Grundbereich abgebildet, der zusätzlich zur Ressourcenmenge die Property-Menge umfaßt. Den Elementen der Property-Menge werden Mengen von Paaren über R durch die Abbildung ext (Extension einer Property) zugeordnet, was der Intuition entspricht.

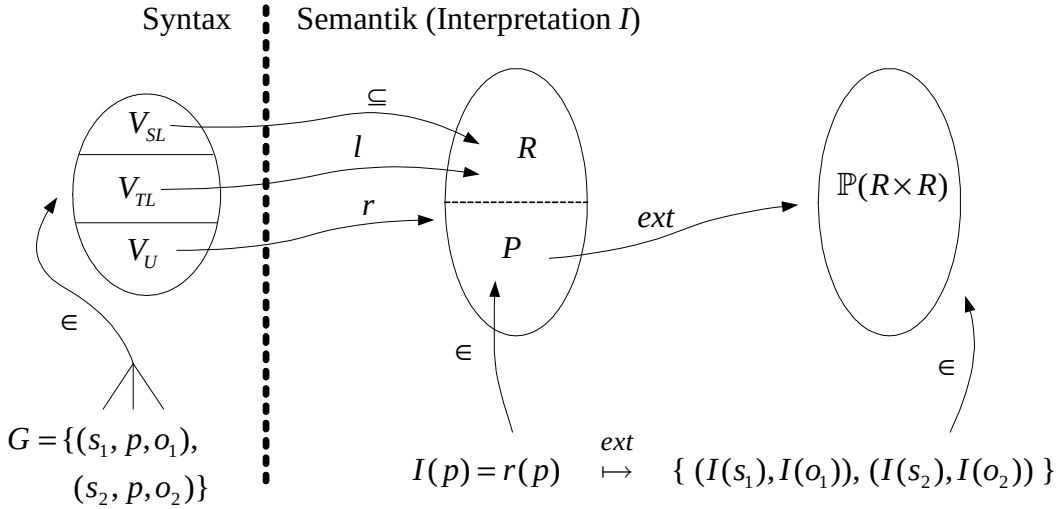

Bild 2.4: RDF-Semantik

Ähnlich zu der Vorgehensweise bei klassischen Interpretationen werden Interpretationen im RDF-Kontext auf Tripelmengen ausgedehnt. Sei dazu G ein RDF-Grundgraph über einem

Vokabular V, $t = (s, p, o) \in G$ und I eine Interpretation von V. I wird wie folgt fortgesetzt: $I(x) = x$, falls $x \in V_{SL}$, $I(x) = l(x)$, falls $x \in V_{TL}$, $I(x) = r(x)$, falls $x \in V_U$ und schließlich

(i) t ist *gültig* in I, falls $I(p) \in P$ und $(I(s), I(o)) \in ext(I(p))$.

(ii) I *erfüllt* G, falls alle $t \in G$ gültig in I sind.[28]

Wie in [Hay04, Abschnitt 1.4] bemerkt, gilt gewöhnlich $P \subseteq R$ für ein Modell eines Grundgraphen G. Genauer: mindestens dann, wenn ein Prädikat p als Subjekt in einem Tripel des Graphen vorkommt, also $(p, x, y) \in G$, muß $I(p) \in R$ sein, da andernfalls $(I(p), I(y)) \notin R \times R$ und damit $(I(p), I(y)) \notin ext(I(x))$, was die Bedingung (i) verletzen würde. Die Argumentation für p in der Objektposition verläuft analog.

Die Bedingungen (i) und (ii) werden im RDF-Kontext *semantische Bedingungen* genannt. Bild 2.4 zeigt einen aus zwei Tripeln bestehenden Graphen G und ein Modell, da genau diese semantische Bedingung gilt.

2.2.1.4 *Blank nodes*

Eine besondere Art von RDF-Namen sind die sogenannten *blank nodes* (auch *anonyme Knoten* genannt), welche für die Existenz einer Ressource stehen. Pragmatisch betrachtet sind sie Bezeichner für Knoten des RDF-Graphen, die nicht für eine bekannte Ressource stehen müssen, sondern lediglich der lokalen Identifizierung von Knoten dienen. Ein Beispiel ist der Graph

{ (*bsp:klaus*, *bsp:wohntIn*, *a*), (*a*, *bsp:stadt*, *'Berlin'*), (*a*, *bsp:teil*, *'Steglitz'*),
(*bsp:klaus*, *bsp:wiegt*, *g*), (*g*, *bsp:wert*, *'120'*), (*g*, *bsp:einheit*, *'Pfund'*)},

der ausdrücken soll, das Klaus in Berlin-Steglitz wohnt und 120 Pfund wiegt. Der *blank node* *a* steht nun für eine Adresse in Steglitz, von der man weiß, das sie existiert. *g* dient als lokaler

[28] Es sei bemerkt, daß hier nicht wie in [Hay04, Abschnitt 1.4] eine boolschen Funktion verwendet wird, sondern der begrifflichen Einfachheit wegen und zur Verdeutlichung der Parallelen zur FOL auch von Gültigkeit und Erfüllbarkeit gesprochen wird, das im folgenden wiederum mit $\models$ notiert wird. Wie üblich spricht man bei einer erfüllenden Interpretation auch von einem *Modell*. Die Folgerungsbeziehung ist wie in der FOL definiert.

Bezeichner des Gewichts, bestimmt durch das Aggregat *(120, Pfund)*. Weitere Beispiele für diese Art RDF-Name sind in [Vol04, Abschnitt 3.3] zu finden.

Formal betrachtet ist dies die Einführung von Variablen mit existentieller Quantifizierung. Wie üblich setzt man dafür eine *Belegung* $\beta : \mathbf{B} \longrightarrow R$ voraus und definiert für eine gegebene Interpretation I und Vokabular V die Erweiterung I_β durch $I_\beta(x) = \beta(x)$ für alle $x \in \mathbf{B}$ und $I_\beta(x) = I(x)$ für alle $x \in V \setminus \mathbf{B}$. Haben wir nun eine Interpretation I und einen RDF-Graphen G, so kann die folgende semantische Bedingung aufgestellt werden:

(i) I ist ein Modell von G, falls eine Belegung β existiert, so daß I_β Modell von G ist.

2.2.1.5 RDF- und RDFS-Interpretationen

Die bisher mögliche Art der Modellbildung für die syntaktischen Objekte (RDF-Graphen) ist noch sehr allgemein. Die einzigen Bedingungen sind mit (i), (ii) aus Abschnitt 2.2.1.3 sowie (i) aus Abschnitt 2.2.1.4 gegeben. Es wird jedoch den Namen der Vokabulare V_{RDF} und V_{RDFS} keinerlei Bedeutung zugewiesen. Dies wird wiederum durch die Aufstellung von semantischen Bedingungen und die Angabe von sogenannten *axiomatischen Tripeln* vorgenommen. Letztere haben dann zusammen mit der semantischen Bedingung gültig zu sein in einer Interpretation, wenn es denn als Modell dienen soll. Der Klarheit wegen berücksichtigen wir im weiteren keine Reifikation, Container und getypten Literale, also nur $V_{RDF} = \{$*rdf:type*, *rdf:Property*$\}$.

Eine *RDF-Interpretation* eines Vokabulars V ist eine einfache Interpretation I des Vokabulars $V \cup V_{RDF}$, für die folgende Bedingungen gelten müssen:

- $x \in P$ genau dann, wenn $(x, I($*rdf:Property*$)) \in ext(I($*rdf:type*$))$ und
- I ist ein Modell von $\{($*rdf:type*, *rdf:type*, *rdf:Property*$)\}$.

Die Bedingungen sorgen dafür, daß *rdf:type* und alle URIs, die in einem RDF-Graphen als Prädikate vereinbart wurden, auch tatsächlich auf semantischer Seite in P enthalten sind. Sie dienen aber mehr als Vorbereitung auf die wesentlich wichtigeren Bedingungen für eine Interpretation, nämlich die Bedingungen, welche die Validität bezüglich eines RDF-Schemas einbeziehen. Denn aus pragmatischer Sicht ist RDF als Sprache zur Aufstellung von Ontologien zu betrachten, bei der die Frage der Validität eine entscheidende Rolle spielt.

Eines der wesentlichen Konstrukte in Sprachen zur Informationsmodellierung ist das der Klassenbildung, in RDFS durch das Konstrukt *rdfs:Class* getragen. Zum Zwecke der Präzisierung (vgl. [Vol04, Abschnitt 3.3.5]) ist für eine Interpretation I die Abbildung $cext : R \longrightarrow \mathbb{P}(R)$ (*Klassenextension*) definiert durch

$$cext(x) = \{y \mid (y, x) \in ext(I(\textit{rdf:type}))\},$$

die einer Ressource x die Menge derjenigen Ressourcen zuordnet, die Instanzen von x sind. Beispielsweise ergeben sich aus Bild 2.2 unter anderem folgende Extensionsmengen in einer Herbrand-Interpretation: [29] $cext(\textit{bsp:Person}) = \{\textit{bsp:klaus}\}$ und $cext(\textit{bsp:Bereich}) = \{\textit{bsp:fußball}, \textit{bsp:theater}\}$. Aus Bild 2.3 können $cext(\textit{rdfs:Class}) = \{\textit{bsp:Bereich}, \textit{bsp:Person}, \textit{bsp:Dozent}\}$ und $cext(\textit{rdf:Property}) = \{\textit{bsp:hobby}, \textit{bsp:alter}\}$ bestimmt werden.

Die Semantik der in Abschnitt 2.2.1.2 auf Seite 32 eingeführten RDFS-Konstrukte wird, wie schon bei den RDF-Konstrukten, über eine spezielle Interpretation festgelegt: eine *RDFS-Interpretation* eines Vokabulars V ist zunächst einmal eine RDF-Interpretation I von $V \cup V_{RDF} \cup V_{RDFS}$. Neben der semantischen Bedingung, das R wirklich die Menge der Ressourcen denotiert, nämlich $R = cext(I(\textit{rdf:Resource}))$, ist vor allem die folgende Auswahl von Bedingungen in einer Anwendung von Bedeutung:

(i) Falls $(x, y) \in ext(p)$ und $(p, c) \in ext(I(\textit{rdfs:domain}))$, so ist $x \in cext(c)$.

(ii) Falls $(x, y) \in ext(p)$ und $(p, c) \in ext(I(\textit{rdfs:range}))$, so ist $y \in cext(c)$.

(iii) $ext(I(\textit{rdfs:subClassOf}))$ und $ext(I(\textit{rdfs:subPropertyOf}))$ sind transitiv und reflexiv.

(iv) Falls $(x, y) \in ext(I(\textit{rdfs:subPropertyOf}))$, so sind $x, y \in P$ und $ext(x) \subseteq ext(y)$.

(v) Falls $(x, y) \in ext(I(\textit{rdfs:subClassOf}))$, so sind $x, y \in cext(I(\textit{rdfs:Class}))$ und $cext(x) \subseteq cext(y)$.

Die ersten beiden Bedingungen dienen der Bestimmung der Signaturverträglichkeit: falls eine Ressource x die Eigenschaft p besitzt und zum Quellbereich von p die Klasse c gehört, so muß x eine Instanz von c sein. Die Bedeutung der Zielklassen-Bedingung ist analog. Auch die

[29] Zur allgemeinen Definition der Herbrand-Interpretation siehe [Llo87, Kapitel 1] und zur RDF-Adaption [Hay04, Appendix A].

letzten drei Bedingungen folgen der in der Informationsmodellierung üblichen Semantik der Spezialisierungs-Konstruktoren für Eigenschaften und für Klassen, nämlich deren Transitivität und Reflexivität sowie deren Interpretation mit der Teilmengenbeziehung.

Ferner müssen eine Reihe von axiomatischen Tripeln in einer RDFS-Interpretation gültig sein, von denen hier exemplarisch sechs die Signaturen betreffende aufgelistet sind.

(vi) (*rdfs:domain*, *rdfs:domain*, *rdf:Property*), (*rdfs:domain*, *rdfs:range*, *rdfs:Class*), (*rdfs:range*, *rdfs:domain*, *rdf:Property*), (*rdfs:range*, *rdfs:range*, *rdfs:Class*) und (*rdfs:subClassOf*, *rdfs:domain*, *rdf:Class*), (*rdfs:subClassOf*, *rdfs:range*, s*rdf:Class*).

Diese axiomatischen Tripel bilden die Signaturen für die Konstrukte der Schema-Sprache und stellen damit das RDF-Schema für RDF-Schemata dar. Eine genaue Aufführung der restlichen Bedingungen für RDF- und RDFS-Interpretationen ist in [Hay04, Kapitel 3 u. 4] zu finden.

Am besten läßt sich die Bedeutung der Bedingungen wiederum am Herbrandmodell H der Graphen aus Bild 2.2 und 2.3 verdeutlichen. Zunächst einmal sind per definitionem die Tripel (*bsp:maria*, *rdf:type*, *bsp:Dozent*) und (*bsp:Dozent*, *rdfs:subClassOf*, *bsp:Person*) gültig in H. Der klassischen Semantik in der Informationsmodellierung folgend müßte wegen der Subklassenbeziehung auch gelten, daß *bsp:maria* Instanz von *bsp:Person* ist. Genau dies wird durch eine RDFS-Interpretation erzwungen. Soll nämlich H eine RDFS-Interpretation sein, so gilt aufgrund von Bedingung (v) $cext(\textit{bsp:Dozent}) \subseteq cext(\textit{bsp:Person})$, also der Tripel (*bsp:maria*, *rdf:type*, *bsp:Person*). Desweiteren folgt aus (*bsp:klaus*, *bsp:hobby*, *bsp:fußball*) und (*bsp:hobby*, *rdfs:range*, *bsp:Bereich*) wegen (ii) der in Bild 2.2 schon gegebene Tripel (*bsp:fußball*, *rdf:type*, *bsp:Bereich*).

2.2.1.6 *Entailment rules*

Aufgrund der modelltheoretischen Definition der Semantik ist man sehr gut in der Lage, eine wichtige Fragestellung zu präzisieren, nämlich die semantische Äquivalenz von zwei RDF-Graphen, die formuliert wird als die wechselseitige Folgerungsbeziehung[30] zwischen ihnen. Eines der für die Praxis relevanten Ergebnisse dazu ist das *Interpolations-Lemma*, das besagt, daß zur Feststellung der Folgerungsbeziehung lediglich die Subgraph-und Instanz-Eigenschaft überprüft werden muß (siehe [Hay04, Kapitel 2]).

[30] Sie wird im RDF-Rahmen *simple entailment* genannt (siehe auch Fußnote 28).

Genau wie Interpretationen nur bezüglich eines Vokabulars betrachtet werden können, kann die Folgerungsbeziehung auch auf ein Vokabular V bezogen werden. Genauer: G *V-entails* G', falls in jeder V-Interpretation, in der G gültig ist, auch G' gültig ist. Von besonderer Wichtigkeit in diesem Zusammenhang ist das RDFS-*Entailment*. Zwei Graphen G, G' können als äquivalent bezüglich der RDFS-Semantik betrachtet werden, falls G RDFS-*entails* $G2'$ und umgekehrt. Mit dem *RDFS entailment Lemma*, dessen konstruktiver Beweis [Hay04, Appendix A] auf den sogenannten *RDFS entailment rules* [Hay04, Kapitel 7] und den axiomatischen Tripeln für RDFS aufbauen, ist ein Verfahren zur Überprüfung gegeben. Diese *entailment rules* stellen sozusagen eine Operationalisierung der Bedingungen (i) bis (v) aus Abschnitt 2.2.1.5 für die Konstruktion einer RDFS-Herbrandinterpretation dar. Drei von ihnen seien hier exemplarisch aufgeführt, wobei c, d, p, x, y entsprechende Variablen seien:

(rdfs2) Falls $(p,$ *rdfs:domain*$, c)$ und (x, p, y) in G enthalten sind, so füge $(x,$ *rdf:type*$, c)$ zu G hinzu.

(rdfs3) Falls $(p,$ *rdfs:range*$, c)$ und (x, p, y) in G enthalten sind, so füge $(y,$ *rdf:type*$, c)$ zu G hinzu.

(rdfs9) Falls $(d,$ *rdfs:subClassOf*$, c)$ und $(x,$ *rdf:type*$, d)$ in G enthalten sind, so füge $(x,$ *rdf:type*$, c)$ zu G hinzu.

Die Verwendung dieser Regeln zur Überprüfung der Validität eines RDF-Graphen bezüglich eines Schemas wird im nächsten Abschnitt im Zusammenhang mit der axiomatischen Semantik erläutert.

2.2.1.7 Zur axiomatischen Semantik

Bis zur Verabschiedung des Standards zur RDF-Semantik im Februar 2004 wurde in mehreren Veröffentlichungen eine Semantik beschrieben, und zwar nicht modelltheoretisch, sondern indirekt über eine Transformation in Sätze einer logischen Sprache mit bereits definierter Semantik, was auch als axiomatische Semantik bezeichnet wird.[31] [FM01] verwendet hierfür das *Knowledge Interchange Format* (KIF) als Zielsprache, in [GH03] wird die Sprache $\mathrm{L_{BASE}}$ definiert und [CK00] nutzt Datalog als Sprache zur logischen Repräsentation der Tripel.

Die allgemeine Transformation von Tripeln nach FOL kann wie folgt durchgeführt werden: ein Tripel (s, p, o) wird auf die atomare Formel $statement(s, p, o)$ eines dreistelligen Prädikats

[31] Es sei bemerkt, daß die Transformation auch über mehrere Stufen erfolgen kann.

$statement$ abgebildet.[32] Die in den Tripeln vorkommenden RDF-Namen werden auf Konstanten und die *blank nodes* auf unterschiedliche Variablen abgebildet. Ein RDF-Graph wird dann als Existenzabschluß der Konjunktion aller seiner transformierten Tripel repräsentiert.

Eine der Hauptinteressen ist die logische Folgerung von Formeln aus einem transformierten RDF-Graphen unter Anwendung von Regeln. Für diese Zwecke reicht es, lediglich die Skolem-Form zu betrachten. Die Skolemisierung einer (keine Allquantoren beinhaltenden) Formelmenge (hier notiert mit sk) eliminiert den Existenzquantor und seine Variablen, indem jede durch ihn gebundene Variable mit einer neuen nicht im Original vorkommende Konstante ersetzt wird. Aufgrund der Erfüllbarkeitsäquivalenz der Skolemisierung gilt nun $sk(F) \models x$ gdw. $F \models x$ für eine Formelmenge F und eine Formel x, die keine in F vorkommenden Skolem-Konstanten enthält. Damit ist es möglich, uns bei Fragen der Folgerung auf $sk(F)$ zu beschränken (vergl. auch [Hay04, Appendix A] / *Skolemization Lemma*) und die so transformierten Tripel in Datalog zu repräsentieren.

Die semantischen Bedingungen für RDFS einschließlich der für RDF bzw. die daraus hergeleiteten *entailment rules* sowie die axiomatischen Tripel werden wie eben beschrieben in eine Menge von Datalog-Regeln resp. -Fakten transformiert, die im weiteren mit ER_{RDFS} bezeichnet wird. Die drei Regeln (rdfs2), (rdfs3) und (rdfs9) können dann mit lediglich kleiner syntaktischer Änderung übernommen werden:

(rdfs2Log)
forall *x,c,p,y*
statement (x, rdf:type, c) ← *statement (p, rdfs:domain, c)* **and** *statement (x, p, y).*

(rdfs3Log)
forall *x,c,p,y*
statement (y, rdf:type, c) ← *statement (p, rdfs:range, c)* **and** *statement (x, p, y).*

(rdfs9Log)
forall *x,c,d*
statement (x, rdf:type, c) ←
statement (d, rdfs:subClassOf, c) **and** *statement (x, rdf:type, d).*

Eine axiomatische Semantik eines RDF-Graphen G auf der Grundlage von Datalog ist gegeben durch die wohlbekannte Datalog-Semantik für $\mu(G) \cup ER_{\text{RDFS}}$, wobei μ die oben erläuterte Transformation nach Datalog bezeichnet.

Die Regeln aus ER_{RDFS} können nun unterschiedlich verwendet werden. Zum einen in der beabsichtigten konstruktiven Weise derart, daß die durch Regelanwendung entstehenden Tripel

[32] In [FM01] wird der Prädikatsname *PropertyValue* und in [CK00] der Name *statement* verwendet.

zu einem Graphen hinzugefügt werden. Zum anderen können sie, wie es üblich ist in der Datenmodellierung, als Constraints zur Überprüfung der Validität bezüglich eines RDF-Schemas herangezogen werden. Im letzteren Fall würde man überprüfen, ob bestimmte durch Regelanwendung entstehende Tripel aus dem Instanz-Graphen ableitbar sind.

Nehme man beispielsweise an, daß statt des Statements *(bsp:klaus, bsp:hobby, bsp:fußball)* das Statement *(bsp:klaus, bsp:hobby, bsp:maria)* in dem Graphen G aus Bild 2.2 auf Seite 31 angeben wäre. Mit Hilfe der Regel (rdfs3Log) und dem entsprechenden Tripel aus dem Graphen S (Bild 2.3) läßt sich dann die folgende Ableitung vornehmen.

statement (bsp:hobby, rdfs:range, bsp:Bereich) $\in$ S
statement (bsp:klaus, bsp:hobby, bsp:maria) $\in$ G
$\vdash_{\text{rdfs3}}$
statement (bsp:maria, rdf:type, bsp:Bereich)

Da aber $\mathsf{G} \cup \mathsf{S} \cup ER_{\text{RDFS}} \nvdash$ *statement (bsp:maria, rdf:type, bsp:Bereich)*, würde mit *(bsp:klaus, bsp:hobby, bsp:maria)* der Graph G nicht mehr valide bezüglich S sein. Eine Umformung der *entailment rules* derart, daß sie direkt als Constraints verwendbar sind, indem Prädikate wie *range_violation* angeboten werden, die das eben erläuterte Verfahren realisieren, sind in [CK00] zu finden, wenngleich auf einer älteren Version des Standards basierend [BG00].

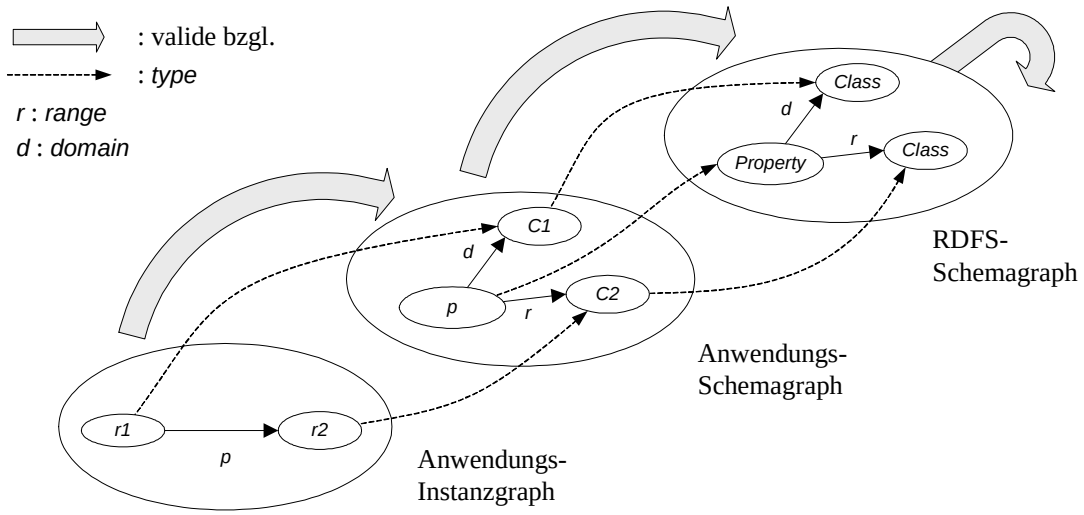

Bild 2.5: RDFS-Hierarchie

Aus der Validitätsbeziehung ergibt sich eine Hierarchie, die in Bild 2.5 dargestellt ist. Oben

rechts steht das RDF-Schema für RDFS selber, gegeben durch die axiomatischen Tripel für eine RDFS-Interpretation (siehe 2.2.1.5.(vi)), dessen Validität selbstbezüglich definiert ist, worauf im nächsten Abschnitt noch näher eingegangen wird. Dieses Schema bildet die Signatur-Constraints für das Anwendungsschema, welches wiederum die Signatur-Constraints für den Instanzgraphen der Anwendung bildet.

2.2.1.8 Zur Paradoxie

Eine nahe liegende modelltheoretische Semantik für RDF wäre diejenige, die Prädikate nicht, wie im Standard definiert, auf eigenständige Objekte der Trägermenge abbildet, denen dann Extensionen zugeordnet werden, sondern jedem Prädikat direkt eine Relation innerhalb der Interpretation zuweist. Bild 2.6 zeigt eine schematische Darstellung derartiger Interpretationen, welche im Gegensatz zur Interpretation des Standards aus 2.2.1.3 durch die Prädikatszuordnung

$$p \mapsto I(p) \subseteq R \times R \ .$$

gekennzeichnet sind (vergl. Bild 2.4 auf Seite 34).

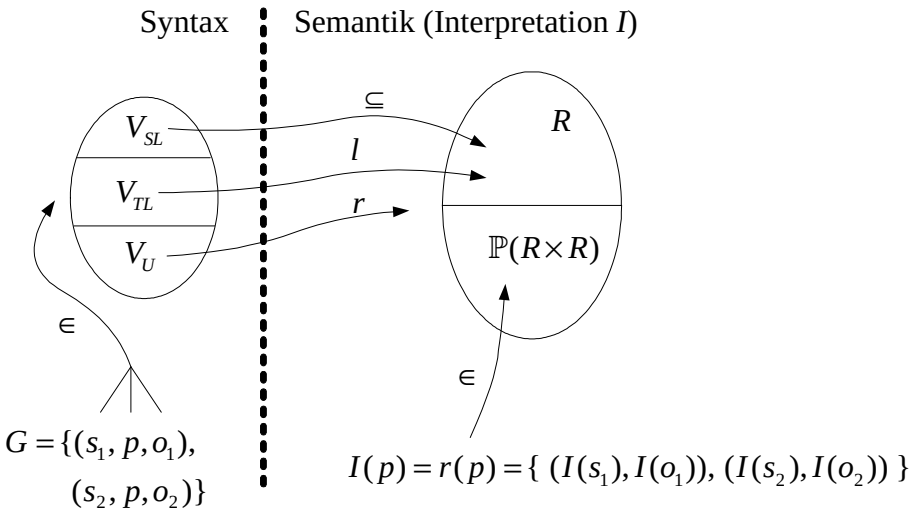

Bild 2.6: Semantik mit direkter Prädikatszuordnung

Diese Art Interpretationen haben jedoch zwei wesentliche Konsequenzen, die an einem Beispiel verdeutlicht werden sollen. Dazu seien die Tripel

(Modul, ***subClassOf****, Unit)*,
(nutztSchnittstelleVon, ***subPropertyOf****, istAbhängigVon)*.

aus der Modellierung von Software-Artefakten gegeben, welche in einem RDF-Repository abgelegt sind. Möchte ein Nutzer das Repository abfragen nach allen Tripeln, die als Prädikat eines der beiden subsumierenden Prädikate besitzen, so kann er dies über die Vereinigung zweier Anfragen mit jeweils einem der Prädikate tun. Eine wesentlich elegantere und angemessenere Art, zumal wenn es eine Vielzahl dieser Prädikate gibt, ist die Einführung des Prädikats *subsumingProperty*, welches eine Generalisierung der beiden sein soll, also

*(****subClassOf****,* ***subPropertyOf****, subsumingProperty)*,
*(****subPropertyOf****,* ***subPropertyOf****, subsumingProperty)*. $(*)$

Anfragen mit Hilfe von *subsumingProperty* leisten nun das Gewünschte. Hierbei ergibt sich jedoch die erste Konsequenz: ist der Tripel $(*)$ in einer Interpretation I gemäß der Semantik der direkten Prädikatszuordnung gültig, so muß

$$I(\textbf{\textit{subPropertyOf}}) = \{(I(\textbf{\textit{subPropertyOf}}), I(\textit{subsumingProperty})), \ldots\}$$

sein. Dies jedoch läßt die zugrunde gelegte ZF-Mengenlehre nicht zu.[33]Eine zweite Konsequenz ist die, daß ohne einen Übergang zur Logik höherer Ordnung, deren Unvollständigkeit den praktischen Einsatz ausschließt, keine Quantifizierung über Eigenschaften mehr möglich ist, was in der Praxis jedoch häufig vorkommt wie z.B. die Abfrage aller Eigenschaften einer bestimmten Ressource.

Eben diese beiden Probleme werden durch die Semantik des RDF-Standards umgangen, indem die indirekte Prädikatszuordnung

$$p \mapsto I(p) \mapsto ext(I(p)) \subseteq R \times R$$

durchgeführt wird. Eine weiteres Beispiel für die Anwendung dieser Art Semantik, nämlich im Rahmen der KIF-Standardisierung der Wissensrepräsentation, ist in [HM01] zu finden.

Darüberhinaus macht die RDF-Standardsemantik die im letzten Abschnitt hervorgehobene Validierung des RDFS-Schemas ‚durch sich selber' erst möglich (siehe Bild 2.5; man spricht

[33]Es handelt sich bei der Aussage $(x, y) \in x$ um eine Variante der logischen Antinomie von Russell (auch Mengenparadoxon genannt).

auch von einem zyklischen Metamodell), da die Interpretation des dazu notwendigen axiomatischen Tripels *(rdfs:domain, rdfs:domain, rdf:Property)* unter der Semantik mit direkter Prädikatszuordnung auch zum Mengenparadoxon führen würde.

Statt *rdf:type* direkt der Elementbeziehung $\in$ zuzuordnen, wie es vergleichsweise mit dem Instanzkonstruktor $:$ in der *Description Logic* getan wird, wird die Elementbeziehung bei RDF über die Abbildung $cext$ ausgedrückt (siehe Seite 37). Damit ist es, wiederum das Mengenparadoxons vermeidend, möglich, daß Klassen sich selber als Instanz beinhalten können und zwei verschiedene Klassen die gleiche Extension besitzen. Ist letzteres bei einer Applikation nicht erwünscht, so kann die sehr liberale RDF-Semantik durch die im Standard [Hay04, Abschnitt 4.2] aufgestellten semantischen Bedingungen, genannt *extensional semantic conditions*, einschränkt werden.

Die liberale RDF-Semantik, die unter anderem

- ein zyklisches Metamodell,
- die Möglichkeit zu einer nicht beschränkten Anzahl von Modellhierarchiestufen,
- die im obigen Beispiel motivierte Möglichkeit, daß Prädikatsextensionen das Prädikat selber enthalten dürfen,
- ein nicht leichtes Verständnis der RDFS-Spezifikation im Vergleich zur klassischen Metamodellierung und
- die Möglichkeit zur Nutzung der Vokabularien höhere Hierarchiestufe in allen darunter liegenden

zur Folge haben, wird in dem Vorschlag [PH01] eingeschränkt. Dort wird eine fixe Anzahl von Modellhierarchiestufen vorgeschlagen, angelehnt an die vierstufige Modellarchitektur in UML.

2.2.2 OWL

RDF kann als Ontologie-Basissprache verstanden werden, da bewußt die Generalität im Vordergrund steht und auf eine Vielfalt von die Ausdrucksstärke erhöhenden Konstrukten verzichtet wird. Seine Rolle als Basissprache kommt in der im Rahmen des Standards entwickelten

Architektur des *Semantic Web* (auch *Semantic Web Tower* genannt) in Bild 2.7 zum Ausdruck. Dort ist RDF oberhalb der XML-Schicht angesiedelt. Oberhalb von RDF wird ein erweitertes Ontologie-Vokabularium zur Verfügung gestellt, welches ausdrucksstärkere Konstrukte wie beispielsweise Konstrukte zur Angabe von Kardinalitäten für Relationen umfaßt.

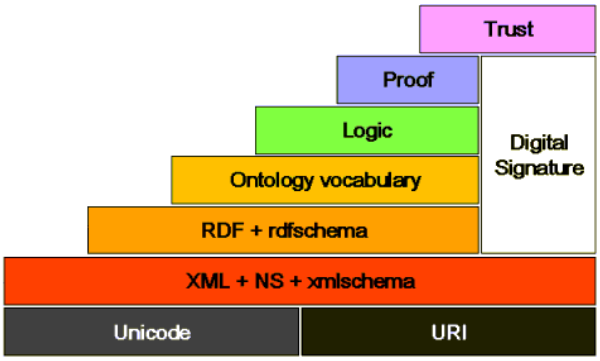

Bild 2.7: *Semantic Web* - Architektur

Die Sprache dieser oberhalb von RDF liegenden Schicht, welche die ausdrucksstärkeren Konstrukte bietet und *Ontology Web Language* (OWL) genannt wird, ist ebenfalls als Standard verabschiedet. Der Standard umfaßt im wesentlichen, aufbauend auf den allgemeinen Anforderungen und Anwendungsfällen in [Hef04], die Sprachreferenz [BvHH+04] und die abstrakte Syntax und Semantik [PSHH04]. OWL kann als direkter Nachfolger von DAML-OIL angesehen werden [Hor02a, Hor02b] und ist damit geprägt von der *Description Logic*-Gemeinde (siehe Abschnitt 2.1.2). Die historische Entwicklung nachvollziehend wird in Abschnitt 2.2.2.1 OWL für sich stehend, also ohne Berücksichtigung der semantischen Integration mit RDF betrachtet. In Abschnitt 2.2.2.2 wird auf diese Integration eingegangen.

2.2.2.1 OWL-Vokabular

Die Möglichkeiten der RDF-Schicht zusammenfassend, unterstützt das RDF-Vokabular und insbesondere RDFS, das die Basiselemente zur Bildung von Schemata anbietet,[34]

- Klassen- und Eigenschaftsdeklaration wie *(Person, type, Class)* oder *(kind, type, Property)*,

[34] Im weiteren wird oft der Namensraum, wenn aus dem Zusammenhang ersichtlich, weggelassen.

- Spezialisierungen wie *(Dozent, subClassOf, Person)* oder *(vater, subPropertyOf, elternteil)*,
- Signaturen wie *(alter, domain, Person)* und *(alter, range, Literal)*,
- Instantiierungen wie *(maria, type, Dozent)* und
- Eigenschaftszuweisungen wie *(maria, alter, 27)*.

Mit dem OWL-Vokabular der Ontologie-Schicht kann man nun, dem DL-Ansatz folgend, Ausdrücke über Klassen bilden, Klassen in Abhängigkeit von Eigenschaftswerten definieren und Eigenschaften als symmetrisch oder transitiv kennzeichnen. Beispiele hierfür, notiert in der Syntax aus Abschnitt 2.1.2, wären:

- ein lehrender Student ist genau der Dozent, der studiert:
  LehrenderStudent $\equiv$ ***and*** *(Student, Dozent)* ,
- die Vorfahren-Relation ist der transitive Abschluss der invertierten Kind-Relation:
  vorfahre $\equiv$ ***transitive*** *(**inverse** (kind))* ,
- ein Achtzehnjähriger ist genau die Person, die achtzehn Jahre alt ist:
  Achtzehnjährig $\equiv$ ***and*** *(Person, **hasValue** (alter, 18))* und
- eine Durchschnittsfamilie hat zwei oder drei Kinder:
  Durchschnittsfamilie $\sqsubseteq$ ***and*** *(**at-least** (2, kind), **at-most** (3, kind))* .

Eine Vorgabe bei der Entwicklung des Standards war es vor allem, das die konkrete Syntax von OWL-Ontologien die RDF-Syntax zu sein hat, was hauptsächlich eine Reifikation der DL-Ausdrücke nach sich zieht. Als Veranschaulichung soll die in Fragment 2.6 exemplarisch dargestellte Übersetzung eines DL-Ausdrucks in einen (doch schwer lesbaren) RDF-Graphen dienen. Dort erfolgt eine Reifikation des ***all***-Konstrukts über eine RDF-Liste. Die Form $\_x$ steht dabei für einen *blank node*. *c* ist hier sowohl eine Instanz von *owl:Class* als auch eine Instanz von *rdf:Class* zur Gewährleistung der Kompatibilität gemäß der in Bild 2.7 angegebenen Schichtung. Der vollständige Satz von Übersetzungsregeln, bezogen auf die abstrakte Syntax aus [PSHH04, Abschnitt 2] ist in [PSHH04, Abschnitt 4] definiert. Die reine Tripel-Syntax, sei es in der *N-Triple*-Notation aus [GB04] oder auch in RDF/XML, ist natürlich nicht zur direkten Eingabe gedacht, weswegen im Zuge der Verbreitung von OWL Werkzeuge angeboten

DL-Ausdruck:

c ≡ ***and*** *(d, e)*

Tripel-Repräsentation:

(c, rdf:type, owl:Class), (c rdf:type rdfs:Class),
(c, owl:intersectionOf, _l1),
(_l1, rdf:first, d), (_l1, rdf:rest, _l2),
(_l2, rdf:first, e), (_l2, rdf:rest, rdf:nil)

Fragment 2.6: Beispiel für die Übersetzung eines DL-Ausdrucks

werden müssen, die gemäß dieser Übersetzungsregeln den RDF-Graphen zum Zwecke des Austausches exportieren.

OWL wurde nicht nur von der DL-Gemeinde beeinflußt, sondern auch geprägt von dem *Frame*-Paradigma, dem Wissensrepräsentationssprachen wie [GEF+99] und [CFF+98] anhängen. Der Grundgedanke, der wiederum als Vorlage für die objektorientierte Modellierung diente, ist die Bündelung von Eigenschaften einer Klasse ausgehend von der Klassendefinitionen selber. In einer puren DL spielt es beispielsweise keine Rolle, ob man zwei Ausdrücke $c \sqsubseteq d$ und $c \sqsubseteq e$ anstatt des einen $c \sqsubseteq$ ***and*** *(d, e)* angibt. Letztere Form wird in einer *Frame*-orientierten DL nahegelegt, wenn nicht erzwungen. Allerdings erfolgt kein Zwang auf sprachlicher Ebene wie bei der objektorientierten Programmiersprache Java, sondern dieser Zwang wird eher über Werkzeuge zur Erstellung von Ontologien wie [GEF+99] oder [BHGS01] ausgeübt. OWL erzwingt dies auch nicht, wenngleich ihre abstrakte Syntax dies unterstützt.

Wie in [HPSvH03] ausgeführt, ist OWL als Nachfolger von DAML-OIL eine Erweiterung der in Abschnitt 2.1.2.1 angegebenen DL-Sprachklasse um den Nominal-Konstruktor und um konkrete Datentypen (siehe Abschnitt 2.1.2.6). Das OWL-Vokabular besteht nun

- aus Konstrukten, die den Klassenkonstruktoren zur Bildung von Klassenausdrücken und den Prädikaten zur Bildung von *terminological axioms* bzw. *assertions* dieser DL direkt entsprechen,
- aus Konstrukten, die Abkürzungen für DL-Ausdrücke darstellen (das Konstrukt ***hasValue*** *(alter,18)* ist beispielsweise eine solche Abkürzung und zwar für den DL-Ausdruck ***all*** *(alter, {18})*) und schließlich

- aus Konstrukten, die dem Austausch und der Verwaltung von Ontologien im Web-Kontext und der Angabe von Annotationen dienen.

OWL-Ausdruck (abstrakte Syntax)	**DL-Ausdruck**
ObjectProperty (p	
super (p_1*) , . . . , super (*p_n*)*	$p \sqsubseteq p_i$
domain (c_1*) , . . . , domain (*c_n*)*	***at-least****(1,*p*)* $\sqsubseteq c_i$
range (d_1*) , . . . , range (*d_m*)*	***top*** $\sqsubseteq$ ***all****(*p*,* d_i*)*
[inverseOf (q*)]*	$p \equiv$ ***inverse****(*q*)*
[symmetric]	***inverse****(*p*)* $\sqsubseteq p$
[transitive]	***transitive****(*p*)* $\sqsubseteq p$
[functional]	***top*** $\sqsubseteq$ ***at-most****(1,* p*)*
[inverseFunctional]	***top*** $\sqsubseteq$ ***at-most****(1,* ***inverse****(*p*))*
)	
restriction (p	
hasValue (v*)*	***all*** *(*p*,* $\{v\}$*)*
)	

Tabelle 2.1: OWL-Abkürzungen für DL-Ausdrücke

Tabelle 2.1 zeigt die Abbildung der wesentlichen Konstrukte der zweiten Art in einen DL-Ausdruck. Konstrukte der ersten Art, welche als Basis dienen, werden nicht aufgeführt, da sie, wie gesagt, direkte Entsprechungen sind. Eine ausführliche Erläuterung dieser Konstrukte und die der dritten Art sind in der Sprachreferenz [BvHH+04] zu finden. In der Tat sind die in Tabelle 2.1 aufgeführten Konstrukte die einzigen, die nicht nur eine syntaktische 1:1-Umformung darstellen, wenngleich die Grenzen fließend sein mögen. Die entsprechenden Konstrukte für die konkreten Datentypen, auf die hier nicht weiter eingegangen wird, werden analog definiert. Eine vollständige Abbildungsvorschrift des in der abstrakten Syntax notierten OWL-Vokabulars nach DL-Ausdrücken ist in [HPSvH03, Abschnitt 7] und auch [Vol04, Abschnitt 3.4] zu finden, welche auf der für OIL angegeben Regeln in [DFvH+00] basiert.

Besondere Beachtung verdienen die Konstrukte *domain* und *range* aus Tabelle 2.1. Sie stellen eine direkte Übernahme der entsprechenden RDFS-Signatur-Konstrukte dar. Die rechte Spalte kann sozusagen als Definition der Semantik dieser Konstrukte aus DL-Sicht angesehen werden. Im Gegensatz zu [HPSvH03] wird hier der DL-Ausdruck ***inverse****(*p*)* $\sqsubseteq p$ anstatt ***inverse****(*p*)* $\equiv p$ zur Definition eines symmetrischen Prädikats verwendet, da er mehr der Intuition entspricht.[35] Der OWL-Ausdruck p *hasValue(*v*)* soll für die Menge aller Individuen stehen,

dessen Wert für die Property p gleich v ist, was im DL-Ausdruck indirekt mit Hilfe des Nominal-Konstruktors ausgedrückt wird.[36]

2.2.2.2 RDF-Kompatibilität und Semantik

OWL stellt für sich genommen eine vollwertige DL dar, dem mit der Definition einer eigenen modelltheoretischen Semantik in [PSHH04, Abschnitt 3] Rechnung getragen wird. Sie entspricht weitestgehend der üblichen Semantik-Definition für eine DL, erweitert um die RDF-bezogenen Konstrukte wie Annotationen und konkreten Datentypen auf XML-Schema-Basis.

Das Ziel bei der Entwicklung des Standards war aber die Passbarkeit in die Ontologie-Schicht der *Semantic Web*-Architektur (siehe Bild 2.7). Diese Schichtung erfordert eine syntaktische und semantische ‚Kompabilität', welche letztendlich vom Standard eingehalten wurde. Auf dem Weg zu dieser Kompatibilität lagen einige Probleme, die in [PSF02, HPSvH03] detailliert erläutert werden. Sie entstanden letztendlich durch die Vorgaben, daß OWL in der RDF/XML-Syntax notiert werden muß und daß die OWL-Semantik als Erweiterung von RDFS zu definieren war, genauso wie RDFS eine Erweiterung von RDF ist (siehe Abschnitt 2.2.1.5).

Eines der syntaktischen Probleme ist die Parsierung von RDF-Graphen, welche OWL-Ontologien repräsentieren sollen. Denn auch die Angabe eines RDF-Schemas für OWL kann es nicht verhindern, daß RDF-Graphen aus OWL-Sicht keinen Sinn machen. Dies wäre der Fall, wenn beispielsweise die letzten beiden Zeilen aus Fragment 2.6 fehlen würden, also nur ein unvollständiger DL-Ausdruck vorhanden wäre. Eine Lösung dieses Problems ist die Festschreibung der Parsierung wie in [Bec04a]. Die Serialisierung zu RDF-Graphen hingegen ist leicht durch eine direkte Anwendung der Regeln aus [PSHH04, Abschnitt 4] zu realisieren.

Die semantischen Probleme liegen darin begründet, daß die Semantik für das OWL-Vokabular auf der Semantik des RDFS-Vokabulars aufbauen soll. Dies schließt ein, daß die RDFS-Semantik der wohlbekannten Konstrukte zur Instantiierung und Spezialisierung in die OWL-Semantik einfließen muß. So wie die semantischen Bedingungen für RDFS auf Seite 37 unter Abschnitt 2.2.1.5 den Raum der möglichen Modelle einschränken, müssen die semantischen Bedingungen für das ausdrucksstärkere OWL-Vokabular den Modellraum einschränken. Eine detaillierte Auflistung ist in [PSHH04, Abschnitt 5] zu finden.

[35] Man beachte, daß die Äquivalenz der beiden Ausdrücke direkt aus $R^- \subseteq R \Rightarrow R \subseteq R^-$ für eine beliebige Relation R folgt. Zumindest für das *transitive*-Konstrukt verwendet [Vol04] auch die Inklusion.

[36] Die Nutzung der Äquivalenz $x = y \Leftrightarrow x \in \{y\}$ mag verständlicherweise für denjenigen, der es aus Sicht einer klassischen Programmiersprache betrachtet, umständlich erscheinen, zeigt aber zugleich eine gewisse Orthogonalität der DL.

2.2.2.3 OWL- Subsprachen

OWL als direkte Erweiterung von RDF ermöglicht aus syntaktischer Sicht die beliebige Kombinierbarkeit des RDF(S)- und OWL-Vokabulars. Diese uneingeschränkte Sprachausprägung wird *OWL-Full* genannt. OWL-Full hat, wenn man es aus der DL-Sicht betrachtet, zwei Nachteile. Zum einen gibt es an manchen Stellen unintuitive Effekte bezüglich der erwarteten logischen Folgerungen [HPSvH03, Abschnitt 6] und zum anderen ist diese Sprache nicht entscheidbar wegen der Kombinierbarkeit von Kardinalitätsangaben mit der Transitivitätskennzeichnung für Prädikate. Um diese beiden Probleme zu vermeiden, ist im Standard zusätzlich die Sprachausprägung *OWL-DL* definiert, welche die Kopplung der beiden Vokabularien derart einschränkt, daß OWL wiederum (so wie im Abschnitt 2.2.2.1) als eine vollwertige DL angesehen werden kann, mit der entsprechenden Semantik und abstrakten Syntax aus [PSHH04, Abschnitt 3].[37] OWL-DL erlaubt die Verwendung von bestehenden Werkzeugen gemäß den Zielen des Standards, verdeutlicht durch das Zitat *„OWL DL was designed to support the existing Description Logic business segment and to provide a language subset that has desirable computational properties for reasoning systems“* [BvHH$^+$04, Abschnitt 1.2].

Daher gibt es im wesentlichen die folgenden Beschränkungen für OWL-DL-Ontologien (vergl. [BvHH$^+$04, Abschnitt 8] und [Vol04, Abschnitt 3.4]).

- Die Namensmengen für Klassen, Prädikate und Individuen sind paarweise disjunkt.
- Das RDF(S)- und OWL-Vokabular ist nicht in den Namensmengen für Klassen, Prädikate und Individuen enthalten.
- Axiome, d.h. *terminological axioms* und *assertions*, haben keine zyklische Form (wie z.b. $(x,$ owl:allValuesFrom, $x)$).
- Für jedes transitive Prädikat sowie dessen Invertierungen und Spezialisierungen ist keine Kardinalitätsangabe vorhanden.

Die ersten beiden Bedingungen spiegeln die Definition von OWL-DL-Interpretationen in [PSHH04, Abschnitt 5] wieder. Aus der Ersten folgt insbesondere, daß Individuen keine Klassen darstellen können, während aus der Zweiten folgt, daß sowohl das RDF(S)- als auch das OWL-Vokabular nicht durch eine OWL-DL-Ontologie verändert werden kann (beispielsweise durch

[37] OWL-DL kann als syntaktische Variante der DL-Sprachklasse $\mathcal{SHOIN}(\mathbf{D})$ angesehen werden [HPSvH03].

das Statement *(owl:cardinality, rdfs:domain, rdfs:Class)*). Die dritte Bedingung soll unter anderem die in [HPSvH03, Abschnitt 6] aufgeführten unintuitiven Effekte bezüglich erwarteter logischen Folgerungen verhindern. Die letzte Bedingung gewährleistet, wie oben erwähnt, die Entscheidbarkeit von OWL-DL.

Eine weitere Subsprache ist *OWL-Lite*. Sie verbietet hauptsächlich den Gebrauch der OWL-Konstrukte

oneOf, unionOf, complementOf, hasValue, disjointWith und *DataRange* .

Ferner erzwingt sie an vielen Stellen die Angabe von Klassennamen statt der sonst möglichen Klassenausdrücke und läßt als Kardinalität nur 0 und 1 zu. Eine vollständige Liste der Einschränkungen ist in [BvHH+04, Abschnitt 8] zu finden. OWL-Lite ist primär für diejenigen gedacht, die herkömmliche Klassifikationshierarchien repräsentieren wollen, ohne dabei fortgeschrittenen Gebrauch von der DL-Ausdrucksstärke zu machen. Allerdings sind einige Einschränkungen mehr als Anwendungsregeln zu betrachten, da mit [Vol04, Korrolar 3.4.1] die Ausdrückbarkeit zumindest für

unionOf, complementOf und *disjointWith*

gefolgert werden kann. Aus Sicht der klassischen Informationsmodellierung ist man mit OWL-Lite in der Lage, Klassenhierarchien und den Schnitt von Klassen zu bilden, Assoziationen zwischen Klassen und Assoziationshierarchien zu definieren sowie Klassen mit Hilfe der zwei DL-Basis-Restriktionen auf Assoziationen einschließlich Kardinalitätsangabe zu bilden.

OWL-DL und OWL-Lite haben gegenüber F-Logic und OWL-Full bzw. RDF den entscheidenden Nachteil, das Klassen nicht als *first class citizens* existieren. Das bedeutet, daß Klassen nicht selber klassifiziert und in der Rolle einer Instanz einer (Meta-)Klasse angesehen werden können sowie keine Eigenschaftszuweisungen besitzen. In manchen Anwendungen ist diese Eigenschaft jedoch erforderlich, wie beispielsweise in [SM01, Seite 5] ausgeführt wird: „*... one may ask for questions like 'show me the concept taxonomy including only those concepts for which you have some news in the last week' ...*“. Diese Eigenschaft macht ist leider auch nicht besonders geeignet für die Repräsentation und die Verwaltung von Metamodellen, wie im Standard [BvHH+04, Abschnitt 1.2] selber erwähnt: „*The complete OWL language (called OWL Full to distinguish it from the subsets) relaxes some of the constraints on OWL DL so as to make available features which may be of use to many database and knowledge representation systems, but which violate the constraints of Description Logic reasoners*“ und „*OWL*

Full will typically be useful for people who want to combine the expressivity of OWL with the flexibility and metamodeling features of RDF“[BvHH+04, Abschnitt 8.1].

2.2.3 TRIPLE

TRIPLE wurde von M. Sintek und S. Decker erstmals im Jahr 2001 vorgestellt [SD01] und baut auf den Konzepten aus [DBSA98] auf. Der Entwurf dieser logischen Regel-, Anfrage- und Transformationssprache orientierte sich an den folgenden Zielen.

- Unterstützung der RDF-Konzepte zur Repräsentation von Ressourcen, Properties und Beziehungssetzungen.
- Abfragbarkeit von RDF-Instanzen und RDF-Schemata in einheitlicher Art und Weise, d.h. die Unterstützung der Behandlung von RDF-Schema-Elementen als *first class citizens*.

Weitere Ziele, die über die bloße Repräsentierbarkeit von RDF(S)-Graphen hinausgingen und nicht direkt mit Standardsystemen wie relationale Datenbanken oder Dokumentmanagement-Systemen aufgrund der fehlenden Inferenzkomponenten umgesetzt werden können, betreffen die RDF(S)-Semantik, nämlich die Einbeziehung der semantischen Bedingungen aus [BG04] (siehe Abschnitt 2.2.1.5)

- für Anfragen und nutzerdefinierte Regeln und
- für die Validierung von RDF-Graphen bezüglich RDFS-Graphen (siehe auch Abschnitt 2.2.1.7).

Aufgrund der Ähnlichkeit der Grundstrukturen von F-Logic und RDF (vergl. [DBSA98]) wurde im wesentlichen F-Logic als Vorbild für TRIPLE herangezogen. Der Grund dafür liegt in dem gemeinsamen Ausgangspunkt der frame-basierten Sprachen mit dem zentralen Konstrukt des Tripels *(subject, predicate, object)*.[38] Nicht nur syntaktische Ähnlichkeiten favorisieren F-Logic als Vorbild, sondern auch die Ähnlichkeiten in der Semantik, weshalb die F-Logic-Semantik als ein Kandidat zur Definition der RDF-Semantik vorgeschlagen wurde [YK02].

[38] Oft verwendete alternative Begriffe für *predicate* sind z.B. *property*, *feature*, *slot* und *attribute*. Mit diesem Konstrukt ist bei allen in dieser Arbeit behandelten Sprachen die binäre Relation *predicate*(*subject*, *object*) beziehungsweise in reifizierter Form die ternäre Relation *r(subject, predicate, object)* intendiert.

Eines der wesentlichen Merkmale von TRIPLE ist es jedoch, wie oben als Ziel formuliert, die RDFS-Semantik nicht explizit zu übernehmen, sondern diese über eine spezielle Regelmenge einzubinden. Im folgenden wird TRIPLE vorgestellt, und zwar auf der Grundlage von [DSN02]. Der Grund für diese Wahl ist der höhere Formalisierungsgrad, wobei allerdings die dort gegebenen Definitionen teilweise von denen in [SD01, SD02] (semiformal) gegebenen abweichen. Beispielsweise werden keine Prädikate zugelassen, sondern Formeln auf *(subject, predicate, object)*-Formeln beschränkt.[39]

2.2.3.1 Basisbausteine

Die Basisbausteine von TRIPLE sind im wesentlichen an denen von F-Logic orientiert (siehe Abschnitt 2.1.1.1). TRIPLE kann als eine Erweiterung einer Untermenge von F-Logic um RDF-spezifische Konstrukte und Konstrukte zur Kontextbildung für Regeln angesehen werden. Das Alphabet ist durch eine Menge von *Resource-Konstruktoren* $\mathbf{F}$, einer unendlichen Menge von *Variablen* $\mathbf{V}$, einer unendlichen Menge von Zeichenketten $\mathbf{H}$, den Symbolen

- *:*, *@* und *->*, einschließlich den gängigen Klammer- und Interpunktionssymbolen,
- den Kontextoperatoren ***intersect***, ***union*** und ***diff*** sowie den logischen *Konnektoren* ***and***, ***or***, ***not***, $\leftarrow$ und *Quantoren* ***forall*** und ***exists***

gegeben.[40] Der Unterschied zu F-Logic auf dieser Ebene besteht im wesentlichen darin, daß Symbole zur Signaturbildung und Mengenkonstruktion wegfallen und eine eigene Menge für Literale $\mathbf{H}$ eingeführt wird, die in F-Logic als Konstanten in $\mathbf{F}$ zur Verfügung stehen. Neben der üblichen prädikatenlogischen Termbildung stehen in TRIPLE zwei RDF-spezifische Konstrukte zur Verfügung:

- Sind n, l Terme, so ist $n:l$ ein Term, genannt *Ressource-id*. n resp. l wird, RDF entsprechend, *lokaler Name* resp. *Namensraum* genannt.
- Sind s, p, o Terme, so ist *s [p -> o]* ein Term, genannt *statement-id*.

[39] Zur Erläuterung von TRIPLE ist diese Beschränkung unerheblich.

[40] Aufgrund der erwähnten Ähnlichkeit werden dieselben Bezeichner für die Alphabetsmengen verwendet. Auf eine Einbeziehung von RDF/XML-Spezifika wie *language identifiers* etc. wird hier verzichtet.

2.2.3.2 Moleküle und Formeln

Im Rahmen von RDF bilden eine Menge von Statements einen RDF-Graphen. Diese wiederum stellen Einheiten dar, welche in weiteren Operationen wie beispielsweise der Vereinigung als Argumente vorkommen. TRIPLE trägt dieser Tatsache durch die Einführung von speziellen Konstrukten Rechnung, den sogenannten *Kontextausdrücken*, welche die Zuordbarkeit von Statements zu diesen Einheiten zulassen:

- Ein Term ist ein Kontextausdruck.
- Sind c_1, c_2 Kontextausdrücke, so auch *(*c_1 ***intersect*** c_2*)* , *(*c_1 ***union*** c_2*)* und *(*c_1 ***diff*** c_2*)* .
- Ist s eine Statement-id und c ein Kontextausdruck, so bezeichnet man s ***@*** c als *T-Molekül*.

Die Bildung von *T-Formeln* ausgehend von dem Molekül mit Hilfe der Konnektoren und Quantoren geschieht wie üblich:

- *T-Moleküle* sind T-Formeln.
- Sind G und H T-Formeln, so auch $(G \leftarrow H)$, $(G$ ***and*** $H)$, $(G$ ***or*** $H)$ und $($***not*** $G)$.
- Ist $v \in \mathbf{V}$ und G eine T-Formel, so auch $($***exists*** $v\, G)$ und $($***forall*** $v\, G)$.

Bemerkung 2.1. (syntaktische Abkürzungen/ Konventionen) Es steht

- s *[*p_1 *->* o_1*; . . . ;* p_n *->* o_n*]* für *(* s *[*p_1 *->* o_1*]* ***and*** $\ldots$ ***and*** s *[*p_n *->* o_n*]* *)* ,
- s *[*p_1 *->* o_1 *[*p_2 *->* o_2*]]* für *(* s *[*p_1 *->* o_1*]* ***and*** o_1 *[*p_2 *->* o_2*]* *)* ,
- s *[*p *->* *{*o_1*,. . .,*o_n*}]* für *(* s *[*p *->* o_1*]* ***and*** $\ldots$ ***and*** s *[*p *->* o_n*]* *)* und
- *(*X_1 ***and*** $\ldots$ ***and*** X_n*)* ***@*** c für *(*X_1 ***@*** c ***and*** $\ldots$ ***and*** X_n ***@*** c*)* .

Diese Transformation wird *flatten* genannt. Die Form

forall $x_1, \ldots, x_n$ *@* C *{*
$F_1.$ $\ldots$ $F_m.$
}

wird *Kontextblock* genannt, wobei C ein Kontextausdruck ist mit Variablen nur aus $x_1, \ldots, x_n$ ($n \geq 0$), die in keinem F_i gebunden vorkommen dürfen (ist $n = 0$, so schreibt man nur ***@ C***). F_i ist eine T-Formel derart, daß sie nach Anwendung von *flatten* Moleküle enthalten darf, denen der Kontextausdruck fehlt, also in denen lediglich eine Statement-id angegeben ist. Der obige Block steht dann als Abkürzung für

forall $x_1, \ldots, x_n$ ${F_1}'.$ $\ldots$ ***forall*** $x_1, \ldots, x_n$ ${F_m}'.$

wobei ${F_i}'$ die T-Formel bezeichnet, die durch Einsetzen von C in die fehlenden Kontext-Stellen aus F_i nach Anwendung von *flatten* entsteht.

```
@ bsp : instanceData {

    bsp : klaus [ rdf : type -> bsp : Person;
                  bsp : hobby -> bsp : fußball ].
    ...
}

@ bsp : schema {

    bsp : hobby [ rdf : type -> rdfs : Property;
                  rdfs : domain -> bsp : Person;
                  rdfs : range -> bsp : Bereich ].
    ...
}
```

Fragment 2.7: TRIPLE Beispiel

Der Vollständigkeit wegen ist in Fragment 2.7 die TRIPLE-Form von Teilen aus den RDF-Graphen in Bild 2.2 und 2.3 auf den Seiten 31, 33 dargestellt.

2.2.3.3 Parametrisierbare Kontexte

Eines der wesentlichen Konzepte in TRIPLE ist das des *parametrisierbaren Kontextes*. Der Begriff parametrisierbarer Kontext steht für die Verwendung von Kontextblöcken, die durch einen Variablen beinhaltenden Kontextausdruck definiert sind. [41]

```
forall x @ flat2(x) {

    forall s, p, o
        s [ p -> o ] ←
            s [ p -> o ] @ x  or  exists y  s [ p -> y [ p -> o ] ] @ x .
}
```

Fragment 2.8: Parametrisierbarer Kontext

Fragment 2.8 zeigt solch einen Kontextblock. Die in ihm angegebene Regel drückt aus, dass alle ein- oder zweistufigen *p*-Beziehungen zwischen Subjekt *s* und Objekt *o*, die im Kontext *x* gelten, in dem Kontext *flat2(x)* als einstufige vorhanden sind, wobei *x* der Parameter des Kontextblocks ist. Gemäß der Abkürzungsvorschrift auf Seite 54 ist dieser Block äquivalent zu einer einzigen Regel, nämlich zu der im Block enthaltenen Regel erweitert um die Variable *x* im forall-Quantorteil und versehen mit dem Kontextausdruck *flat2(x)* im Kopf-Statement.

Will man beispielsweise alle *p*-Beziehungen, die im Kontext *k* gelten, wie eben beschrieben transformieren, so stellt man die Anfrage[42]

```
forall s, p, o  ←  s [ p -> o ] @ flat(k) .
```

Die Lösungsmenge ist nun ein RDF-Graph im Kontext *flat(k)*, der als Transformationsergebnis gemäß dem obigen Kontextblock anzusehen ist. Eine Komposition von n Transformationen bedarf nicht eines zusätzlichen Konstrukts sondern kann direkt über Kontextausdrücke der Form $trans_1(trans_2(\ldots trans_n(\ldots)\ldots))$ realisiert werden.

Die Transformation von RDF-Modellen, unter anderem in [SD02] beschrieben, stellt eine der Hauptanwendungen des parametrisierbaren Kontextes dar. Eine Transformation von Model-

[41] Ein parametrisierbarer Kontext wird oft auch *parameterized view* genannt. Die in der Abkürzungsvorschrift auf Seite 54 erzwungene Form des Kontextausdrucks (Allquantifizierung aller Variablen) läßt, wie in der Literatur bemerkt, eine Interpretation des äußersten Funktors als Skolemfunktion zu.

[42] Wie üblich sind Anfragen Regeln mit leerer Konklusion.

len anderer Sprachen wie z.B. UML kann indirekt über sie repräsentierende RDF-Modelle vorgenommen werden, was in [BBLS04] näher erläutert wird.

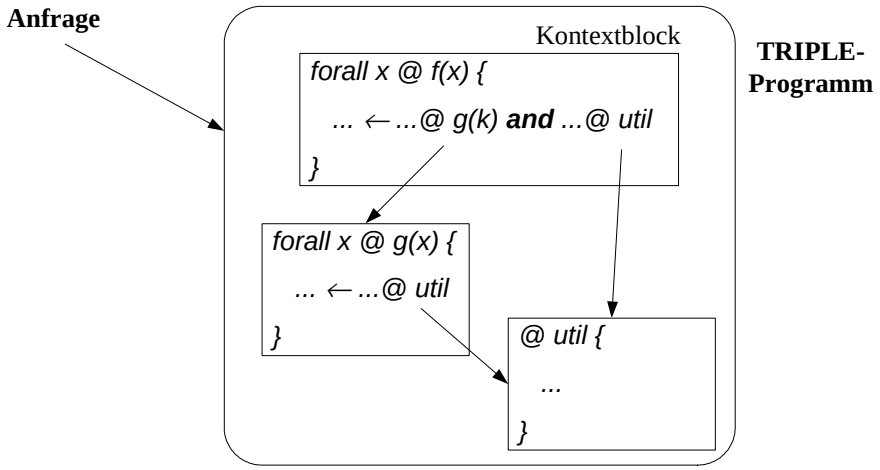

Bild 2.8: TRIPEL-Kontextblöcke

Eine weitere Anwendung von parametrisierbaren Kontexten ist die Modularisierung von Formelmengen. Bild 2.8 zeigt eine Hierarchie von Kontextblöcken. Der oberste Kontextblock nutzt den mittleren parametrisierbaren Block durch die Belegung des Parameters mit einer Konstante *k* sowie den nicht-parametrisierbaren Block *util*. Weitere Anwendungsfälle für Kontexte und entsprechende Erweiterungen von TRIPLE sind in [DSB+05] zu finden.

2.2.3.4 Axiomatische Semantik

In den ersten Papieren [SD01, SD02] wurde TRIPLE als eine auf RDF aufbauende logische Sprache vorgestellt. Erst mit [DSN02] wurde eine modelltheoretische Semantik erarbeitet, allerdings konkurrierend und abweichend zu der RDF Semantik [Hay04]. Aus diesem Grunde und da der Fokus der vorliegenden Arbeit auf der axiomatischen Semantik liegt, wird auf die modelltheoretische Semantik hier nicht näher eingegangen, sondern die Besonderheiten von TRIPLE an Hand der axiomatischen Semantik behandelt.

Die axiomatische Semantik eines TRIPLE-Programs P ist induktiv definiert durch eine (hier mit μ bezeichnete) Transformation, welche P auf prädikatenlogische Formeln abbildet. Die in den TRIPLE-Papieren aufgeführte Transformation, dort allerdings nach Hornlogik, kann wie

folgt präzisiert werden.[43]

(1) (i) Für Terme, die keine Ressource- oder statement-ids sind, ist μ die eineindeutige Zuordnung zu prädikatenlogischen Termen.

(ii) $n : l\ ^{\mu} = resource(n^{\mu}, l^{\mu})$ für Terme n, l.

(iii) $s\ [p \text{ -> } o]\ ^{\mu} = statement(s^{\mu}, p^{\mu}, o^{\mu})$ für Terme s, p, o.

(iv) $< s >\ ^{\mu} = s^{\mu}$ für statement-ids s.

(v) $(s\ @\ (c_1\ \textbf{intersect}\ c_2)\ )^{\mu} = (s@c_1)^{\mu} \wedge (s@c_2)^{\mu}$
$(s\ @\ (c_1\ \textbf{union}\ c_2)\ )^{\mu} = (s@c_1)^{\mu} \vee (s@c_2)^{\mu}$
$(s\ @\ (c_1\ \textbf{diff}\ c_2)\ )^{\mu} = (s@c_1)^{\mu} \wedge \neg\ (s@c_2)^{\mu}$
$(s\ @\ c)\ ^{\mu} = true(s^{\mu}, c^{\mu})$
für statement-ids s, Kontextausdrücke c_1, c_2 und Term c.

(vi) Für T-Formeln (außer T-Molekülen) ist μ die Fortsetzung auf ihre Argumente.

Man beachte, daß Transformationsvorschrift (1.iii) analog zur axiomatischen Semantik von RDF unter Abschnitt 2.2.1.7 auf Seite 39 definiert ist. Die Besonderheiten von TRIPLE sind vor allem ihre mengen-algebraischen Operationen für Kontexte und ihre Transformation definiert durch (1.v). Für die RDF *blank nodes* (siehe Abschnitt 2.2.1.4) gibt es in TRIPLE keine speziellen Konstrukte, da sie als existenz-quantifizierte Variablen angegeben werden können. Abweichend zum RDF-Standard werden Ressourcen nicht als atomar behandelt. Außerdem weicht die Reifikation von statements vom RDF-Standard ab: sie wird sozusagen durch bloße Subtermbildung ersetzt. Die Aussage *klaus [glaubt -> < exkurs10 [fuehrung -> maria] >]* steht in TRIPLE nämlich für das Faktum

$$statement(klaus, glaubt, statement(exkurs, fuehrung, maria)).$$

Eine wesentliche in [DSN02] vorgestellte Unterklasse von TRIPLE wird Horn-TRIPLE genannt, welche zunächst die Bildung von T-Formeln (siehe Abschnitt 2.2.3.2) auf Hornformeln

[43] Der besseren Lesbarkeit wegen wird $\mu(x)$ mit x^{μ} notiert. Es sei bemerkt, daß im Gegensatz zu der in [SD01] gegebenen originalen Definition ;-Ausdrücke hier fehlen, da sie als Abkürzungen (siehe Seite 54) verstanden werden. Man beachte, daß die mit der Anwendung von μ eingeführten Prädikate (z.B. $statement$) spezielle nicht in P vorkommende Prädikate sind.

einschränkt. Dies ist jedoch nicht ausreichend, da die Semantik von Kontextausdrücken einer Disjunktion beziehungsweise Negation von atomaren Formeln gemäß (1. v) entsprechen kann. Daher wird der Kopf von Horn-TRIPLE-Regeln derart eingeschränkt, daß der Kopf nur Terme oder ***intersect***-Ausdrücke und daß der Rumpf nur Terme, ***intersect***-Ausdrücke oder ***union***-Ausdrücke enthalten darf. ***diff***-Ausdrücke sind aufgrund der erforderlichen Negation generell nicht zugelassen.[44]

Eine Möglichkeit, die Ausdrucksstärke zu erhöhen, ist durch die Zulassung von beliebigen prädikatenlogischen Ausdrücken in dem Rumpf einer Regel gegeben. Die Regeln werden dann mit Hilfe der Lloyd-Topor-Transformation in Hornformeln mit Negation (*negation as failure*) im Rumpf transformiert [Llo87].[45] Darüberhinaus würde die Zulassung von Negation eine Verwendung des ***diff***-Konstrukts im Rumpf der Regeln erlauben.

[44] Die Zulassung von ***intersect***-Ausdrücken im Kopf und ***union***-Ausdrücken im Rumpf sind durch die logische Äquivalenz $(a \wedge b \leftarrow c \vee d) \Leftrightarrow ((a \leftarrow c) \wedge (a \leftarrow d) \wedge (b \leftarrow c) \wedge (b \leftarrow d))$ begründet.

[45] In [Llo87] werden Hornformeln mit prädikatenlog. Ausdrücken im Rumpf als *(extended) programs* und Hornformeln mit Negation im Rumpf als *normal programs* bezeichnet. Eine Optimierung der Transformation bezüglich der Disjunktionsregel ist in [Dec02, Abschnitt 7.3.2] aufgeführt.

2.3 DL und Regeln

Da *terminological axioms* und *assertions* aus Sicht der Prädikatenlogik lediglich als atomare Formeln der speziellen Prädikate $\sqsubseteq$ und *:* angesehen werden können, liegt es nahe, darüber beliebige logische Formeln zuzulassen. Denn wie in [HAMS05, Hor05] erläutert, erhöht die Integration von DL und einer Regelsprache die Ausdrucksstärke erheblich, was das folgende Beispiel, angelehnt an die Beispielregel aus [HPS04], verdeutlichen soll.

> ***forall*** *a,b,c (a, c) : onkelVon* $\leftarrow$ *(a, b) : elternteilVon* ***and*** *(b, c) : bruderVon.*
> *onkelVon* $\sqsubseteq$ *verwandtMit.*

Damit können *role assertions* nicht nur über Fakten, sondern in Abhängigkeit von anderen *role assertions* definiert werden. Ohne den Einsatz von Regeln, also allein in OWL-DL, sind diese Abhängigkeiten nicht ausdrückbar.[46]

Bezüglich der Integration von DL und einer Regelsprache lassen sich zwei Aspekte hervorheben, nämlich die

- Integrationsstärke und
- Mächtigkeit der beiden Komponenten.

Mit Integrationsstärke ist der Umfang der Schnittstelle zwischen der DL und der Regelsprache gemeint, das heißt, inwieweit die eine Komponente auf die Elemente der anderen zugreifen kann. Mit dem Aspekt der Mächtigkeit der beiden Komponenten ist gemeint, inwieweit ein Ansatz zur Integration den Umfang der Konstrukte der einzelnen Komponenten einschränkt.

Bild 2.9 zeigt einen wechselseitigen Zugriff a) und b) zwischen den Komponenten, wie es in dem Beispiel erforderlich ist. Der Zugriff a) erfolgt im Rumpf der Regel auf die in DL definierten Fakten für die Relationen *elternteilVon* und *bruderVon*. Die im DL-Teil definierte *verwandtMit*-Relation wiederum referenziert über b) die *onkelVon*-Relation.

Im folgenden werden einige Ansätze zur Integration von DL und Regeln in Hinblick auf die oben erwähnten Aspekte vorgestellt. Zu diesem Zwecke seien atomare Formeln, die keine Funktionssymbole und keine Klassen-oder Rollennamen der DL-TBox enthalten, *non-DL-Atome* genannt. Als *DL-Atome* seien Atome der Form

[46] Es sei aber bemerkt, daß mit einer DL, die allgemeine Rollenkompositionen der Form $P \circ Q \sqsubseteq R$ erlaubt, das Beispiel ausdrückbar ist. Die Erweiterung führt jedoch zur Nicht-Entscheidbarkeit der DL [HPS04].

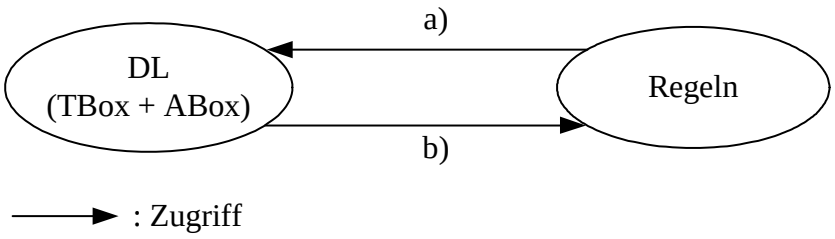

Bild 2.9: DL und Regeln

$c(t), r(t_1, t_2)$,

also ein- oder zweistellige Atome, bezeichnet, die direkt den Ausdrücken

$t : c$, $(t_1, t_2) : r$

der vorangegangenen Abschnitte entsprechen, wobei t, t_i Variablen oder Konstanten sind und c, r ein Klassen- resp. Rollenname bezeichnet (siehe *concept assertion* resp. *role assertion* in Abschnitt 2.1.2.3). Die allgemeine Form von Regeln in den hier behandelten Ansätzen ist

$$A_0 \leftarrow A_1, \ldots, A_n, B_1, \ldots, B_m \ , \qquad (*)$$

wobei $A_0, \ldots, A_n$ non-DL-Atome und $B_1, \ldots, B_m$ DL-Atome sind. A_0 kann entweder DL-Atom oder ein non-DL-Atom sein ($n, m \geq 0$).[47] DL-Atome stellen die Verbindung zu der DL-Komponente dar. Sie können als Anfragen an die DL-Komponente interpretiert werden (vergl. *instance checking* und *retrieval problem* in Abschnitt 2.1.2.5 auf Seite 28).

Schon vor der OWL-Standardisierung gab es zwei entscheidene Ansätze zur Integration [DLNS98, LR96]. In AL-Log wurde eine Kombination mit Datalog-Regeln favorisiert, jedoch mit der Einschränkung, daß für B_i nur die Form $c(t)$ und für A_0 nur non-DL-Atome zugelassen sind und daß, falls t eine Variable ist, sie in mindestens einem A_i vorkommen muß (AL-Log-*safety*). Zusätzlich gilt die *safety*-Bedingung von Datalog, die besagt, daß jede im Regelkopf vorkommende Variable auch im Regelrumpf vorkommen muß. Während die Mächtigkeit der Regelsprache auf Datalog beschränkt ist, ist die Mächtigkeit der DL-Komponente bei AL-Log

[47] O.b.d.A. stehen zuerst die DL-Atome.

beschränkt auf die DL $\mathcal{ALC}$, welche äquivalent ist zu der in Abschnitt 2.1.2.1 definierten DL verringert um *number restrictions*, *inverse roles* und *transitive roles*, also unterhalb von OWL-DL liegt. Ferner ist kein wechselseitiger Zugriff erlaubt, sondern nur der von der Regel- auf die DL-Komponente (siehe a in Bild 2.9).

Die in [LR96] vorgestellte Integration CARIN kann als Erweiterung von AL-Log angesehen werden und zwar derart, daß sowohl die Mächtigkeit der gewählten DL-Klasse als auch die der Regelsprache erhöht wurde. Die zugrunde gelegte DL-Klasse ist $\mathcal{ALCNR}$, welche äquivalent ist zu der in Abschnitt 2.1.2.1 definierten DL erweitert um Rollenkonjunktionen und verringert um *inverse roles* und *transitive roles*. CARIN läßt innerhalb der Regelrümpfe beliebige DL-Atome zu.[48] Im Gegensatz zu AL-Log ist die Beschränkung aufgehoben, daß die Variablen der DL-Atome in $A_1, \ldots, A_n$ vorkommen müssen. Jedoch muß, wie bei AL-Log, A_0 ein non-DL-Atom sein, weshalb kein wechselseitiger Zugriff zwischen der DL- und Regelkomponente möglich ist, sondern die Regelkomponente oberhalb der DL-Komponente angesiedelt ist.

Die beiden eben vorgestellten Ansätze waren der Ausgangspunkt für die Arbeiten zur Integration von OWL-DL und Regeln [HPS04, MSS04]. Diesen Arbeiten ist die OWL-DL-Mächtigkeit und die Aufhebung der einseitigen Zugriffsbeziehung a) aus Bild 2.9 gemein, das heißt, daß sowohl Regeln ‚oberhalb' von DL als auch umgekehrt definiert werden können.

Der Ansatz [HPS04] stellt die OWL-DL-Regelsprache ORL vor, allerdings mit der Einschränkung der obigen Form $(*)$, daß keine non-DL-Atome in der Regel zugelassen sind. Daher muß A_0 ein DL-Atom sein, was die volle Integration von DL und Regeln ermöglicht. Der Grund für die Beschränkung dürfte sein, daß damit eine Erweiterung der OWL-Semantik um beliebige Relationen umgangen wird – man beschränkt sich auf die monadischen Typprädikate und binären Relationen gemäß dem OWL-Vokabular. Wie bei CARIN dürfen $B_1, \ldots, B_n$ beliebige DL-Atome sein. Neben der Einbettung der Regelsprache in die OWL-Semantik wurde in [HPS04] ihre syntaktische Einbettung in die konkrete XML-Syntax von OWL vorgenommen.

Ein weiterer Ansatz für eine OWL-DL-Regelsprache ist in [MSS04] vorgestellt worden. Er erweitert den ORL-Ansatz derart, daß non-DL-Atome im Kopf und Rumpf der Regel wieder zugelassen sind, und daß ein Gleichheitsprädikat von der Sprache direkt zur Verfügung gestellt wird. Um jedoch den Nachteil von ORL, nämlich ihre Nicht-Entscheidbarkeit, zu beheben, beschränkt man sich auf Regeln, die *DL-safe* sind: die Regel der Form $(*)$ ist *DL-safe*, falls alle ihre Variablen in $A_1, \ldots, A_n$ vorkommen.[49]

[48] Ferner sei bemerkt, daß hier nur auf *rekursives CARIN*, also nicht auf die einfachere Variante, welche Regelrekursion verbietet, eingegangen wird.

[49] Die Entscheidbarkeit von AL-Log basiert auch auf dieser Bedingung. Es sei bemerkt, daß die Entscheidbarkeit von CARIN über die sogenannte *role-safety* gewährleistet wird, die fordert, daß mindestens eine Variable

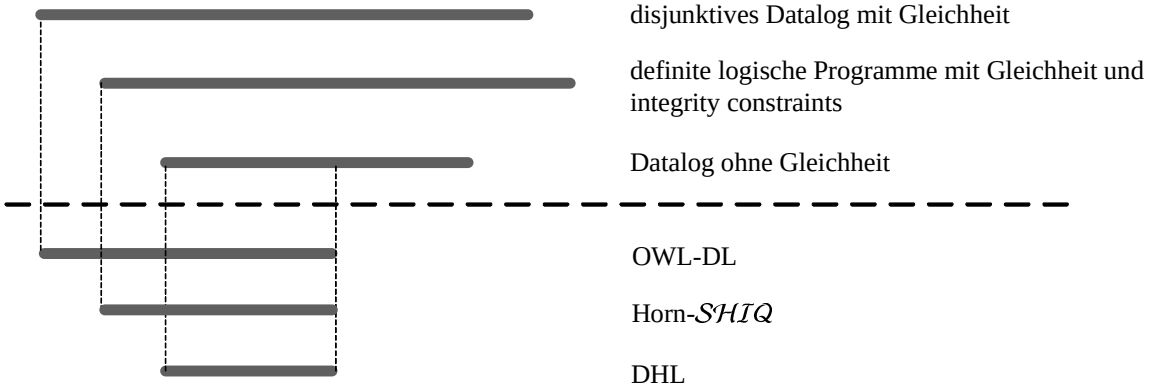

Bild 2.10: DHL Einordnung

Der letzte in diesem Abschnitt behandelte Ansatz [Vol04, GHVD03] nähert sich einer Integration von der Seite der Regelsprache. Der Ausgangspunkt der Untersuchungen ist, wie auch bei [MSS04], die axiomatische Semantik der DL (siehe Seite 26, Abschnitt 2.1.2.4). Der Fokus der Untersuchungen in [Vol04] liegt dabei auf der Bestimmung des DL-Fragments *description horn logic* (DHL), der mit Regeln der Form $(*)$, also mit Datalog ohne Gleichheit, axiomatisierbar ist, was die sogenannten Klasse der *description logic programs* (DLP) als Ergebnis hat. Der jüngste Ansatz aus der Gruppe der Ansätze, die sich auf Horn-Logik beschränken, wird in [HMS05] behandelt. Dort wird das entsprechende DL-Fragment *Horn-*$\mathcal{SHIQ}$ vorgestellt.

Das in Bild 2.10 dargestellte Diagramm zeigt eine Einordnung von DHL bezüglich ihrer Mächtigkeit. DHL liegt im Schnitt von Datalog ohne Gleichheit und OWL-DL. [Vol04] und [HMS05] leiten darüberhinaus her, daß sich mit definiten logischen Programmen mit Gleichheit und integrity constraints der axiomatisierbare DL-Umfang erhöhen läßt (in [HMS05] zu Horn-$\mathcal{SHIQ}$), da teilweise mit integrity constraints die klassische Negation und mit der Zulassung von Funktionssymbolen der Existenzquantor im Regelkopf ermöglicht wird. Mit Hilfe von disjunktivem Datalog mit Gleichheit läßt sich OWL-DL in vollem Umfang axiomatisieren [MSS04, HMS04].

Den Abschnitt abschließend sei in Tabelle 2.2 eine Übersicht über die Eigenschaften der hier vorgestellten Ansätze zur DL-Regelintegration gegeben. Die Integrationsstärke bezieht sich auf Bild 2.9 auf Seite 61. OWL-DL entspricht der Sprachklasse $\mathcal{SHOIN}(\mathbf{D})$. Die Sprachklasse von DHL kann nicht mit der gängigen DL-Klassifikation eingeordnet werden, liegt aber

eines DL-Atoms der Form $c(x, y)$ in einem non-DL-Atom des Rumpfes vorkommen muß.

Ansatz	**Integrations-stärke**	**DL-Mächtig-keit**	**Bedingungen** für Regeln der Form $(*)$ (Seite 61)
AL-Log	a)	$\mathcal{ALC}$	B_i eingeschränkt auf $c(t)$ DL-safety, AL-Log-*safety* A_0 nur non-DL-Atom
CARIN	a)	$\mathcal{ALCNR}$	role-safety A_0 nur non-DL-Atom
ORL	a)+b)	OWL-DL	keine non-DL-Atome in der Regel
[MSS04]	a)+b)	OWL-DL	DL-safety
DLP/[HMS05]	a)+b)	DHL/Horn-$\mathcal{SHIQ}$	- / DL-safety

Tabelle 2.2: Übersicht zur Integration von DL und Regeln

unterhalb von OWL-DL und Horn-$\mathcal{SHIQ}$ (zur detaillierteren DL-Zuordnung siehe [Vol04]).

Aufgrund der besonderen Bedeutung von DLP für die vorliegende Arbeit und ihrer Relevanz für den praktischen Einsatz [HHK$^+$05, HSS06] wird im folgenden die bei der Definition der in dieser Arbeit vorgestellten Ontologie-Beschreibungssprache verwendete Transformation von DLP nach Datalog aufgeführt. Dazu werden zunächst die wohlbekannten Lloyd-Topor-Eigenschaften (für den monotonen Fall) aufgeführt.

Satz 2.2. (**Lloyd-Topor monoton**) Seien A, B, C prädikatenlogische Ausdrücke. Dann gilt

(i) $A \rightarrow (B \wedge C)$ genau dann, wenn $A \rightarrow B \wedge A \rightarrow C$.

(ii) $(A \vee B) \rightarrow C$ genau dann, wenn $A \rightarrow C \wedge B \rightarrow C$.

(iii) $A \rightarrow (B \rightarrow C)$ genau dann, wenn $A \wedge B \rightarrow C$.

(iv) $A \rightarrow \neg B$ genau dann, wenn $A \wedge B \rightarrow \mathit{false}$. □

Die nachstehende Definition beschreibt die Transformation von DLP nach Datalog.

Definition 2.3. (**DLP-Transformation nach [Vol04]**)

concept-descriptions, -axioms :

$$
\begin{array}{rcll}
\phi_{\mathcal{LP}}(C \equiv D) & \longrightarrow & \phi_{\mathcal{LP}}(C \sqsubseteq D), \phi_{\mathcal{LP}}(D \sqsubseteq C) & \\
\phi_{\mathcal{LP}}(C \sqsubseteq D) & \longrightarrow & \phi^L_{\mathcal{LP}}(C, y_i) \rightarrow \phi^R_{\mathcal{LP}}(D, y_i) & \\
\phi^R_{\mathcal{LP}}(A, x) & \longrightarrow & A(x) & \\
\phi^R_{\mathcal{LP}}(\textbf{\textit{and}}(C, D), x) & \longrightarrow & \phi^R_{\mathcal{LP}}(C, x) \wedge \phi^R_{\mathcal{LP}}(D, x) & \\
\phi^R_{\mathcal{LP}}(\textbf{\textit{all}}(R, C), x) & \longrightarrow & R(x, y_i) \rightarrow \phi^R_{\mathcal{LP}}(C, y_i) & \\
\phi^R_{\mathcal{LP}}(\textbf{\textit{some}}(R.\{o\}), x) & \longrightarrow & R(x, o) & \\
\phi^L_{\mathcal{LP}}(A, x) & \longrightarrow & A(x) & \\
\phi^L_{\mathcal{LP}}(\textbf{\textit{and}}(C, D), x) & \longrightarrow & \phi^L_{\mathcal{LP}}(C, x) \wedge \phi^L_{\mathcal{LP}}(D, x) & \\
\phi^L_{\mathcal{LP}}(\textbf{\textit{or}}(C, D), x) & \longrightarrow & \phi^L_{\mathcal{LP}}(C, x) \vee \phi^L_{\mathcal{LP}}(D, x) & \\
\phi^L_{\mathcal{LP}}(\textbf{\textit{some}}(R, C), x) & \longrightarrow & R(x, y_i) \wedge \phi^L_{\mathcal{LP}}(C, y_i) & \\
\phi^L_{\mathcal{LP}}(\textbf{\textit{some}}(R, \{o\}), x) & \longrightarrow & R(x, o) & \\
\phi^R_{\mathcal{LP}}(\textbf{\textit{not}}(A), x) & \longrightarrow & \neg A(x) & (*)
\end{array}
$$

role-axioms :

$$
\begin{array}{rcl}
\phi_{\mathcal{LP}}(P \equiv Q) & \longrightarrow & \phi_{\mathcal{LP}}(P \sqsubseteq Q), \phi_{\mathcal{LP}}(Q \sqsubseteq P) \\
\phi_{\mathcal{LP}}(P \equiv \textbf{\textit{inverse}}(Q)) & \longrightarrow & (Q(y, x) \rightarrow P(x, y)), (P(y, x) \rightarrow Q(x, y)) \\
\phi_{\mathcal{LP}}(P \sqsubseteq Q) & \longrightarrow & Q(x, y) \rightarrow P(x, y) \\
\phi_{\mathcal{LP}}(\textbf{\textit{transitive}}(P) \sqsubseteq P) & \longrightarrow & P(x, z) \wedge P(z, y) \rightarrow P(x, y)
\end{array}
$$

assertions :

$$
\begin{array}{rcl}
\phi_{\mathcal{LP}}(a : D) & \longrightarrow & \phi^R_{\mathcal{LP}}(D, a) \\
\phi_{\mathcal{LP}}((a, b) : P) & \longrightarrow & P(a, b)
\end{array}
$$

Tabelle 2.3: Transformation $\phi_{\mathcal{LP}}$

(i) $\mathrm{lloydTopor}$ bezeichne die Transformation gemäß den Implikationen von links nach rechts aus Satz 2.2.(i)-(iv).

(ii) Sei D ein DLP und $\phi_{\mathcal{LP}}$ gegeben durch Tabelle 2.3, so ist die Transformation dlpd von DLP nach Datalog definiert durch

$$\mathrm{dlpd}(D) = \mathrm{lloydTopor}(\phi_{\mathcal{LP}}(D)).$$

Ein mit dlpd erzeugtes Datalog-Programm sowie deren Formeln und Atome werden *DLP-konform* genannt. □

Bemerkung 2.4. DLP-Konformität dient zur einheitlichen Charakterisierung von Programmen, Formeln und Atomen. Bezogen auf Atome ist dieser Begriff natürlich austauschbar mit dem Begriff DL-Atom. Tabelle 2.3 zeigt genau diejenigen Transformationsregeln für $\mathcal{L}_2$-DLPs von Volz, welche nicht des Gleichheitsprädikats bedürfen.[50] DLPs, die dadurch erfasst

[50] Da für unsere Zwecke eine Trennung von T- und A-Box nicht notwendig ist, wird hier nicht die Definition $\mathcal{KB}_i^{DLP}$ für eine Wissensbasis der Stufe i verwendet, sondern von einem $(\mathcal{L}_i)$-DLP gesprochen.

werden, werden in dieser Arbeit mit $\mathcal{L}_0^{IC}$-DLPs bezeichnet, die sich nur in der Hinzunahme des ***not***-Konstrukts (siehe $(*)$ in Tabelle 2.3) von $\mathcal{L}_0$ unterscheiden.

Bemerkung 2.5. (***subsumption check*** **nach [Vol04, Abschnitt 5.4]**) Die Prüfung, ob zwei Klassen in einer Subklassenbeziehung stehen, kann nicht auf Datalogebene direkt vorgenommen werden, da keine Implikationen ableitbar sind. Der Test, ob A eine Subklasse von B ist bei einem gegebenen DLP P, wird vorgenommen mit Hilfe des Tests, ob $B(c)$ aus $P \cup \{A(c)\}$ ableitbar ist, wobei A, B Klassennamen sind und c eine noch nicht in P vorkommende Konstante ist. Der Test ist nur anwendbar, wenn keine *negation as failure* verwendet wird.

Der in Bemerkung 2.5 dargestellte Zusammenhang wird in dem nachstehenden Satz in allgemeinerer Form formuliert, da er für die Entwicklung der Sprache in Kapitel 4 noch benötigt wird. Es bezeichne $p[x]$ eine Formel, in dem die Variable x als einzige freie Variable vorkommt, und $p[x/t]$ die Formel, die durch Ersetzung von x mit t in p entsteht.

Satz 2.6. Seien $p[x], q[x], F$ prädikatenlogische Formeln resp. eine Formelmenge, in denen die Konstante s nicht vorkommt. Dann gilt $F \models p[x] \rightarrow q[x]$ genau dann, wenn $F \cup \{p[x/s]\} \models q[x/s]$.

Beweis:

$F \models p[x] \rightarrow q[x]$

gdw. $F \models \forall x(p[x] \rightarrow q[x])$

gdw. $F \cup \{\exists x \neg(p[x] \rightarrow q[x])\}$ ist unerfüllbar

gdw. $F \cup \{\neg(p[x/s] \rightarrow q[x/s])\}$ ist unerfüllbar (Erfüllbarkeitsäquivalenz der Skolemisierung)

gdw. $F \models p[x/s] \rightarrow q[x/s]$

gdw. $F \cup \{p[x/s]\} \models q[x/s]$ □

3 Anforderungen an eine integrative Ontologie-Beschreibungssprache

3.1 Anwendungsorientierte Motivation und Anforderungen

3.1.1 Integration von Klassen und Begriffsnetzen

Die klassische Vorgehensweise bei der Informationsmodellierung ist von einer Separation geprägt, die im folgenden an Hand eines kleinen Beispiels erläutert wird, nämlich der Mo-

dellierung eines Informationsportals für die Domäne IT-Weiterbildung mit dem Ausschnitt *Lehrgänge*. Es ist erforderlich,

a) ein Begriffsnetz zur Konzeptualisierung der Domäne IT-Weiterbildung und

b) ein Klassendiagramm für Lehrgänge und deren Eigenschaften

zu erstellen (siehe Bild 3.1).[1]

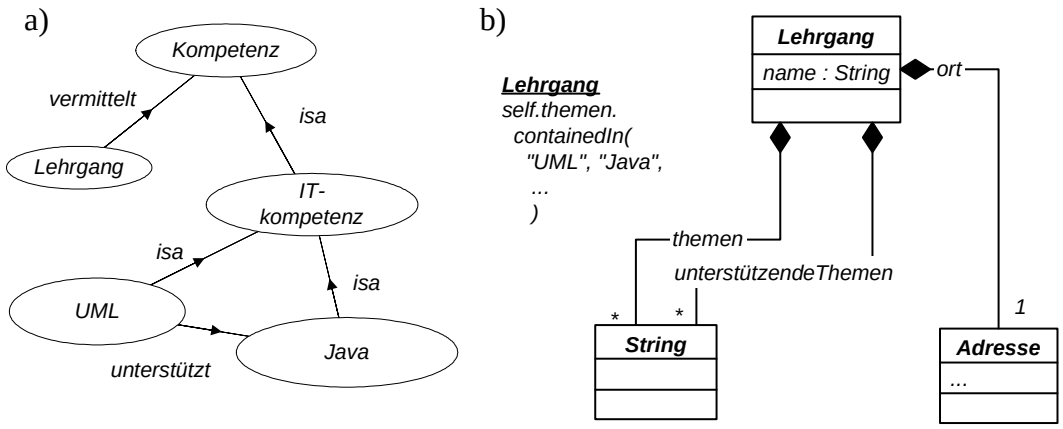

Bild 3.1: Modellausschnitte der Domäne IT-Weiterbildung

a) stellt Begriffe und deren Beziehungen untereinander in den Vordergrund. Dort werden Aussagen getroffen wie beispielsweise ‚Ein Lehrgang vermittelt Kompetenz' oder ‚Java-Programmierung ist eine IT-Kompetenz'. b) zielt auf eine Klassifizierung mittels der Daten eines Lehrganges wie sein Name, Ort und die zu vermittelnden Themen ab. Bei b) steht eine extensionsorientierte Charakterisierung von Lehrgängen mit Hilfe der Definition ihrer Eigenschaften im Vordergrund. Das Modell hat den Zweck, den Instanzraum, also die Menge der möglichen Objektnetze, einzuschränken. Bei der Entwicklung von a) hingegen steht nicht die Charakterisierung von Begriffsextensionen im Vordergrund, sondern – vergleichbar mit der Erstellung eines Thesaurus – die Aufführung der Begriffe der Domäne und Aussagen, in welcher Beziehung diese stehen.

[1] Eine weitere Dimension, nämlich die Integration der syntaktischen Modellierung (über formale Grammatiken) wird in [Kut94] behandelt.

Eine offensichtliche Konsequenz dieser Separation ist eine begriffliche Mehrfachdefinition. Ein Beispiel dafür ist der *Lehrgang*, der zum einen im Klassendiagramm mit den Eigenschaften *Name* und *Ort* und zum anderen im Begriffsnetz mit seiner Beziehung zu *Kompetenz* vorkommt. Eine weitere Art der Mehrfachdefinition bezieht sich nicht auf die Ebene der vergleichbaren syntaktischen Elemente, sondern kommt nur implizit vor. Ein Beispiel hierfür ist der OCL-Ausdruck in Diagramm b), welcher die möglichen Lehrgangsthemen auf Zeichenketten beschränkt, die aber zugleich auch als Begriffe in a) definiert sind.[2] Ein weiteres Beispiel ist die Festlegung, daß zu jedem Lehrgangsthema die unterstützenden Themen aufgeführt sollen. Bei einer integrativen Lösung könnte ein entsprechende Randbedingung für diese Eigenschaft aus der *unterstützt*-Beziehung in a) abgeleitet werden.

Der Vermeidung einer Mehrfachdefinition von Strukturen und Benennungen in der begriffs- und klassenorientierten Modellierung dient die folgende Anforderung.

> **Anforderung I (Begriffs- und Klassenkonstrukte):** Die Beschreibungssprache sollte die Konstrukte aus der begriffs- und klassenorientierten Modellierung umfassen.

3.1.2 Offene Instanzketten

Eines der Schlüsselkonstrukte in der Informationsmodellierung ist die Instanzbildung. Eine Instanzbeziehung $(a \ \textit{instanceOf} \ b)$ drückt aus, daß a zu der Extension der intensionalen Definition b gehört. Dabei kann b eine formale Grammatik und a eine Zeichenkette im Fall der syntaktischen Modellierung oder eine Klasse resp. Objekt im Fall der klassenbasierten Modellierung sein. Diese Beziehungssetzung erzwingt, daß Struktur und Inhalt von a der Struktur- und Inhaltsdefinition gegeben durch b genügt.

Bild 3.2 soll diesen Sachverhalt und die Bildung von Instanzketten an dem Lehrgangsbeispiel verdeutlichen:[3] für jede Instanz von *Lehrgang* muß die Eigenschaft *zertifiziertMit* bestimmt sein. Die Instanz *L1474* besitzt das Zertifikat *IHK-Z-09*. Lehrgänge als Gesamtheit betrachtet gehört wie auch das Studium zu der Menge von Qualifizierungsarten. Qualifizierungsarten besitzen unter anderem die boolsche Eigenschaft, ob sie zu einem akademischen Abschluß führen.

[2] Auf den Ansatz, Begriffe des Netzes einfach als Klassen in dem Diagramm b) zu notieren, wurde hier bewußt verzichtet, da dies in der Praxis meist nicht geschieht. Zudem ist diese Gleichsetzung zu einschränkend, da der Begriffsraum mehr als nur Klassen umfaßt (vergl. das Konzept der Ressource aus RDF in Abschnitt 2.2.1).

[3] Die hier eingeführte Legende gilt auch für die nachfolgenden Abbildungen.

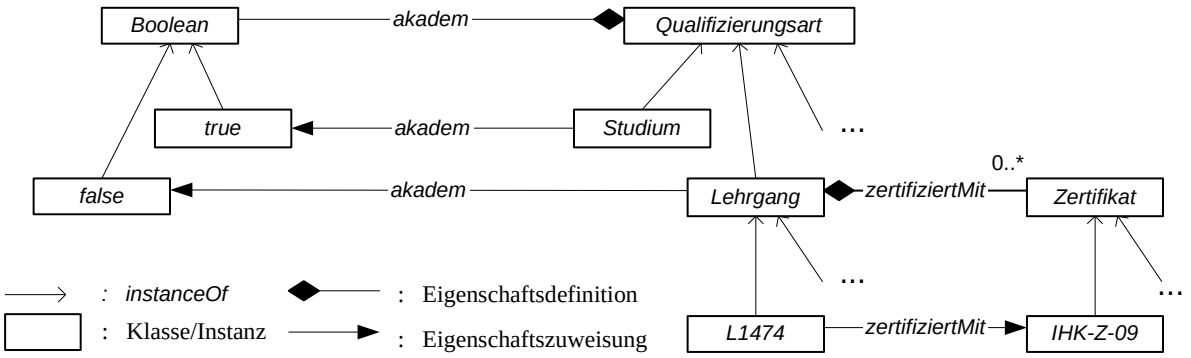

Bild 3.2: Instanzketten

Folgerichtig besitzen Instanzen von *Qualifizierungsart*, nämlich *Lehrgang* und *Studium*, eine entsprechende Eigenschaftszuweisung. Somit können intensionale Definitionen wiederum selber zu einer Extension intensionaler Definitionen gehören, also Modellelemente eine Doppelrolle einnehmen: zum einen als Klasse und zum anderen als Instanz, d.h. Klassen werden als *first class citizens* behandelt. Da für eine Domäne beliebig aufeinander folgende Instanzbeziehungen möglich sind, wird auch von *offenen* Instanzketten gesprochen. Dies spiegelt die nachstehende (rekursiv definierte) Anforderung wieder.

Anforderung II (Klassen als Instanzen): Die Beschreibungssprache sollte zulassen, daß Klassen als Instanzen einer Klasse definiert werden können.

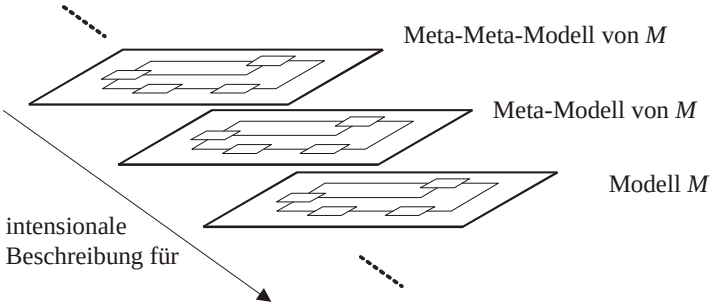

Bild 3.3: Modellhierarchie

Der durch die Rekursivität gegebene Instantiierungsprozess bestehend aus dieser ausgezeichneten Beziehungssetzung zwischen Klassen und ihren Instanzen läßt eine Hierarchie von Modellen entstehen, in der das *Metamodell* die Typen für die nächst tiefere Hierarchiestufe *Modell* bereitstellt (Bild 3.3)[4]. Das Modell selber dient wiederum als Metamodell für die nächst tiefere Stufe (vergl. [Ode95, AK00]).

3.1.3 Begriffskontexte

Der Zusammenhang, in dem ein Begriff steht, der Begriffskontext, ist aus Modellsicht in kanonischer Art und Weise durch die Struktur des Begriffsnetzes gegeben. Die einen Begriff über Beziehungen umgebenden Begriffe bilden seinen Kontext, der im weiteren als *impliziter* Kontext bezeichnet wird. Demgegenüber steht eine *explizite* Kontextbildung, bei der innerhalb des Begriffsnetzes Begriffe in eine Kontextbeziehung zu anderen Begriffen gesetzt werden. Bild 3.4 zeigt eine Verfeinerung der Kontextarten, die im folgenden näher erläutert wird.

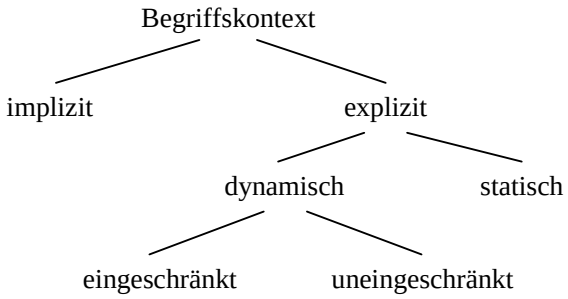

Bild 3.4: Arten des Begriffskontextes

Die Angabe eines expliziten Kontextes dient zur Auflösung von Begriffsmehrdeutigkeiten, beispielsweise zur Unterscheidung des Homonyms *Java* zum einen als Programmiersprache und zum anderen als Kaffeesorte. Zu diesem Zwecke kann ein Thesaurus gemäß dem Standard [Ins94] die Kontextterme *Java(Programmierung)* und *Java(Kaffee)* zur Verfügung stellen.[5]

Die Untergliederung in einen statischen und dynamischen Kontext bezieht sich auf die Begriffsverwendung bei der Kategorisierung von Objekten. Statische Kontexte sind im Begriffs-

[4] Man beachte, daß hier zur Veranschaulichung die mögliche Zyklizität dieser Beziehungssetzung, die in den F-Logic-basierten Ansätzen und in dem hier vorgestellten Ansatz möglich ist, nicht dargestellt wird. Der Bezugspunkt M ist hier beliebig gewählt. Üblicherweise wird die tiefste Ebene als Datenebene bezeichnet.

[5] Diese werden im Standard *parenthetical qualifiers* genannt.

netz verankert, das heißt während der Kategorisierung stehen lediglich die dort vorhandenen Kontextterme zur Verfügung. Dahingegen läßt eine dynamische Kontextbildung zu, daß während der Kategorisierung eine Bildung von neuen Kontexttermen, die nicht im Begriffsnetz vorkommen müssen, erfolgen kann.

Eine uneingeschränkte dynamische Kontextbildung läßt die Bildung von beliebigen Kontextermen bei der Kategorisierung zu. Eine Einschränkung kann auf dem impliziten Kontext basieren und zwar derart, daß Terme entlang der Spezialisierungs- und Instanz-Beziehung gebildet werden. Diese Art Kontextbildung wird im folgenden als *taxonomische Kontextbildung* bezeichnet. Die damit gebildeten Kontextterme werden im weiteren *Kontextpfade* genannt.

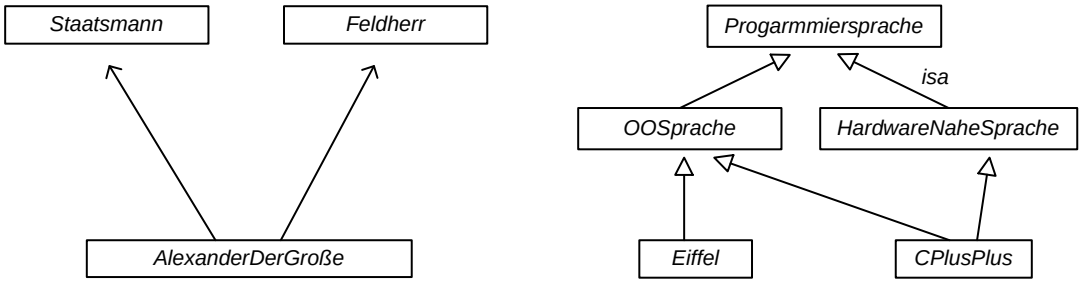

Bild 3.5: Taxonomischer Kontext

Bild 3.5 zeigt ein Beispiel für die taxonomische Kontextbildung: *geschichtsKurs013* hat als Thema ‚Alexander der Große als Feldherr' und *lehrgang1474* hat als Thema ‚CPlusPlus als hardwarenahe Programmiersprache'. Die erste Kategorisierung basiert auf der Instanzbeziehung während letztere auf der Spezialisierungsbeziehung basiert. Generell sind nicht nur zweistellige Kontextpfade, sondern auch mehrstellige entlang den taxonomischen Beziehungen zugelassen. Zusammenfassend ergibt sich die folgende Anforderung.

> **Anforderung III (Taxonomische Begriffskontexte):** Die Beschreibungssprache sollte spezielle Konstrukte zur taxonomischen Kontextbildung umfassen.

3.1.4 Kategorisierung

Sowa beschreibt in seinen Betrachtungen zu Ontologien den Begriff Kategorie wie folgt: *„the word category is associated with the process of classifying entities according to some monadic predicate. In that sense, a category may be considered a type [class] used for the purpose of classification“* [Sow00].[6] Diese allgemeine Charakterisierung sagt zunächst nichts darüber aus, welche Begriffsmengen bei der Kategorisierung herangezogen werden dürfen.

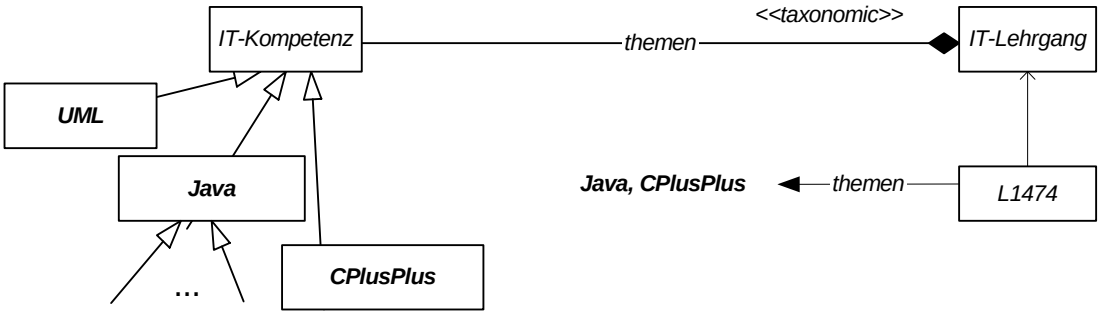

Bild 3.6: Taxonomische Kategorisierung

Eine spezielle Art der Beschränkung des Begriffsraums für die Kategorisierung, nämlich auf Taxonomien, wird in Bild 3.6 verdeutlicht. Die Definition der Eigenschaft *themen* besitzt den Wertebereich *IT-Kompetenz* und ist ausgezeichnet als eine kategorisierende Eigenschaft, was ausdrücken soll, daß bei Instantiierung dieser Eigenschaft beliebige Mengen über denjenigen Modellentitäten[7] zugeordnet werden, die unter *IT-Kompetenz* subsumiert werden können.

Dieses Beispiel verdeutlicht zugleich die in Abschnitt 3.1.1 postulierte Integration von Begriffsnetzen und Klassendefinitionen. Die Möglichkeiten zur Begriffsreferenzierung werden explizit im Modell durch entsprechende Beziehungssetzungen, annotiert mit *taxonomic*, verankert und nicht, wie in den Modellen a) und b) aus Bild 3.1, durch bloße Gleichheit der Zeichenketten indirekt ausgedrückt, was leicht mit einer redundanten und damit fehlerträchtigen Nachführung bei Erweiterung des Begriffsraums unterhalb *IT-Kompetenz* einhergehen kann. Eine allgemeinere Form der Definition des Wertebereichs von kategorisierenden Eigenschaften, und zwar mit Hilfe von Anfragen, wird in Abschnitt 3.1.6 vorgestellt. Zusammenfassend läßt sich die nachstehende Anforderung aufstellen.

[6] Diese Beschreibung basiert auf der üblichen Interpretation von Klassen als monadische Prädikate, so wie auch in der axiomatischen Semantik der DL in Abschnitt 2.1.2.4 aufgezeigt.

[7] Ganz allgemein können damit Klassen, Instanzen oder Kontextpfade (siehe Abschnitt 3.1.3) gemeint sein.

Anforderung IV (Taxonomische Kategorisierung): Die Beschreibungssprache sollte spezielle Konstrukte zur Definition der für eine taxonomische Kategorisierung zulässigen Modellelemente beinhalten.

3.1.5 Beziehungskontexte

So wie in Abschnitt 3.1.3 Begriffe in einen Kontext gestellt werden können, führt auch die Kontextualisierung von Beziehungssetzungen zu einer höheren Ausdrucksstärke. Damit können alle Beziehungsarten, seien es Generalisierungs- und Instantiierungs-Beziehungen, Eigenschaftsdefinitionen und -zuweisungen oder allgemein benannte Beziehungen in einen Kontext gestellt werden. Bezüglich der Pragmatik der Kontextualisierung lassen sich drei Arten unterscheiden, nämlich die

- Modularisierung,
- statische Relativierung und
- dynamische Relativierung (Transformation)

von Beziehungssetzungen. Das Ziel der Modularisierung ist die Aufteilung der Gesamtontologie in einzelne Teile, wobei sich der Kontextbezeichner dann als Modulbezeichner ansehen läßt.

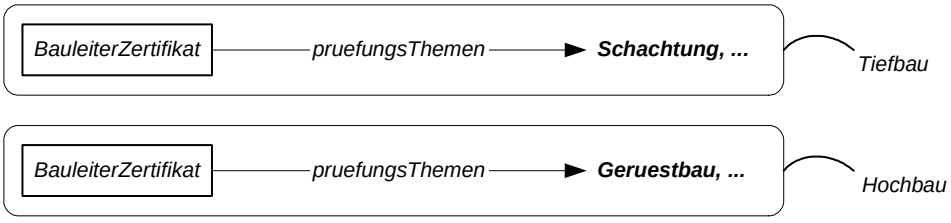

Bild 3.7: Beziehungskontext

Über diese klassische Verwendung hinausgehend kann der Kontext zur Relativierung von Beziehungssetzungen verwendet werden. Beispielsweise bestimmen die Beziehungssetzungen aus Bild 3.7 die Prüfungsthemen für ein Zertifikat zum Bauleiter relativ zu den verschiedenen

Sparten. Es sei bemerkt, daß ganz allgemein dieses Konstrukt dazu dienen kann, Eigenschaften derselben Modellelemente in verschieden Kontexten (beziehungsweise aus verschiedenen Sichten) zu betrachten. Ein weiteres Beispiel wäre die Spezifikation der Aktivität Materialbeschaffung als Bestandteil eines Unternehmensmodells – einmal im Kontext Management mit ihren Beziehungen zu Organisationseinheiten und zum anderen im Kontext Technik mit ihren Beziehungen zu IT-Systemen.

Die letzte Verwendungsart des Kontextes ist die der Transformation. Darunter soll die Überführung von Beziehungssetzungen zwischen Kontexten mit Hilfe von Regeln verstanden werden. Die in TRIPLE notierte Regel

```
forall s,o
    s [ pruefungsThemen -> o ] @ HochTief  <-
        s [ pruefungsThemen -> o ] @ Hochbau  or  s [ pruefungsThemen -> o ] @ Tiefbau.
```

definiert z.B. den Kontext *HochTief* als Vereinigung aller *pruefungsThemen*-Beziehungen aus den Kontexten *Hochbau* und *Tiefbau*. Zusammenfassend läßt sich die nachstehende Anforderung formulieren.

> **Anforderung V (Beziehungskontexte):** Die Beschreibungssprache sollte spezielle Konstrukte zur Definition von Beziehungskontexten umfassen.

3.1.6 Klassen-definierende Anfragen

Die klassische Informationsmodellierung (beispielsweise mit ER oder UML) geht von einer statischen Definition von Klassen aus. Die Bildung von Klassen ist beschränkt auf die Subklassenbildung *(A_1 subClassOf A_2, ..., A_n)* mit den Klassennamen A_i. Die Zielklassen von Eigenschaftsdefinitionen sind auf Klassennamen beschränkt. Die in Abschnitt 3.1.2 gestellte Forderung, das Klassen als *first class citizens* zu betrachten sind, sind als Voraussetzung dafür anzusehen, Anfragen zu bilden, die Klassen als Ergebnis zurückliefern, womit zusätzlich der Wertebereich von Eigenschaften mit Hilfe von Anfragen festgelegt werden kann. Im folgenden soll der Einsatz und die Sinnhaftigkeit von klassendefinierenden Anfragen an Hand eines Beispiels erläutern werden. Dabei soll die umgangssprachliche Form des aufzustellenden Modells gegeben sein durch

a) *Eine Berufsausbildung befähigt nur zu einer Qualifizierung, die nicht zu einem akademischen Abschluß führt.*

b) *Ein IHK-Experten-Lehrgang ist ein Lehrgang, der zertifiziert ist mit einem Zertifikat der IHK-Einordnung „schwer‘.*

c) *Eine Web-Weiterbildung ist eine Weiterbildung, die nur Kenntnisse vermittelt, welche im Web-Engineering oder -Design vorausgesetzt werden.*

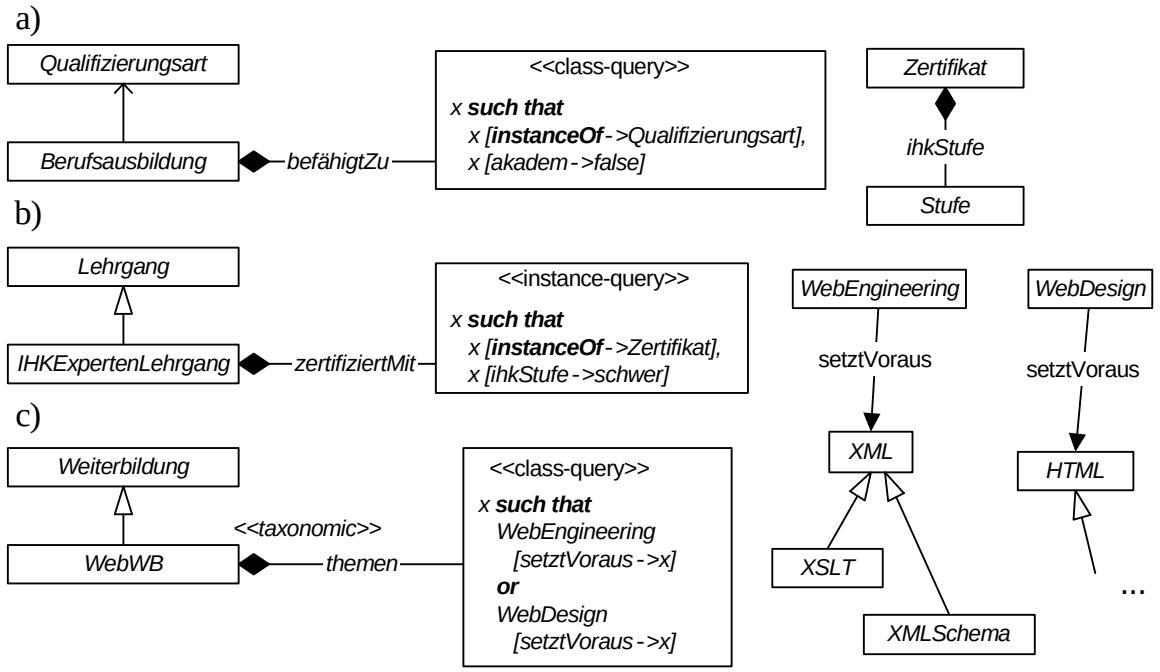

Bild 3.8: Klassen-definierende Anfragen

Die rechte Seite des Bildes 3.8 zeigt eine Modellierung dieser drei Aussagen. Auf der linken Seite sind die dazu verwendeten zusätzlichen Elemente aufgeführt.

Alle drei Modellausschnitte verwenden eine Anfrage zur Spezifikation des Wertebereichs der entsprechenden Eigenschaften. Im Falle der Klassen liefernden Anfrage in Modellausschnitt a) (<<*class-query*>>) ist dann die Zielklasse der Eigenschaft die Vereinigung aller Klassen, die zur Extension der Anfrage gehören. Das heißt, daß die Zielklasse der Eigenschaft *befähigtZu* die Vereinigung aller Klassen ist, die zur Extension von *Qualifizierungsart* gehören und nicht zu einem akademischen Abschluß führen. Dies kann beispielsweise die Klasse *Lehrgang*, *Weiterbildung* oder ähnliches sein. Für eine ganz konkrete Berufsausbildung, nämlich einer Instanz

von *Berufsausbildung*, muß dann der Eigenschaft *befähigtZu* ein ganz konkreter Lehrgang oder eine Weiterbildung zugewiesen werden.

Eine weitere Anfrageverwendung (<<*instance-query*>>) ist in Modellausschnitt b) zu sehen. Dort allerdings ist der Wertebereich direkt gegeben durch die Anfrageauswertung. Der Wertebereich ist sozusagen gleich der Menge derjenigen Elemente, die durch Auswertung der Anfrage geliefert werden. Das heißt, daß für eine Instanz von *IHKExpertenlehrgang* der Eigenschaft *zertifiziertMit* ein beliebiges Element dieser Menge zugewiesen werden kann. Sie umfaßt aber nur Zertifikate, welche die Eigenschaftzuweisung *ihkStufe=schwer* besitzen.

Modellausschnitt c) schließlich kombiniert eine Klassen-definierende Anfrage mit einer Kategorisierung. Das Ergebnis der Anfrageauswertung über dem Modellausschnitt ist die Menge $\{$*XML*, *HTML*$\}$. Entsprechend den Erläuterungen aus Abschnitt 3.1.4 kann nun der Eigenschaft *themen* eine Menge von Kontextpfaden zugeordnet werden, die aus den Taxonomien beginnend mit *XML* oder *HTML* stammen.

Man kann insbesondere Klassen-definierende Anfragen als ein dynamisches Einfügen von künstlichen Generalisierungsbeziehungen betrachten. Ausschnitt a) zeigt dies besonders deutlich. Statt der Diskriminierung durch die Bedingung *akadem=false* könnten alle dementsprechenden Klassen unter der künstlichen Oberklasse *NichtAkademischeQualifizierungsart* subsumiert werden, der dann als Zielklasse der Eigenschaft verwendet wird. Jedoch ist die Flexibilität und die Übersichtlichkeit stark eingeschränkt, da bei komplexeren Bedingungen, die über bool'sche Bedingungen hinausgehen, die Anzahl der künstlichen Oberklassen, die der Modellentwickler einfügen müßte, sehr groß werden kann, was zudem die Wahrscheinlichkeit von fehlerhafter Modellerstellung erhöhen würde. Das hier vorgestellte Konstrukt zur flexiblen Klassenbildung schlägt sich in der folgenden Anforderung nieder.

> **Anforderung VI (Klassen-definierende Anfragen):** Die Beschreibungssprache sollte Konstrukte zur dynamischen Klassenbildung mittels Anfragen umfassen.

3.2 Zusammenfassender Vergleich

Die bisher aufgestellten Anforderungen werden von den in Abschnitt 2 erörterten Sprachen in unterschiedlicher Art und Weise unterstützt. Tabelle 3.1 gibt eine Übersicht, in der für die einzelnen Sprachen der Erfüllungsgrad angezeigt wird. Eine direkte Unterstützung bedeutet, daß die Sprache ein entsprechendes Konstrukt anbietet. Bei einer indirekten Unterstützung ist die entsprechende Funktionalität nur mit Hilfe mehrerer Konstrukte abbildbar/simulierbar. OWL-Full ist nicht explizit aufgeführt, da es als Zusammenführung von RDF und OWL-DL angesehen werden kann.

Anforderung	**Beschreibungssprache**				
	F-Logic	RDF	OWL-DL	OWL-DL + Regeln	TRIPLE
I Begriffs- und Klassenkonstrukte	+++	+++	-	-	+++
II Klassen als Instanzen	+++	+++	-	-	+++
III Taxonomische Begriffskontexte	++	+	+	+	++
IV Taxonomische Kategorisierung	++	+	-	+	++
V Beziehungskontexte	++	+	+	+	+++
VI Klassen-definierende Anfragen	++	-	+	+	++

- wird nicht unterstützt
+ wird teilweise unterstützt
++ wird indirekt unterstützt
+++ wird direkt unterstützt

Tabelle 3.1: Erfüllungsgrad

Anforderung I und II stehen in Abhängigkeit insofern, daß II als Voraussetzung von I anzusehen ist, weshalb hier auf beide zusammen eingegangen wird. Anforderung II erzwingt die Behandlung von Klassen als *first class citizens*. DL sieht jedoch nur zwei Modellebenen vor, nämlich die Klassen- und Instanzebene (TBox resp. ABox), im Gegensatz zu der hier vorgeschlagenen Hierarchie beliebiger Tiefe. Es ist sowohl nicht möglich, Klassen in frei benannte Beziehung zu setzen, als auch Klassen als Individuen zu interpretieren, welche dann

mit anderen Individuen Beziehungen eingehen. Genauer: die Angabe des DL-Fragments

$$c \sqsubseteq d, \quad o : c, \quad (c, d) : r1, \quad (o, c) : r2 \quad ,$$

welches mit dem dritten Ausdruck zwei Klassen und mit dem vierten eine Klasse und ein Individuum in Beziehung setzt, ist nicht möglich.[8] Damit wird Anforderung II durch OWL-DL(+Regeln) nicht unterstützt.

Eine Konsequenz davon ist die Nichterfüllung von Anforderung I durch OWL-DL(+Regeln). Denn genau II müßte erfüllt werden, um Ausdrücke wie

$$\begin{aligned} &Lehrgang \sqsubseteq \textbf{all}\ (zertifiziertMit, Zertifikat), \\ &Lehrgang : Qualifizierungsart \\ &(Lehrgang, Kompetenz) : vermittelt \quad , \end{aligned}$$

die *Lehrgang* als Klasse und Instanz definiert und zugleich innerhalb einer Beziehungssetzung verwendet, zu erlauben (vergl. Bild 3.1 auf Seite 68 und Bild 3.2 auf Seite 70).

Die restlichen Sprachen erfüllen Anforderung I und II. Die Basisbausteine in F-Logic (*Grund-id-Terme*) bezeichnen sowohl Klassen als auch Instanzen und machen damit offene Instanzketten möglich (siehe Abschnitt 2.1.1.1). In RDF und damit auch TRIPLE sind die Basisbausteine Ressourcen, mit denen sich zunächst alle Begriffe notieren lassen (siehe Abschnitt 2.2.1). RDF kennt nun zusätzlich das Konstrukt *Class*, womit Ressourcen als Klassen ausgezeichnet werden können. Dies ermöglicht eine explizite Unterscheidung zwischen Begriffen, für die Extensionen vorgesehen sind (z.B. Klasse *Lehrgang*) und solchen, die nicht extensionsorientiert sind und vorzugsweise zur Kategorisierung herangezogen werden (z.B. Begriff *UML*).[9]

Anforderung III fordert die Unterstützung von taxonomischen Begriffskontexten. Auf der Ebene der Sprachelemente entspricht dies der einer ausgezeichneten Termbildung. RDF erlaubt nur eingeschränkt eine explizite Bildung von Begriffskontexten, denn es verwendet als Basisbaustein die Form $n : l$, wobei n den Namensraum und l den lokalen Namen bezeichnet (siehe Abschnitt 2.2.1.1), womit unterschiedliche Vokabularien gebildet werden können. n läßt sich also als Kontext des lokalen Namens l ansehen. Allerdings werden die Namensräume bei der Definition von Vokabularien festgelegt und können nicht dynamisch bei der Verwen-

[8] Dies wird besonders deutlich, wenn man das Fragment bzgl. der axiomatischen Semantik betrachtet (siehe μ auf Seite 26). Denn das Transformationsergebnis $\forall x.\ c(x) \rightarrow d(x), r1(c, d)$ des ersten und dritten Ausdrucks ist nur innerhalb der Logik zweiter Stufe formulierbar.

[9] Natürlich können im Laufe eines Projektes Begriffe als Klassen ausgezeichnet werden.

dung von Begriffsnetzen gebildet werden. Somit sind nur statische Begriffskontexte in RDF und damit auch in OWL-DL(+Regeln) erlaubt.

F-Logic und TRIPLE hingegen erlauben eine beliebige Termbildung (siehe Abschnitt 2.1.1.1 auf Seite 9 und Abschnitt 2.2.3.1 auf Seite 53), womit sich explizite, dynamische Kontexte wie beispielsweise

ctx(HardwareNaheSprache, CPlusPlus)

bilden lassen. Allerdings unterstützen F-Logic und TRIPLE Anforderung III nur indirekt. Zum einen gibt es kein ausgezeichnetes Konstrukt zur expliziten, dynamischen Kontextbildung und zum anderen kein ausgezeichnetes Mittel zur Einschränkung der Kontextbildung auf taxonomische Kontexte.

Die Erfüllung von **Anforderung IV** ermöglicht es, den Begriffsraum spezifisch für Kategorisierungen festzulegen. Dies entspricht einer über die Angabe der Zielklasse hinausgehenden Wertebereichsbestimmung für Attribute, welche zur Kategorisierung vorgesehen sind. Eine Standardform ist die taxonomische Kategorisierung aus Abschnitt 3.1.4. Sie kann als Erweiterung der taxonomischen Kontextbildung aus Anforderung III angesehen werden und zwar derart, daß man sich nun auch auf bestimmte Taxonomien beschränken kann. Das entsprechende Konstrukt stellt eine Einbettung der in [BS02] vorgestellten Konzepte zur Kategorisierung von Artefakten aus dem Bereich Bildung und Beschäftigung [BNES03] in die hier vorgestellte integrative Ontologie-Beschreibungssprache dar.

RDF bietet nur teilweise eine Unterstützung von Anforderung IV. Natürlich lassen sich alle Begriffe, die zu dem ausgewählten Wertebereich eines kategorisierenden Attributs gehören sollen, als Instanzen einer entsprechenden Klasse vereinbaren. Jedoch ist die taxonomische Kategorisierung bei RDF eingeschränkt, da sie zwei Bedingungen an die Beschreibungssprache stellt: erstens muß der Wertebereich des Attributs aus Klassen bestehen können und zweitens muß der Wertebereich dynamisch in Abhängigkeit der Klassenbeziehungen (hier die Spezialisierungsbeziehung) definiert werden können. RDF erfüllt aufgrund des Fehlens einer Regelsprache nur die erste Bedingung. OWL-DL erfüllt aufgrund der Nichterfüllung von Anforderung II beide Bedingungen nicht. OWL-DL+Regeln hat zumindest den Vorteil, daß Wertebereiche eines Attributs dynamisch über Regeln bestimmt werden können. Allerdings dürfen die Wertebereiche nur Individuen enthalten.

Aufgrund der Tatsache, daß F-Logic und TRIPLE Anforderung II erfüllen und die Definition von Regeln zulassen, läßt sich eine Definition von Wertebereichen mit Hilfe von Regeln

vornehmen, welche die Semantik der taxonomischen Kategorisierung axiomatisiert. Das in F-Logic notierte Fragment

ITLehrgang [themen ⇒⇒ ITThemen].
forall *x x : ITThemen ← x :: ITKompetenz.*

realisiert die in Bild 3.6 auf Seite 73 intendierte Semantik des Konstrukts *taxonomic*. Allerdings gilt auch hier, das die Unterstützung eines derartigen Konstrukts nicht explizit angeboten wird.

So wie die Erfüllung von Anforderung III es erlaubt, Begriffe in einen Kontext zu setzten, ermöglicht die Erfüllung von **Anforderung V** nun, Beziehungen in einen Kontext zu stellen. Damit wird nicht nur eine einfache Modularisierung unterstützt, sondern auch die Relativierung von Beziehungen. Auf abstrakter Sprachebene bedeutet die Kontextualisierung den Übergang von Tripeln[10]

$(subject, predicate, object)$ zu Quadrupeln $(subject, predicate, object, context)$.

Dabei sind zwei Fälle zu unterscheiden: sind für $context$ nur Konstanten zugelassen, so entspricht das der einfachen Modularisierung beziehungsweise statischen Relativierung; sind die Quadrupel innerhalb von Regeln und für $context$ auch Variablen zugelassen, so entspricht das der dynamischen Relativierung (siehe Abschnitt 3.1.5).

RDF unterstützt die Anforderung teilweise, da die Aufteilung der Gesamtontologie mit Hilfe von unterschiedlichen RDF-Graphen nur dem ersten Fall gleichkommt. OWL-DL(+Regeln) besitzt ein einfaches Modulkonzept, welches als Modulbezeichner nur URIs zuläßt, was auch dem ersten Fall entspricht. Gegenüber RDF besitzt man bei OWL-DL(+Regeln) aufgrund von expliziten Modulbezeichnern zumindest den Vorteil, diese als *first class citizens* in anderen Beziehungen als Subjekt oder Objekt zu verwenden.

TRIPLE erfüllt die Forderung nach der Angabe von Beziehungskontexten in vollem Umfang. F-Logic unterstützt in der originalen Definition [KLW95] Anforderung V nur indirekt.[11] Jedoch sei bemerkt, daß jüngste Erweiterungen für die Ontobroker-Implementierung [DEFS99] das gleiche Modulkonzept wie TRIPLE (siehe Abschnitt 2.2.3.3) und damit diese Anforderung unterstützt.

[10] vergl. Fußnote 38 auf Seite 52.

[11] Indirekt insofern, daß sich der Kontext immer als letztes Argument eines Atoms mitführen läßt. Damit verläßt man jedoch die syntaktische Ebene der *object molecules*. FLORA2 [YKZ03] unterstützt zumindest die Modularisierung beziehungsweise die statische Relativierung.

Wie in Abschnitt 3.1.6 erläutert, sind Klassen-definierende Anfragen ein sehr starkes Ausdruckmittel, welches es erlaubt, mit Hilfe von Anfragen (bzw. allgemeiner mit Regeln) den Wertebereich von Attributen festzulegen. RDF erfüllt die entsprechende **Anforderung VI** nicht, da dort Wertebereiche von Attributen einzig über eine Konjunktion von *range*-Statements festgelegt werden können. OWL-DL läßt in eingeschränktem Maße komplexere Wertebereichsdefinitionen zu, weshalb es teilweise Anforderung VI erfüllt. Der Grund dafür ist die Möglichkeit zur Angabe von Klassenausdrücken als Zielklassen für Attribute (siehe Vokabular auf Seite 46). Beispielsweise läßt sich ein Teil des in Bild 3.8 unter b) dargestellten Modells in OWL-DL wie folgt formulieren:

IHKExpertenLehrgang ≡ ***all*** *(zertifiziertMit,* ***all*** *(ihkStufe, {schwer}))*

Denn der Wertebereich von *zertifiziertMit* wird damit als die Menge derjenigen Objekte definiert, dessen Attribut *ihkStufe* der Wert *schwer* zugewiesen ist.

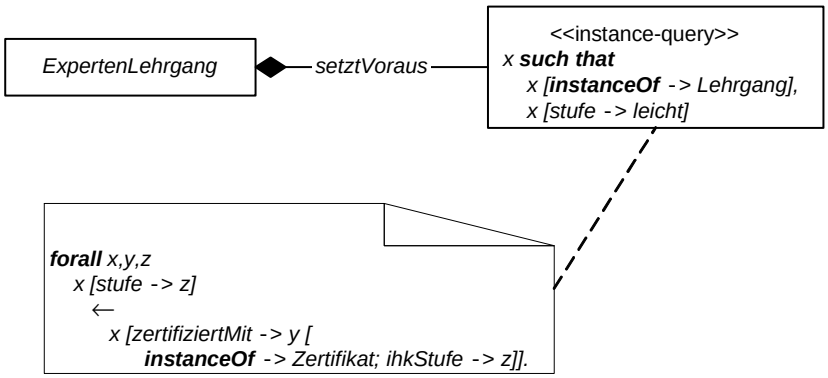

Bild 3.9: Klassen-definierende Anfragen mit Regeln

Allerdings beschränkt sich die dynamische Zielklassenbildung auf Basis von Attributwerten auf den Einsatz der DL-Konstrukte ***all*** und ***some***. Mit Hilfe von Regeln wären weitaus komplexere Bedingungen realisierbar. Man betrachte dazu den Modellausschnitt aus Bild 3.9. Er soll ausdrücken, daß ein Expertenlehrgang einen Lehrgang voraussetzt, der als leicht eingestuft ist. Die Einstufung eines Expertenlehrgangs erfolgt allerdings nicht über eine entsprechende Attributzuweisung, sondern ist als Regel definiert, die ausdrückt, daß alle Lehrgänge implizit die IHK-Stufe ihres zugeordneten Zertifikats besitzen. Dies ist mit der hier betrachteten OWL-DL aus Abschnitt 2.2.2.3 nicht ausdrückbar.[12]

Mit OWL-DL+Regeln ist man dahingegen in der Lage, die Klassen-definierende Anfrage aus Bild 3.9 mit dem folgenden Fragment auszudrücken.

ExpertenLehrgang ≡ ***all*** *(setztVoraus, LeichtLehrgang).*
all *(stufe, {leicht})* ⊑ *LeichtLehrgang.*
forall *x,y,z (x, z) : stufe* ← *(x, y) : zertifiziertMit, (y, z) : ihkStufe.*

Die Regel (siehe OWL-DL+Regeln in Abschnitt 2.3 auf Seite 60) in der dritten Zeile des Fragments definiert das Attribut *stufe* in Abhängigkeit von der Zertifikatsstufe, worüber in der zweiten Zeile die Zielklasse des *setztVoraus*-Attributs definiert wird.

Nicht nur die erläuterte Einschränkung bezüglich des *instance-query*-Konstrukts ist entscheidend, sondern auch die Tatsache, daß das Konstrukt *class-query* in OWL-DL(+Regeln) nicht ausdrückbar ist. Der Grund dafür ist, daß dieses Konstrukt eine Behandlung von Klassen als Instanzen erforderlich macht. Die weiter oben behandelte Nichterfüllung von Anforderung II für OWL-DL(+Regeln) verbietet jedoch ein derartiges Konstrukt.

F-Logic und TRIPLE unterstützen Anforderung VI indirekt, da mit Hilfe von Regeln die entsprechenden Konstrukte realisierbar sind. Beispielsweise axiomatisiert das in F-Logic notierte Fragment

```
WebWB :: Weiterbildung [ themen =>> WebThemen ].

forall y
    y : WebThemen  <-
        y :: x and
        (WebEngineering [ setztVoraus -> x ] or WebDesign [ setztVoraus -> x ]).
```

den in Bild 3.8 auf Seite 76 unter c) angegebenen Modellausschnitt.

[12] Es sei bemerkt, daß das hier aufgeführte Beispiel nur in einer DL mit Rollenkompositionen der Form $P \circ Q \sqsubseteq R$ (*zertifiziertMit* ∘ *ihkStufe* ⊑ *stufe*) ausdrückbar ist. Eine DL, die diese Form zuläßt, ist aber nicht entscheidbar [HPS04].

3.3 Axiomatisierung

Bevor in späteren Abschnitten eine präzise Definition der Semantik der Ontologie-Beschreibungssprache $\mathcal{I}$ vorgenommen wird, werden in diesem Abschnitt die der Semantikdefinition zu Grunde liegenden Mittel erörtert. Als Mittel zur Beschreibung der Semantik wird in der vorliegenden Arbeit die der Axiomatisierung mit einer logischen Sprache gewählt. Wenngleich die direkte modelltheoretische Semantik einige Vorteile bietet, wie z.B. die Implementierung einer maßgeschneiderten Inferenzkomponente,[13] wird hier der Axiomatisierung mit einer existierenden logischen Regelsprache der Vorzug gegeben, da

- sie direkt für eine prototypische Implementierung verwendet werden kann,
- Nachimplementierungen von bestehenden Optimierungen der Regelsprache vermieden werden können und
- im allgemeinen eine breitere Akzeptanz im Vergleich zu den modelltheoretischen Ansätzen vorherrscht.

Die eben genannten Gründe bezogen auf eine Realisierung beziehungsweise Akzeptanz sind aber nicht die einzigen. Weitere Gründe beziehen sich auf das Wesen Ontologie-Modellierungssprachen an sich. Denn

- bei allen Ontologie-Modellierungssprachen steht die logische Charakterisierung im Vordergrund und
- für all diese Sprachen existiert eine axiomatische Semantik in FOL (beziehungsweise einer Untermenge wie Datalog).

Die damit gegebene gemeinsame Basis fördert die hier angestrebte Integration von Teilen der vorgestellten Ontologie-Modellierungssprachen sowie der in Abschnitt 3.1 auf Seite 67 geforderten Konstrukte und zwar gemäß der in [GH03, Abschnitt 2] erörterten Vorteile gegenüber der modelltheoretischen Semantik: *„... we believe that a melange of model theories will adversely impact developers building agents that implement proof systems for these layers, since*

[13] vergl. [KLW95, Seite 83]: *„... a direct semantics for a logic suggests ways of defining proof theories tailored to that logic“*. Es sei aber bemerkt, daß beispielsweise für F-Logic alle bisherigen Implementierungen Sprachen aus dem Bereich deduktive Datenbanken [FHK+97] [SS04, Kapitel 2] beziehungsweise Prolog [YK00] als Zielsprache verwenden.

the proof systems will likely be different for each layer, resulting in the need to micro-manage small semantic variations for various dialects and sub-languages ...“. Wie [Dec02, Abschnitt 7] hervorhebt, kann die Vorgehensweise der Axiomatisierung mit der aus dem Bereich der sogenannten domänen-spezifischen Sprachen (*domain specific languages*) verglichen werden, welche dem gleichen Prinzip der Abbildung von domänenspezifischen Konstrukten auf eine bestehende Programmier- oder Spezifikationssprache folgt [vDKV00]. Weitere Beispiele für diese Vorgehensweise sind, neben der in [Dec02, Abschnitt 7], die Axiomatisierung im Rahmen einer OWL-DL-Regelsprache mit disjunktivem Datalog [MSS04], die Axiomatisierung von RDF und DAML-OIL mit KIF [FM01] und die von DLP mit Datalog [Vol04].

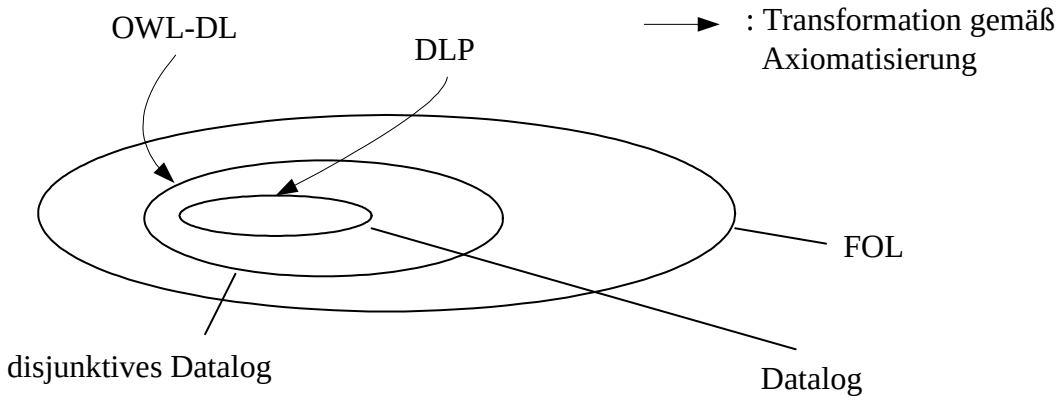

Bild 3.10: Transformation von OWL-DL und DLP

Bild 3.10 zeigt beispielhaft die Transformation von OWL-DL und DLP nach den entsprechenden logischen Sprachen.

Der hier vorgestellte Ansatz für eine Ontologie-Beschreibungssprache $\mathcal{I}$ soll eine Integration von DL-basierten Konstrukten und Konstrukten, die in den Anforderungen aus Abschnitt 3.1 erörtert wurden, umfassen. Aufgrund des ersten Punktes, nämlich der DL-Integration, und der Anforderung, daß Regeln zum Umfang von $\mathcal{I}$ gehören, sind für die Auswahl einer Axiomatisierungssprache die Ansätze aus Abschnitt 2.3 auf Seite 60 gegeben. Diese Auswahl (siehe dortige Tabelle 2.2) schränkt sich jedoch weiter ein. Zunächst einmal entfällt eine Betrachtung der ersten zwei Ansätze AL-Log und CARIN, da sie keinen wechselseitigen Zugriff des DL- und des Regel-Teils erlauben, wie es aber in Anforderung VI benötigt wird (siehe auch Beispiel Bild 3.9). Da so wie in F-Logic und TRIPLE non-DL-Prädikate definierbar sein sollen, beschränkt sich die Auswahl auf die letzten beiden Ansätze, nämlich disjunktives Datalog [MSS04] und Hornlogik mit integrity constraints [Vol04, HMS05].

Da Prolog-Systeme sehr weit verbreitet sind, eine hohe Praxisrelevanz besitzen, zahlreiche Optimierungen erfahren haben und sich durch hohe Performanz auszeichnen, wird als Ausgangspunkt der vorliegenden Arbeit bezogen auf die Axiomatisierungssprache der hornlogische Ansatz gewählt. Wie in [HHK+05, HSS06] zudem dargelegt, ist der damit axiomatisierbare DL-Umfang DLP für den praktischen Einsatz sehr gut geeignet.

3.3.1 Termbildung

Einige der in Abschnitt 3.1 aufgestellten Anforderungen an die Ontologie-Beschreibungssprache erfordern auf semantischer Ebene die Möglichkeit zur Termbildung, d.h. den Übergang von Datalog zur Horn-Logik. Denn Abschnitt 3.1.3 auf Seite 71 verlangt die Repräsentation von Kontextpfaden zum Zwecke der dynamischen Kontextbildung wie beispielsweise

Feldherr / AlexanderDerGroße oder *HardwareNaheSprache / CPlusPlus*

Die klassischen DL- und RDF-orientierten Ansätze verfügen nicht über eine solche Möglichkeit, sondern lassen lediglich nullstellige Terme zur Identifikation von Ressourcen zu, was eine einfache Abbildung auf Datalog (nach entsprechender Auflösung der Skolemfunktionen) gestattet.

Der gleiche Sachverhalt besteht bei Anforderung V auf Seite 75, welche für Transformationen notwendig ist. Beispielsweise definiert das TRIPLE-Fragment

forall *x, y, z, u* $\quad$ *x [y -> z] @ f(u)* $\quad\leftarrow\quad$ *x [y -> z] @ u* ***and*** B

die Transformation von statements im Kontext *u* nach dem Kontext *f(u)* in Abhängigkeit von der Bedingung B (vergleiche Abschnitt 2.2.3.3 auf Seite 56 über parametrisierbare Kontexte). Dies erfordert zumindest bei einer uneingeschränkten Angabe von Beziehungskontexten die Bildung von zusammengesetzten Termen auf der Ebene der Axiomatisierungssprache.

3.3.2 Höherstufigkeit der Logik

Wie in Kapitel 2 öfters bemerkt und im Abschnitt zur F-Logic und zu RDF motiviert, ist es erforderlich, Klassen und Attribute als *first class citizens* zu behandeln. Insbesondere für die Erfüllung von Anforderung II auf Seite 70 ist dies notwendig. Damit muß die Axioma-

tisierungssprache die Möglichkeit zulassen, über diese Basisbausteine zu quantifizieren, was jedoch nicht direkt mit Hilfe der Logik erster Stufe ausdrückbar ist.

Die Verwendung der Logik zweiter Stufe ist wegen ihrer wohlbekannten Unvollständigkeit ausgeschlossen. Daher wird hier der bei F-Logic und TRIPLE[14] verwendete Ansatz gewählt, nämlich eine Zweistufigkeit auf syntaktischer Ebene unter Beibehaltung der Einstufigkeit auf semantischer Ebene [CKW93].

Im Kern handelt es sich bei den Ansätzen um eine spezielle Kodierung von atomaren Formeln:[15]

$$p(t_1, \ldots, t_n) \quad \text{wird kodiert mit} \quad apply(p, t_1, \ldots, t_n) \ .$$

Damit ergibt sich allerdings zunächst eine syntaktische Unverträglichkeit mit der gewählten Basis [Vol04, HMS05] für die Axiomatisierungssprache, welche die Integration von DL gewährleistet, da dort die Instanzbeziehung von x zur Klasse A und das Setzen der Beziehung R zwischen x und y mit den Formeln

$$A(x) \ \text{resp.} \ R(x, y) \quad \text{und nicht mit} \quad apply(A, x) \ \text{resp.} \ apply(R, x, y)$$

repräsentiert wird. Eine entsprechende semantikerhaltende Kodierung des DL-Anteils zum Zwecke der Integration wird in Abschnitt 4.2 behandelt.

3.3.3 Negation

Zunächst verfügt die Basis für die Axiomatisierungssprache nicht vollständig über explizite Konstrukte für die Negation. Constraints, d.h. Hornformeln mit leerer Konklusion, erlauben jedoch die Ausdrückbarkeit der für DL notwendigen klassischen Negation in eingeschränktem Maße. Die klassische Negation ist für viele Fragestellungen in Bezug auf Ontologien nicht ausreichend. Eine für die Praxis wesentlich relevantere Art der Negation ist die sogenannte *negation as failure*. Bei dieser Art der Negation, wie sie auch im Datenbank-Kontext üblich ist, werden alle nicht ableitbaren Fakten als nicht wahr interpretiert.[16]

[14] Zur Erörterung der Möglichkeiten im Zusammenhang mit einem Vorgänger von TRIPLE, nämlich SILRI, siehe [Dec02].

[15] Zur analogen TRIPLE-Kodierung mit Hilfe des Funktors $true$ siehe Abschnitt 2.2.3.4. Die in der Arbeit vorgestellte Ontologie-Beschreibungssprache folgt diesem hier vereinfacht dargestellten Prinzip.

[16] Dieser Interpretation liegt die sogenannte *Closed World Assumption* zu Grunde (siehe [Llo87]).

Insbesondere im Kontext derjenigen Ontologien, die sich als geschlossene Welten verstehen, ist diese Art der Negation besonders wichtig. Betrachte man dazu den Ausdruck

klaus [hobby –> theater] .

Will man erfragen, ob

***not** klaus [hobby –> handball]*

gilt, so erhält man bei der Interpretation von ***not*** mit der klassischen Negation keine Antwort, außer man würde alle existierenden Hobbys für Klaus als negative Information hinzufügen. Diese Art Ausdrücke folgen dem Prinzip, über Fakten Mengen zu definieren. Alle nicht zu einer Menge gehörenden Elemente als negative Fakten aufzuzählen, wäre kontra-intuitiv und in der Praxis schwerlich umzusetzen. Außerdem spielen Constraints, d.h. Regeln mit fehlendem Kopf, zusammen mit der *negation as failure* eine wichtige Rolle. Um z.B. sicherzustellen, daß Klaus mindestens ein Hobby besitzt, würde man das Constraint

***forall** x ← **not** klaus [hobby –> x]*

hinzufügen.

Wie in [BD98] dargelegt, stehen im wesentlichen vier Ansätze für die Semantik der nichtmonotonen Negation zur Verfügung: *negation as finite failure*, *Perfect Model Semantics*, *Stable Model Semantics* und *Well-founded Semantics*. Da *negation as finite failure* und die *Perfect Model Semantics* keine uneingeschränkte Zyklizität bezogen auf die Prädikate der Formelmenge zuläßt, was jedoch auf Grund der in Abschnitt 3.3.2 geforderten Kodierung nicht vermeidbar ist,[17] verbleiben noch die Ansätze *Stable Model Semantics* und *Well-founded Semantics*. Da die *Stable Model Semantics* ein *bottom-up* Ableitungsverfahren erzwingt, wird für die vorliegende Arbeit die *Well-founded Semantics* für die Negation (kurz *wfs-Negation*) verwendet [vGRS91].

Eine bedingungslose Einführung der wfs-Negation führt jedoch zu Problemen der Nutzung der Sprache im Bereich des *Semantic Web*. Aufgrund der hohen Dynamik und Unvollständigkeit im Gegensatz zu Datenbankanwendungen, also der Offenheit des WWW gilt eher die *Open World Assumption*, das heißt, es kann nicht sofort die Falschheit einer Aussage angenommen

[17] Man beachte die Reduzierung auf das Prädikat $apply$.

werden, wenn die Gültigkeit dieser Aussage nicht abgeleitet werden kann. Eine Lösung des Problems, die beiden gegensätzlichen Annahmen zu verbinden, ist die sogenannte *scoped negation*[18], wie sie in [HTdSM05, KdBBF05] vorgeschlagen wird und hier als Basis verwendet wird. Dahinter verbirgt sich Einführung von lokalen Welten, wie sie in TRIPLE und FLORA2 [YKZ03, SD02] vorgenommen wurden. Grundsätzlich gelten hierbei Aussagen nur in einem bestimmten Kontext (vergl. Beziehungskontext in Abschnitt 3.1.5), so auch die Negation. Ist beispielsweise die Aussage

$$A = \quad \textit{standardKollektion [enthält -> ISBN-0521-63028-2] @ bücherei1187}$$

nicht ableitbar, so gilt ***not*** A für den Kontext *bücherei1187*, man betrachtet also lediglich die ‚lokale Welt' *bücherei1187* als abgeschlossen.

3.3.4 Entscheidbarkeit

Eine der wesentlichen Anforderungen an die Axiomatisierungssprache ist die der Entscheidbarkeit. Genauer: Die Transformation eines Satzes der Ontologie-Beschreibungssprache in einen Satz der Axiomatisierungssprache sollte nicht dazu führen, daß Anfragen eine Nicht-Termination des Ableitungsverfahrens bewirken. Der Grund dafür ist der deklarative Charakter von Ontologie-Beschreibungssprachen. Während bei der logischen Programmierung der Entwickler selber dafür sorgen muß, terminierende Programme zu schreiben, dafür aber auf ausdrucksstärkere Konstrukte zurückgreifen kann, muß im Kontext von Ontologien es möglich sein, rein deklarativ die logischen Zusammenhänge zu definieren.

Ein Beispiel dafür ist die Verwendbarkeit der in Anforderung VI auf Seite 77 geforderten klassendefinierenden Anfragen. Denn während bei der Entwicklung einer Anwendung unter Umständen der Entwickler auf Termination achten kann, ist die Klassendefinitionsebene von ihrem Wesen her deklarativ, wie die Definition des Attributs *befähigtZu* der Klasse *Berufsausbildung* aus Abschnitt 3.1.6 verdeutlicht. Die Zielklassen werden definiert als die Klassen, die Instanzen einer akademischen Qualifizierungsart sind. Will man nun die Extension der Attributdefinition innerhalb der Modellierungsphase überprüfen, so muß für die praktische Anwendung sichergestellt sein, das diese Berechnung terminiert, d.h. die Anfrage nach diesen Klassen entscheidbar ist.

Die als Basis gewählten Ansätze DLP/Horn-$\mathcal{SHIQ}$ sind entscheidbar [Vol04, HMS05]. Bei

[18] Entsprechend könnte man von der *Scoped World Assumption* sprechen.

DLP liegt dies im wesentlichen an der Beschränkung des DL-Teils. Bei Horn-$\mathcal{SHIQ}$ sorgt die DL-*safety*-Eigenschaft für die Entscheidbarkeit. Die im letzten Abschnitt hinzukommende Negation unter der *Well-founded Semantics* zerstört diese Eigenschaft nicht. In Kapitel 4 wird diese Basis zur Unterstützung der in diesem Kapitel aufgestellten Anforderungen (siehe z.B. Abschnitt 3.3.1) erweitert. Dort erfolgt auch die Sicherstellung der Entscheidbarkeit.

3.3.5 Gleichheit

Bei der in Abschnitt 2.1.2 eingeführten DL ist es möglich, die Gleichheit von zwei Individuen auszudrücken. Das dazu notwendige Gleichheitsprädikat stellt eigentlich keine besondere Problematik dar, da es, wenn nicht als *build-in* vorhanden, mit zusätzlichen Regeln axiomatisiert werden kann. Die Nutzung des Gleichheitsprädikats hat allerdings für die praktische Anwendung den entscheidenden Nachteil, daß sich die Performanz durchschnittlich um den Faktor 100 verschlechtert [Vol04, Kapitel 8].

Zum anderen führt die Anforderung, beliebige Pfade zur Angabe von Begriffskontexten zuzulassen, im Zusammenhang mit der Gleichheit zu Entscheidbarkeitsproblemen. Der Grund dafür liegt an den Gleichheitsaxiomen[19] selber. Denn das Substitutivitätsaxiom

$$x_1 = y_1 \wedge x_2 = y_2 \rightarrow x_1/x_2 = y_1/y_2$$

für den Begriffskontext-Operator / könnte unendliche viele Fakten der Form $r_1/r_1/\ldots = r_2/r_2/\ldots$ für zwei Konstanten mit $r_1 = r_2$ folgen lassen. Eine entsprechende Abschwächung wird daher später vorgenommen.

Wegen dem Nachteil bezüglich der Performanz wird auf ein in der Sprache implizit vorhandenes Gleichheitskonstrukt verzichtet und für die hier vorgestellte Ontologie-Beschreibungssprache die sogenannte *unique name assumption* (UNA) favorisiert. Sie nimmt an, daß verschiedene Namen immer mit ungleichen Individuen interpretiert werden. Deshalb wird hier die Gleichheit mit der Unifizierbarkeit ausgedrückt. Das bedeutet, daß die Gleichheit über der Termalgebra verwendet wird. Damit ist gegeben, daß zwei Kontextpfade, deren Ressourcen an einer Position ungleich sind, unterschiedlich interpretiert werden.

Die Favorisierung der UNA für die in dieser Arbeit vorgestellten Ontologie-Beschreibungssprache folgt der Priorisierung der im Bereich der Datenbanken und des Software-Engineering üblichen Sicht, in der unterschiedliche Namen auch verschiedene Individuen bezeichnen.

[19] Dies sind die üblichen Axiome zur Reflexivität, Symmetrie, Transitivität und Substitutivität der Gleichheit.

Anders verhält es sich bei der im Bereich des *Semantic Web* üblichen Sicht. Dort können verschiedene Namen durchaus identische Individuen bezeichnen. Die unterschiedlichen Sichten werden in [dBLPF05], das die erstere favorisiert, und in [PSH06] als Vertreter der letzteren diskutiert.

So wie auch bei der Behandlung der Negation (siehe Abschnitt 3.3.3), bei der nicht uneingeschränkt die *Closed World Assumption* angenommen werden kann, ist die strikte Einschränkung auf die UNA aufgrund der hohen Dynamik und Unvollständigkeit des *Semantic Web* im Gegensatz zu Datenbankanwendungen nicht ratsam. Jedoch ist auch hier eine Lokalisierung sinnvoll[20], welche wiederum mit Hilfe des Beziehungskontextes ausdrückbar ist. Wird innerhalb eines Kontextes nicht von der UNA ausgegangen, so können zusätzlich zu den Regeln für die syntaktische Gleichheit die Regeln für die Gleichheitsaxiome diesem Kontext hinzugefügt werden. Diese Möglichkeit wird in der hier vorgestellten Ontologie-Beschreibungssprache realisiert.

[20] Im OWL Bereich wird auch von der *Local Unique Name Assumption* gesprochen [Vra05].

4 Ein integrativer Ansatz für eine Ontologie-Beschreibungssprache

4.1 Syntax

4.1.1 Regelsprache

Im folgenden wird die Syntax der Regelsprache von $\mathcal{I}$ eingeführt.

Definition 4.1. ***(Alphabet)*** Das Alphabet der Sprache $\mathcal{I}$ besteht aus einer Menge von URIs $\mathbf{R}$, genannt *Ressourcen*, einer Menge von *Literalen* $\mathbf{L}$, einer Menge von *Funktionssymbolen* $\mathbf{F}$, einer Menge von *Prädikatensymbolen* $\mathbf{P}$ und einer unendlichen Menge von *Variablen* $\mathbf{V}$. Ferner umfaßt das Alphabet die Menge der *Signaturkonstrukte* $\mathbf{S} = \{$***domain***, ***range***, ***taxonRange***, ***mandatory***, ***functional***$\}$, und die Symbole *::* , *:*, *->*, ***is*** und */* sowie die üblichen Klammer- und Interpunktionssymbole. $\mathbf{R}$, $\mathbf{L}$, $\mathbf{F}$, $\mathbf{P}$, $\mathbf{V}$ und $\mathbf{S}$ sind paarweise disjunkt. □

Für die Mengen $\mathbf{R}$ und $\mathbf{L}$ gelten die gleichen Eigenschaften wie die der RDF-Definition für *URI references* resp. (ungetypte) Literale [KC04]. Die folgende Definition führt die in $\mathcal{I}$ verwendeten Terme ein.

Definition 4.2. ***(Terme)*** Die mit BT bezeichnete Menge der *Beziehungskontext-Terme* (kurz *B-Terme*) und die mit RT bezeichnete Menge der *Ressource-Terme* (kurz *R-Terme*) sind wie folgt induktiv definiert.

(i) Jede Ressource und Variable ist ein B-Term. Sind $t_1, \ldots, t_n$ B-Terme ($n > 0$) und f ein Funktionssymbol, so ist $f\ (\ t_1, \ldots, t_n\ )$ ein B-Term.

(ii) Jede Ressource ist ein R-Term. Ist r eine Ressource und t ein R-Term, in dem r nicht vorkommt, so ist */(t, r)* ein R-Term, auch *Kontextpfad* genannt.[1] □

4.2.(i) entspricht der üblichen Termbildung. Mit B-Termen werden die Beziehungskontexte festgelegt. Mit den mit / gebildeten Kontextpfaden werden die begrifflichen Kontexte repräsentiert. Man beachte, daß die in 4.2.(ii) erzwungene (flache) Listenform verhindert, daß Elemente eines Kontextpfades zusammengesetzte R-Terme sein können. Ferner enthalten R-Terme keine Variablen. Die nachstehende Definition führt die atomaren Formeln von $\mathcal{I}$ ein.

Definition 4.3. ***(Atome)*** *Atome*, auch *atomare Formeln* genannt, sind wie folgt definiert.

(i) Ist $s, p \in \mathbf{R} \cup \mathbf{V}$, $o \in RT \cup \mathbf{L} \cup \mathbf{V}$ und $b \in BT$, so ist

[1] Im weiteren kann / auch als Infixoperator verwendet werden, der per definitionem linksassoziativ ist. Man beachte, daß per definitionem Kontextpfade immer unterschiedliche Ressourcen beinhalten und der Länge > 1 sind. Aus technischen Gründen wird eine Ressource nicht als Kontextpfad (der Länge 1) betrachtet.

s *[* p *->* o *]@* b

ein *Atom*, genannt *Beziehungsatom*.

(ii) Sei $X = \mathbf{S} \setminus \{$***domain***$\}$. Dann ist ist $\hat{\mathbf{S}}$ definiert durch $x(c) \in \hat{\mathbf{S}}$, wobei $x \in X$ und $c \in \mathbf{R} \cup \mathbf{V}$. Ist $s \in \mathbf{R} \cup \mathbf{V}$, $p \in \mathbf{S} \cup \hat{\mathbf{S}}$, $o \in \mathbf{R} \cup \mathbf{L} \cup \mathbf{V}$ und $b \in BT$, so ist

s *[* p ***is*** o *]@* b

ein *Atom*, genannt *Signaturatom*. Falls p gleich ***mandatory*** oder ***functional***, so muß o gleich *true* sein, ansonsten aus $\mathbf{R} \cup \mathbf{V}$.

(iii) Sind $t_1, t_2 \in \mathbf{R} \cup \mathbf{V}$ und ist $b \in BT$, so sind

t_1 *:* t_2 *@* b ***und*** t_1 *::* t_2 *@* b

Atome, genannt *isa-Atome*. Ersteres bezeichnet man als *Instantiierungsatom*, letzteres als *Spezialisierungsatom*. Signatur- und Spezialisierungsatome werden *Schemaatome* genannt.

(iv) Ist $t \in \mathbf{P} \cup \mathbf{V}$, sind $t_1, \ldots, t_n \in \mathbf{R} \cup \mathbf{L} \cup \mathbf{V}$ mit $n \geq 0$ und ist $b \in BT$, so ist

t *(* t_1*,...,*t_n *)@* b

ein *Atom*, genannt *P-Atom*.

(v) Ist $t_1, t_2 \in \mathbf{R} \cup \mathbf{L} \cup \mathbf{V}$ und $b \in BT$, so ist

equal *(* t_1*,* t_2 *)@* b

ein *Atom*, genannt *Gleichheitsatom*. □

Beziehungsatome dienen der wie in F-Logic und TRIPLE gebräuchlichen Beziehungssetzung. Atome umfassen die üblichen prädikatenlogischen Atome, wenngleich in erweiterter Form. Diese Form, nämlich die Zulassung von Variablen an Prädikatsnamenstelle, führt auf der Ebene von $\mathcal{I}$ die in Abschnitt 3.3.2 geforderte Höherstufigkeit ein. Ferner umfassen Atome die ausgezeichneten Beziehungssetzungen der Instantiierung resp. Spezialisierung. Die folgende Definition führt nun Regeln ein.

Definition 4.4. ***(Regeln)*** Jedes Atom A und der Ausdruck $($***naf*** $A)$ sind *Rumpfatome*. $($***naf*** $A)$ wird *negatives Atom* genannt. Sind $A_1, \ldots, A_n$ Atome, $B_1, \ldots, B_m$ *Rumpfatome* $(n, m \geq 0)$, wobei $n = m = 0$ nicht gilt, und kommen alle Variablen aus $A_1, \ldots, A_n$ in mindestens einem B_i vor, so ist

$$A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m \; .$$

eine *Regel*. A_i wird *Kopfatom* genannt. Ist $m = 0$, so spricht man von einem *Faktum*. Ist $n = 0$, so spricht man von einem *Constraint*. Eine Regel wird *definit* genannt, falls sie keine negativen Atome enthält. Eine Menge M von Regeln heißt *den Kontext k bestimmend* (kurz *k-bestimmend*), falls jedes Kopfatom aus M die Form X *@* k hat ($k \in \mathbf{R}$). M heißt *kontextkonstant*, falls alle in ihr vorkommenden B-Terme aus $\mathbf{R}$ sind. □

Bemerkung 4.5. (Konventionen / syntaktische Abkürzungen) Wenn eine Unterscheidung erforderlich ist, wird im folgenden auch von $\mathcal{I}$-Termen, $\mathcal{I}$-Atomen, $\mathcal{I}$-Regeln usw. gesprochen. Der besseren Lesbarkeit wegen steht *s[p]* für *s[p≈>true]*, wobei ≈> gleich –> oder ***is***. Für die Bildung von Blöcken werden die Konventionen aus Bemerkung 2.1 auf Seite 54 übernommen, wobei *flatten* für Signatur-, isa-, P- und Gleichheitsatome entsprechend erweitert wird. Dabei entsprechen B-Terme den dortigen Kontextausdrücken. Fehlt bei einem Atom einer Regel nach Anwendung von *flatten* der Kontextteil *@* b , so wird von *kontextlosen Atomen* gesprochen. Eine Regel nur mit kontextlosen Atomen heißt *kontextlose Regel*. Auf eine Einführung der Abkürzungen gemäß der wohlbekannten Lloyd-Topor-Transformation [Llo87], mit Ausnahme der für *flatten* benötigten Konjunktion im Regelkopf, wird hier verzichtet. Da die Variablenmenge $\mathbf{V}$ disjunkt zu allen anderen Alphabetsmengen ist, hat das in den folgenden Beispielen und anderen Programmfragmenten eine Regel einleitende ***forall***-Konstrukt nur dokumentierenden Charakter. Wie üblich steht $A \leftrightarrow B$ für $A \leftarrow B$ und $B \leftarrow A$.

4.1.1.1 Signaturkonstrukte

Das klassische Signatur-Constraint *p [**range is** a]* erzwingt bei Zuweisungen *s [p –> o]*, daß *o* in Instanzbeziehung zu *a* steht. Wie jedoch in Abschnitt 3.1.4 motiviert, ist das klassische Signatur-Constraint nicht geeignet für kategorisierende Eigenschaften, bei denen gefordert ist, daß *o* in Spezialisierungbeziehung zu *a* stehen soll. Genau dieses Constraint wird mit dem Konstrukt ***taxonRange*** unterstützt, das der semantischen Bedingung

falls p *[* ***taxonRange is*** a *]* und s *[* p –> o *]* , dann $o :: a$

entspricht.

Bild 3.6 auf Seite 73 zeigt eine solche Anwendung in diagrammatischer Form. In Fragment 4.1 ist die entsprechende Repräsentation mit einem $\mathcal{I}$-Block gezeigt.[2] Taxonomische Wertebe-

```
@ Bildung {

    ITLehrgang :: Lehrgang.
    themen [ domain is ITLehrgang ;
             taxonRange is ITKompetenz ;
             mandatory ].

    L1474 : Lehrgang.
    L1474 [ themen -> {UML, Java} ].
}
```

Fragment 4.1: Signaturkonstrukte

reiche lassen sich insbesondere als explizite Verbindung der klassenorientierten Sicht mit der Sicht auf das Modell als taxonomische Begriffsstruktur ansehen. Die Konstrukte zur Kardinalitätsangabe entsprechen den üblichen Konventionen. Auf sie wird noch in Abschnitt 4.1.2.1 eingegangen.

Mit Hilfe des ***domain***- und ***range***-Konstrukts lassen sich globale Eigenschaftsdefinitionen vornehmen, so wie es auch im RDF-Standard möglich ist. Zusätzlich sind in Definition 4.3.(ii) für die Signaturkonstrukte ***mandatory***, ***functional***, ***taxonRange*** und ***range*** ein optionales Argument zugelassen. Mit Hilfe dieser Form lassen sich Eigenschaften lokal zu einer Ressource definieren, wie es in der objektorientierten Programmierung bei Attributen und in F-Logic bei *method expressions* üblich ist. Definiert man beispielsweise

```
p [ range(A) is B;
    range(C) is D ].
```

so ist der Wertebereich von *p* für Instanzen von *A* gleich *B* und für Instanzen von *C* gleich *D*. Dies wird durch die folgende semantische Bedingung gewährleistet.

falls p *[* ***range****(c)* ***is*** a *]*, s *[* p *->* o *]* und $s : c$, dann $o : a$.

Man beachte, daß lokale Eigenschaftsdefinitionen in RDF nicht möglich sind. In F-Logic wie-

[2] Der besseren Lesbarkeit wegen wird bei diesem und den nachfolgenden Beispielen auf die vollständige Form (namespace, local name) für Ressourcen verzichtet. Die vollständige Form und deren Abkürzungen über das RDF namespace Konzept ist in [Bec04b] detailliert erörtert.

derum sind nur nichtoptionale lokale Eigenschaften ausdrückbar.

4.1.2 Ontologiedeklarationen

Die im letzten Abschnitt eingeführten Konstrukte würden zunächst ausreichen, um eine Deklaration von Ontologien mit Hilfe von Regeln vorzunehmen. Alle wesentlichen Basiskonstrukte zur Bildung von Klassen und Individuen sowie Regeln zur Festlegung der vom Nutzer intendierten Semantik sind vorhanden. Wenn man jedoch den Abstraktionsgrad und die Verständlichkeit einer ‚reinen' Definition über Regeln mit dem der Diagramme aus Abschnitt 3.1 betrachtet, so ist eine abstraktere syntaktische Form vorzuziehen. Als Vorbilder hierfür sind die Ansätze für eine Syntax aus [PSHH04, ABdB+05, LdBPF05] aufzuführen.

Demgemäß führt die nachstehende Definition Konstrukte zum Aufbau von Ontologien ein, welche neben klassischen Konstrukten die in den Anforderungen aus Abschnitt 3.1 geforderten Konstrukte umfaßt.[3]

Definition 4.6. ***(Ontologiedeklarationen)*** Eine *Ontologiedeklaration* ist ein Satz der durch die EBNF-Produktionsregeln (1)-(10) definierten Sprache mit dem Startsymbol $ontology$.

(1)	$ontology$	::=	***ontology*** id `{` $resource*$ $rule*$ `}`
(2)	$resource$	::=	id isa $((\textbf{\textit{as}}\ (global\text{-}property \mid class))* \mid \textbf{.})$
(3)	isa	::=	$(\textbf{:}\ id\text{-}list)?$ $(\textbf{::}\ id\text{-}list)?$
(4)	$global\text{-}property$	::=	***property domain*** $id\text{-}list$ $range\text{-}part$ **.**
(5)	$class$	::=	***class*** `{` $local\text{-}property*$ `}`
(6)	$local\text{-}property$	::=	***property*** id $range\text{-}part$ **.**
(7)	$range\text{-}part$	::=	$cardinality$ ***taxon***? ***range*** $(id\text{-}list \mid class\text{-}def\text{-}query)$
(8)	$cardinality$	::=	***mandatory***? ***functional***?
(9)	$class\text{-}def\text{-}query$	::=	$(\textbf{\textit{classQuery}} \mid \textbf{\textit{instanceQuery}})$ `{` vid / $p\text{-}atom$ `}`
(10)	$id\text{-}list$	::=	id $(\textbf{,}\ id)*$

id und vid umfassen die Elemente aus $\mathbf{R}$ resp. $\mathbf{V}$. $rule$ ist eine Regel mit kontextlosen Kopfatomen, für deren Rumpfatome gilt, daß sie entweder kontextlos sind oder einen Beziehungskontext aus $\mathbf{R}$ besitzen (siehe Bemerkung 4.5). $p\text{-}atom$ ist ein kontextloses P-Atom, in dem

[3] Dazu wird die EBNF-Notation verwendet, bei der wie üblich $()$, $|$, $?$, $*$ die Gruppierung, Alternative, Optionalität resp. optionale Liste kennzeichnet.

vid enthalten sein muß. Die Menge der in (2) verwendeten Ressource-Bezeichner und die Menge der in (6) verwendeten müssen disjunkt sein. □

Bemerkung 4.7. Enthalten die in Ontologiedeklarationen enthaltenen Regeln in ihren Köpfen keine Schemaatome, so ist entspricht dies der Einschränkung auf statische Schemata wie z.B. bei relationalen Datenbanken, d.h. Schemata, die sich durch Hinzufügen von Regeln mit Nicht-Schemaatomen im Kopf nicht verändern.

4.1.2.1 Basiselemente

Fragment 4.2 zeigt eine Repräsentation des Diagramms aus Bild 3.2 und des Begriffsnetzes in Bild 3.1.a) als Ontologiedeklaration.

```
ontology Bildung {

    Qualifizierungsart as class {
        property akadem mandatory functional range Literal.
    }
    Studium : Qualifizierungsart.
    Studium [ akadem -> true ].

    Lehrgang : Qualifizierungsart as class {
        property ort mandatory functional range Adresse.
        property zertifiziertMit range Zertifikat.
    }
    Lehrgang [ akadem -> false ].

    L1474 : Lehrgang.
    L1474 [ zertifiziertMit -> IHK-Z-09; ort -> R47 ].

    Lehrgang [ vermittelt -> Kompetenz ].
    IT-Kompetenz :: Kompetenz.
    UML :: IT-Kompetenz.
    Java :: IT-Kompetenz.
    UML [ unterstützt -> Java ].
}
```

Fragment 4.2: Basiselemente für Ontologiedeklarationen

Das Fragment führt alle dort eingeführten Ressourcen ein. Während die im Bereich Kompetenz gegebenen Ressourcen nicht näher spezifiziert sind, sind die aus dem Klassendiagramm

stammenden Ressourcen als Klassen mit ihren Eigenschaftsdefinitionen festgelegt (***as class***). Man beachte, daß die Klassen-Deklarationen lediglich Signatur-Constraints darstellen. Für die Zuweisung an die Eigenschaft *unterstützt* existieren keine solchen constraints.

Auf eine Unterstützung von exakten Kardinalitäten ≥ 1 (wie z.B. (3..6) etc.) wird wegen der Anlehnung an $\mathcal{L}_0^{IC}$-DLPs verzichtet (vergleiche Tabelle 2.3 auf Seite 65). Die Gegenüberstellung der üblichen sonstigen Kardinalitäten gemäß der UML-Notation ist wie folgt festgelegt:

1..1 ***mandatory functional***
*1..** ***mandatory***
0..1 ***functional***
*0..** default .

Ressourcen können sowohl in der Rolle als Beziehung als auch in der Rolle als Klasse festgelegt werden. Beispielsweise definiert das Fragment

```
leihtAus
    as property
        domain Kursteilnehmer range Kursmaterial.
    as class {
        property datum range Literal.
    }
```

die mit dem Ausleihdatum attributierte Ausleihbeziehung. Die beschriebenen Basiselemente realisieren zum einen die Integration von extensionsorientierten Klassenkonstrukten und von in benannter Beziehung stehenden Begriffen und zum anderen ermöglichen sie Instanzen über mehrere Stufen hinweg (siehe Anforderung I und II in Tabelle 3.1 auf Seite 78).

4.1.2.2 Komplexe Wertebereiche

Eine weitere Art, komplexe Wertebereiche festzulegen, ist die der klassendefinierenden Anfragen aus Bild 3.8 auf Seite 76. Fragment 4.3 zeigt eine entsprechende Erweiterung der Ontologiedeklaration um die Teile a) und b) aus Bild 3.8.

Der in der ersten Eigenschaftsdefinition angegebene Wertebereich ***classQuery*** *{x / Q(x)}* steht für eine neu konstruierte Klasse, welche all die Ressourcen x als Unterklasse besitzt, für die *Q(x)* gilt. Die darunter stehende Regel legt fest, wann für eine Ressource *Q* gilt, nämlich falls

```
ontology Bildung {

    Berufsausbildung : Qualifizierungsart as class {
        property befähigtZu range classQuery {x / Q(x)}.
    }
    IHKExpertenLehrgang :: Lehrgang as class {
        property zertifiziertMit range instanceQuery {x / Z(x)}.
    }
    forall x  Q(x) ← x : Qualifizierungsart, x [ akadem -> false ].
    forall x  Z(x) ← x : Zertifikat, x [ ihkStufe -> schwer ].
}
```

Fragment 4.3: Komplexe Wertebereiche

sie eine nichtakademische Qualifizierungsart ist, was z.B. für eine Klasse *Lehrgang* gilt. Somit können Eigenschaftszuweisungen an *befähigtZu* nur mit Instanzen von *Lehrgang* oder anderen das Prädikat *Q* erfüllenden Klassen vorgenommen werden. Die entsprechende Auflösung des Konstrukts ist gegeben mit

(***property*** p ***domain*** a ***range classQuery*** *{*x */* $P(\ldots,x,\ldots)$*}*) steht für $(*)$
(***property*** p ***domain*** a ***range*** *T*) , ***forall*** x x *::* *T* $\leftrightarrow$ $P(\ldots,x,\ldots)$.

Der in der zweiten Eigenschaftsdefinition aus Fragment 4.3 angegebene Wertebereich ***instanceQuery*** *{x / Z(x)}* steht für eine neu konstruierte Klasse, welche alle Ressourcen x als Instanzen besitzt, für die *Z(*x*)* gilt. *Z* gilt für alle Zertifikate der Stufe *schwer*. Die Menge dieser Zertifikate ist nun der Wertebereich der Eigenschaft *zertifiziertMit*.

Während bei dem ***classQuery***-Konstrukt der neue Wertebereich über die Teilmengenbeziehung festgelegt wird, ist der Wertebereich beim ***instanceQuery***-Konstrukt über die Instanzbeziehung definiert, was zur leicht modifizierten Auflösung führt: es erfolgt eine Ersetzung der Äquivalenz in Bedingung $(*)$ durch ***forall*** x x *:* *T* $\leftrightarrow$ $P(\ldots,x,\ldots)$. Man beachte, daß erst die syntaktische Höherstufigkeit der Sprache klassendefinierende Anfragen mit Hilfe von ***classQuery*** möglich macht, da erst in diesem Fall Klassen als Individuen mit Eigenschaften erlaubt sind. Bild 4.1 zeigt nochmals den Unterschied bezogen auf die Extensionen.

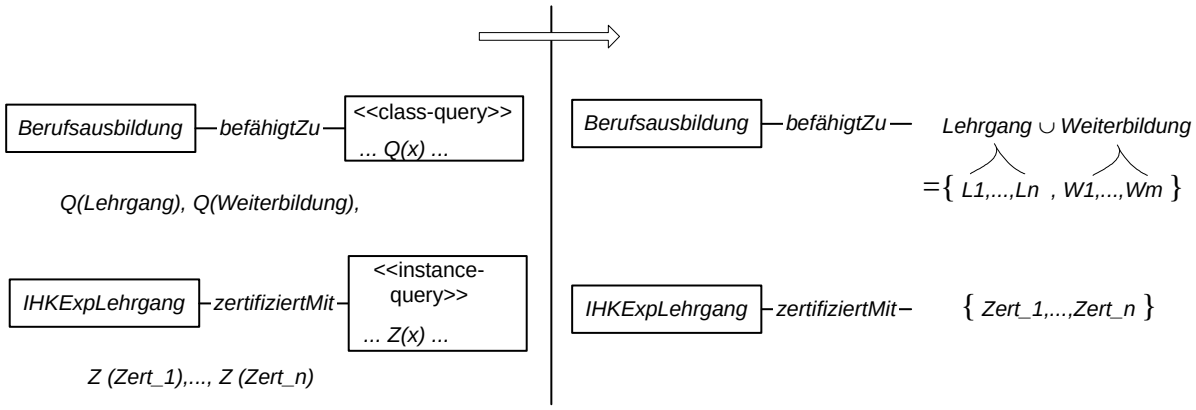

Bild 4.1: Komplexe Wertebereiche

4.1.3 DLP-Einbettungen und Kollektionen

Eine der wesentlichen Anforderungen ist die Integration von DL-Programmen. Wie auf Seite 86 unter Abschnitt 3.3 ausgeführt, ist Hornlogik als Axiomatisierungssprache für $\mathcal{I}$ ausgewählt. Damit ist das in $\mathcal{I}$ einzubettende DL-Fragment beschränkt auf die mit Hornlogik axiomatisierbaren Ansätze. Den Ausführungen von [HHK$^+$05, HSS06] folgend, wird der Ansatz DLP als Grundlage für die Integration von DL-Programmen gewählt.

Dem Nutzer wird mit diesem Ansatz ermöglicht, DL-Programme anzugeben, welche sowohl die in den Ontologiedeklarationen definierten Regeln benutzen, als auch von Ontologiedeklarationen selber genutzt werden können (siehe dazu Abschnitt 2.3 auf Seite 60, Bild 2.9). Wegen des in Abschnitt 3.3.5 motivierten Ausschlusses der Gleichheitsaxiome beschränken wir uns auf auf die DLP-Sprachklasse $\mathcal{L}_0^{IC}$ (siehe Bemerkung 2.4 auf Seite 65).

Da das Konzept der Ontologiedeklaration eine Benennung der Ontologien zum Zwecke der Modularisierung vorsieht, müssen DLP-Programme ebenfalls zur Einbettung in ein Gesamtsystem identifizierbar sein. Ein Gesamtsystem besteht dann aus einer Sammlung von Ontologiedeklarationen und DLP-Einbettungen. Die folgende Definition präzisiert diesen Zusammenhang.

Definition 4.8. ***(DLP-Einbettungen und Kollektionen)***

(i) Sei P ein $\mathcal{L}_0^{IC}$-DLP und $id \in \mathbf{R}$, wobei id nicht in P vorkommt. Dann heißt

der Ausdruck

dlp id *{* P *}*

DLP-Einbettung von P. Das P zugrunde liegende Alphabet ist das aus Definition 4.1. Hierbei werden als Klassennamen und Rollennamen Elemente aus $\mathbf{R}$ sowie als Individuen Elemente aus $\mathbf{R} \cup \mathbf{L}$ verwendet.

(ii) Der Tupel $K = (X_1, \ldots, X_n)$, wobei X_i eine Ontologiedeklaration oder DLP-Einbettung ist, wird *Kollektion* genannt. □

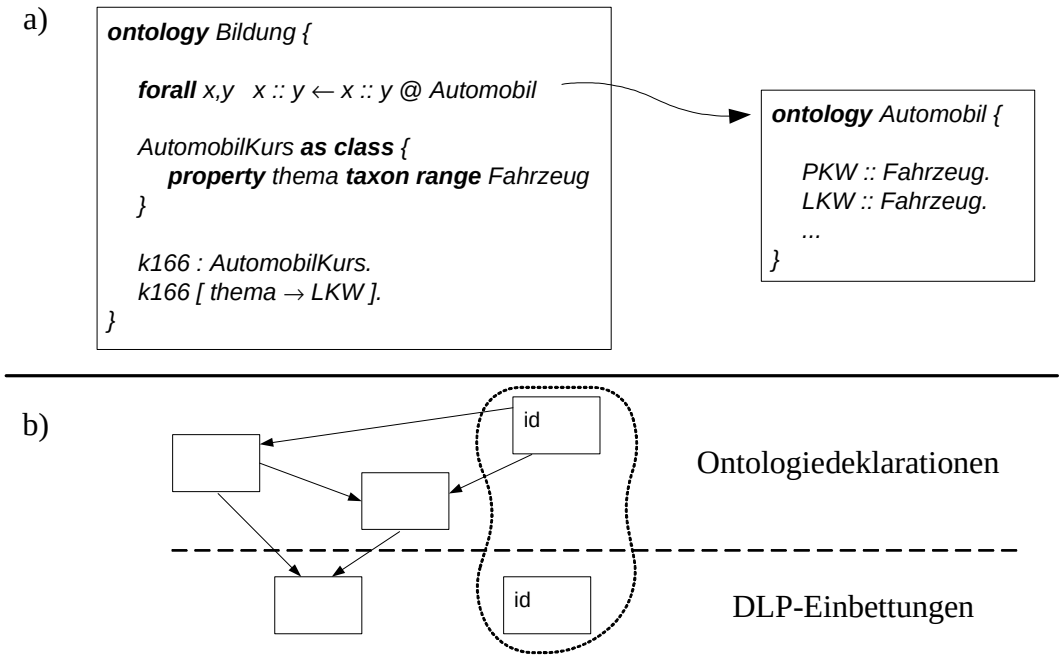

Bild 4.2: Kollektionen

Kontextpfade werden als höherwertige Ausdrücke angesehen und stehen deshalb nur Ontologiedeklarationen, nicht aber DLPs zur Verfügung. Mit Kollektionen wird das Gesamtsystem modularisiert. Bild 4.2 zeigt ein Beispiel für Kollektionen. In a) sind zwei Ontologiedeklarationen gegeben, die Erste mit dem Bezeichner *Bildung* definiert eine Klasse *AutomobilKurs*, welche das Attribut *thema* mit dem taxonomischen Wertebereich *Fahrzeug* besitzt. In der Zweiten wird eine Taxonomie über den im Automobilbau gebräuchlichen Begriffen definiert. Erst durch die Verwendung dieser Taxonomie mit Hilfe der ersten Regel in *Bildung* erhält man eine valide Eigenschaftszuweisung *k166 [thema -> LKW]*.

Bild 4.2.b) soll die Verwendungsbeziehung zwischen Mitgliedern einer Kollektion verdeutlichen. Immer dann, wenn im Rumpf einer Regel einer Ontologiedeklaration id ein zu id ungleicher Beziehungskontext id' für ein Atom A angegeben wird, so wird der Ableitungsprozess in der Ontologiedeklaration beziehungsweise DLP-Einbettung id' fortgesetzt. Wie sprechen dann von der *Verwendungsbeziehung* zwischen id und id' (id *verwendet* id'). Die Definition erzwingt nicht, daß die Verwendungsbeziehung einen azyklischen gerichteten Graph darstellt, jedoch sollte zumindest unter Ontologiedeklarationen diese Bedingung eingehalten werden.

Da die Menge der DLP-Konstrukte im Rahmen dieser Arbeit nicht um Konstrukte zur Angabe von Beziehungskontexten erweitert werden, können Ausdrücke innerhalb einer DLP-Einbettung keine anderen Mitglieder einer Kollektion gezielt verwenden. Jedoch können natürlich zwei Mitglieder mit dem gleichen Bezeichner versehen werden (siehe gestrichelte Linie in Bild 4.2.b). Durch dieses Vorgehen wird die Integration von $\mathcal{I}$-Regeln und DLP-Einbettungen erst möglich.

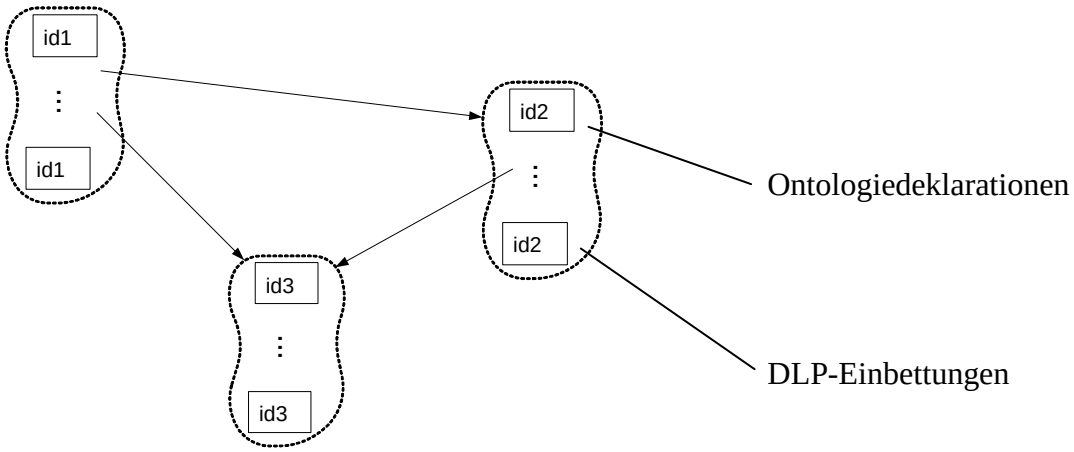

Bild 4.3: Einheiten über einer Kollektion

Genauer betrachtet lassen sich alle Mitglieder einer Kollektion, die denselben Bezeichner besitzen, als eine Einheit auffassen, die dann über die oben erläuterte Verwendungsbeziehung andere Einheiten importieren. Bild 4.3 veranschaulicht diesen Zusammenhang. Idealerweise sollte die für Einheiten sich ergebende Verwendungsbeziehung azyklisch sein.

4.2 Semantik

Als Axiomatisierungssprache ist Hornlogik mit well founded negation und integrity constraints gewählt (Hornlogik$_{wfs,IC}$). Die in diesem Abschnitt definierte axiomatische Semantik ordnet $\mathcal{I}$-Regeln den Formeln der Axiomatisierungssprache zu. Bild 4.4 zeigt den Zusammenhang der axiomatischen Semantik mit den bisher eingeführten Ontologiedeklarationen und DLP-Einbettungen. Nach der Übersetzung von DL-Programmen und Ontologiedeklarationen nach $\mathcal{I}$-Regeln mit Hilfe der Transformationsvorschriften π und Δ wird das Ergebnis auf die Axiomatisierungssprache mit μ abgebildet.

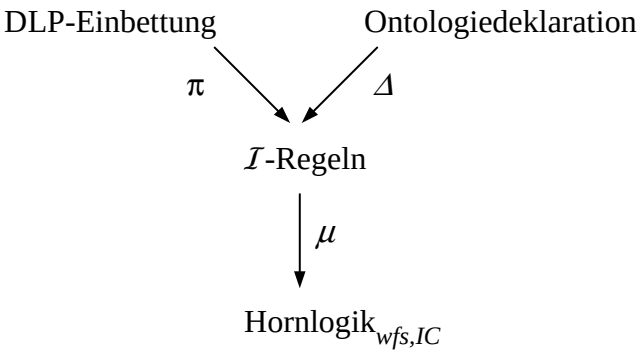

Bild 4.4: Sprachtransformationen

So wie bei den in dieser Arbeit vorgestellten Ontologie-Beschreibungssprachen spezifizieren die semantischen Bedingungen, wie die Signaturelemente zu interpretieren sind. Ferner spezifizieren die semantischen Bedingungen die Behandlung der Ressource-Terme, mit denen der begriffliche Kontext realisiert wird. Das Resultat der Transformationen bis hinunter zu Hornlogik$_{wfs,IC}$ zusammen mit den semantischen Bedingungen stellt die axiomatische Semantik der Ausgangssprachen dar. Präziser: die axiomatische Semantik einer Ontologiedeklaration D ist $\mu(\Delta(D))$ vereinigt mit den semantischen Bedingungen.

In den kommenden Abschnitten werden die Transformationsvorschriften π, Δ, μ und die semantischen Bedingungen sowie die axiomatische Semantik aller bisherigen Sprachteile eingeführt.

4.2.1 Transformation nach $\mathcal{I}$-Regeln

Ontologiedeklarationen stellen im wesentlichen syntaktische Abkürzungen für komplexere Ausdrücke der Regelsprache dar. So wie auch in [ABdB+05, LdBPF05] werden Ontologiedeklarationen daher auf die Regelsprache abgebildet.

Definition 4.9. ***(Transformation von Ontologiedeklarationen nach $\mathcal{I}$)*** Die Transformation $.^{\Delta}$ von Ontologiedeklarationen nach $\mathcal{I}$-Regeln ist definiert durch[4]

(1) $(\textbf{\textit{ontology}}\ id\ \{\ X, Y\ \})^{\Delta}$
$= \quad @\ id\ \{\ X^{\Delta}, Y^{\Delta}\ \}$

(2) $(id : c_1, \ldots, c_n :: d_1, \ldots, d_n\ \textbf{\textit{as}}\ A)^{\Delta}$
$= \quad id : c_1, \ldots, id : c_n, id :: d_1, \ldots, id :: d_n, (A, id)^{\Delta}$

(3) $((\ \textbf{\textit{property domain}}\ d_1, \ldots, d_n\ rp), id)^{\Delta}$
$= \quad id\ [\ \textbf{\textit{domain is}}\ \{d_1, \ldots, d_n\}\ ], (rp, id, \epsilon)^{\Delta}$

(4) $((\textbf{\textit{class}}\ \{\ X\ \}), id)^{\Delta}$
$= \quad (X, id)^{\Delta}$

(5) $((\ \textbf{\textit{property}}\ pid\ rp), id)^{\Delta}$
$= \quad (rp, pid, id)^{\Delta}$

(6) $((\big[\textbf{\textit{mandatory}}\big]\ \big[\textbf{\textit{functional}}\big]\ \big[\textbf{\textit{taxon}}\big]\ \textbf{\textit{range}}\ r), id, cid)^{\Delta}$
$= \quad \big[id\ [\ \textbf{\textit{mandatory}}\ (cid)\ ]\ \big], \big[id\ [\ \textbf{\textit{functional}}\ (cid)\ ]\ \big]\ ,$
$id\ [\ rkey\ (cid)\ \textbf{\textit{is}}\ (r, nid)^{\Delta_1}\ ]\ ,\ (r, nid)^{\Delta_2}$

wobei $rkey$ gleich ***taxonRange***, falls ***taxon*** angegeben,
sonst ***range***. nid ist eine neu erzeugte Ressource.

(7) $(r, nid)^{\Delta_1}$
$= \quad \{id_1, \ldots, id_n\}$, falls r der Form $(id_1, \ldots, id_n)$,
sonst nid.

(8) $((id_1, \ldots, id_n), nid)^{\Delta_2}$
$= \quad \epsilon.$

(9) $((\textbf{\textit{classQuery}}\ \{vid\ /\ p\}), nid)^{\Delta_2}$
$= \quad vid :: nid \leftrightarrow p.$

(10) $((\textbf{\textit{instanceQuery}}\ \{vid\ /\ p\}), nid)^{\Delta_2}$

[4] Zur besseren Lesbarkeit wird $\Delta(x)$ mit x^{Δ} notiert. Auch für die im weiteren Verlauf definierten Operationen ist $op(x)$ und x^{op} gleichbedeutend.

$$= \quad vid : nid \leftrightarrow p.$$

$$(11)\ (rule)^{\Delta} \quad = \quad rule$$

An Variablen in Großbuchstaben werden Listen von syntaktischen Elementen gebunden, wobei $\Delta(X, \ldots) = \Delta(x_1, \ldots), \ldots, \Delta(x_n, \ldots)$ für $X = x_1, \ldots, x_n$. Falls $cid = \epsilon$ in Gleichung (6) werden die Klammern nicht erzeugt. □

Lemma 4.10. Sei D eine Ontologiedeklaration. Dann ist $\Delta(D)$ eine Menge von kontextkonstanten Regeln.

Beweis: Transformationsvorschrift (2)-(10) erzeugen kontextlose Fakten. (11) ist die Identität. Nach Definition 4.6 enthält eine Regel aus D nur Atome, die kontextlos sind oder einen Beziehungskontext aus $\mathbf{R}$ besitzen. Nach Anwendung der Transformationsvorschrift (1) und der entsprechenden Auflösung der kontextlosen Atome mit id gemäß Bemerkung 4.5 erhält man nunmehr eine Menge von kontext-konstanten Regeln. □

Die nachstehende Definition baut auf der von [Vol04] eingeführten Transformation dlpd (siehe Definition 2.3 auf Seite 64) von DLPs nach Datalog auf. Die mit dlpd nach Datalog transformierten DLPs werden nach $\mathcal{I}$-Regeln überführt.

Definition 4.11. ***(Transformation von DLP-Einbettungen nach $\mathcal{I}$)*** Die Transformation π von DLP-Einbettungen nach $\mathcal{I}$ ist wie folgt definiert.

(i) Seien $A(t)$ und $B(t_1, t_2)$ Datalog-Atome. $\tilde{\pi}_x$ ist definiert durch

$$\tilde{\pi}_x(A(t)) = t : A \ @ \ x \quad \text{und} \quad \tilde{\pi}_x(B(t_1, t_2)) = t_1 \ [\ B \ -> \ t_2 \ ] @ x.$$

Ist $Y_1 \wedge \ldots \wedge Y_n \rightarrow X$ eine DLP-konforme Datalog-Formel, so ist

$$\tilde{\pi}_x(Y_1 \wedge \ldots \wedge Y_n \rightarrow X) = \tilde{\pi}_x(X) \leftarrow \tilde{\pi}_x(Y_1), \ldots, \tilde{\pi}_x(Y_n).$$

Für ein DLP-konformes Datalog-Programm D ist $\tilde{\pi}_x$ die Fortsetzung auf die Elemente von D.

(ii) Sei $E = \textbf{dlp}\ id\ \{\ P\ \}$ eine DLP-Einbettung. Dann ist $\pi(E) = \tilde{\pi}_{id}(\mathrm{dlpd}(P))$. □

Die ausgezeichneten ein- und zweistelligen Prädikate als Ergebnis der Transformation dlpd in (ii) entsprechen der Instanzbeziehung resp. der Beziehungssetzung von Individuen und werden dementsprechend in (i) nach $\mathcal{I}$-Regeln transformiert.

Lemma 4.12. Sei E eine DLP-Einbettung. Dann ist $\pi(E)$ eine Menge von kontext-konstanten Regeln.

Beweis: $\tilde{\pi}$ erzeugt nur Regeln mit Atomen im Beziehungskontext id. Da nach Definition 4.8.(i) $id \in \mathbf{R}$, ist $\pi(E)$ eine Menge von kontext-konstanten Regeln. □

4.2.2 Regeltransformation

Im folgenden wird die Transformation von $\mathcal{I}$-Regeln auf Hornlogik$_{wfs,IC}$-Formeln eingeführt.[5] Die nachstehende Definition identifiziert die im weiteren benötigten Teile einer Regelmenge sowie daraus erzeugte Formeln.

Definition 4.13. Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln. K_M, R_M und L_M sind definiert als die Menge der in M vorkommenden Kontextpfade, Ressourcen resp. Literale. Die Menge G_M ist wie folgt konstruiert: für jedes $r \in R_M$ und $l \in L_M$ gilt

$$resource(r), literal(l) \in G_M \ .$$

Für jedes $t = r_1/\ldots/r_k \in K_M$ gilt

$$contextPath(t, r_1, r_k) \in G_M \text{ und}$$
$$contextSubPath(r_1/\ldots/r_i) \in G_M \text{ für } i = 2, \ldots, k.$$

Ferner gehören zu G_M

$$literal(x) \vee resource(x) \rightarrow flatTerm(x),$$
$$flatTerm(x) \vee contextPath(x, y, z) \rightarrow universe(x),$$
$$universe(x) \wedge resource(y) \rightarrow equal(x, x, y),$$
$$literal(x) \wedge resource(y) \rightarrow instanceOf(x, Literal, y),$$
$$flatTerm(x) \wedge resource(y) \rightarrow instanceOf(x, flatUniverse, y),$$

wobei x, y, z Variablen. □

[5] Hier und in allen nachfolgenden Abschnitten ist die Verwendung der lloydTopor-Abkürzungen in Hornlogik$_{wfs,IC}$-Formeln erlaubt (siehe Definition 2.3.(i) auf Seite 64). Zur besseren Lesbarkeit wird, wenn nicht die Verständlichkeit fördernd, der Allabschluß hier und in den folgenden Abschnitten weggelassen.

Die konstruierten Formeln dienen der Erfassung aller bekannten Individuen, wie es auch in [MSS04] vorgenommen wird[6]. Die Annahme der Kontext-Konstanz von M verhindert, daß sich die B-Terme dieser Erfassung entziehen. Ferner werden Literale als Instanzen der entsprechenden Klasse festgelegt sowie alle Konstanten außer Kontextpfade als Instanzen der ausgezeichneten Klasse $flatUniverse$ betrachtet, was für eine korrekte Aufstellung der Signatur-Constraints erforderlich ist.

Definition 4.14. ***(Transformation von $\mathcal{I}$-Regeln nach Hornlogik)*** Sei $M = \{m_1, \ldots, m_n\}$ eine Menge von $\mathcal{I}$-Regeln. Die Transformation $.^{\mu}$ von M nach Hornlogik$_{wfs,IC}$ ist wie folgt definiert.

(1)	M^{μ}	$=$	$\{{m_1}^{\mu}, \ldots, {m_n}^{\mu}\}$
(2)	$(h_1, \ldots, h_m \leftarrow b_1, \ldots, b_n)^{\mu}$	$=$	$(h_1 \leftarrow b_1, \ldots, b_n)^{\mu} \wedge \ldots$
			$\ldots \wedge\ (h_m \leftarrow b_1, \ldots, b_n)^{\mu}$
(3)	$(h \leftarrow b_1, \ldots, b_n)^{\mu}$	$=$	${b_1}^{\mu} \wedge \ldots \wedge {b_n}^{\mu} \rightarrow h^{\mu}$
(4)	$(\textbf{\textit{naf}}\ a)^{\mu}$	$=$	$naf\ a^{\mu}$
(5)	$(\ \textbf{\textit{equal}}\ (x,y)\ @\ b\ )^{\mu}$	$=$	$equal(x, y, b)$
(6)	$(s\ [\ p\ \text{->}\ o\ ]\ @\ b)^{\mu}$	$=$	$statement(s, p, o, b)$
(7)	$(s\ [\ p\ \textbf{\textit{is}}\ o\ ]\ @\ b)^{\mu}$	$=$	$signature(s, p, flatUniverse, o, b)$, falls $p \in \mathbf{S}$
(8)	$(s\ [\ p(c)\ \textbf{\textit{is}}\ o\ ]\ @\ b)^{\mu}$	$=$	$signature(s, p, c, o, b)$, falls $p(c) \in \hat{\mathbf{S}}$
(9)	$(x :: y\ @\ b)^{\mu}$	$=$	$subClassOf(x, y, b)$
(10)	$(x : y\ @\ b)^{\mu}$	$=$	$instanceOf(x, y, b)$
(11)	$(x(x_1, \ldots, x_n)\ @\ b)^{\mu}$	$=$	$predicate(x, x_1, \ldots, x_n, b)$

Ressourcen, Literale und Signaturkonstrukte aus $\mathcal{I}$ werden direkt als Konstanten in Hornlogik$_{wfs,IC}$ übernommen. Funktions-, Prädikatensymbole und Variablen werden direkt übernommen. Auch für R-Terme und B-Terme ist μ die Identität. Ferner sei für ein $id \in \mathbf{R}$ die Transformation $\tilde{\mu}_{id}$ definiert durch $\tilde{\mu}_{id} = \mu \circ \tilde{\pi}_{id}$. □

Die Ergebnisse der Transformation mit μ sind zunächst ‚semantik-frei', d.h. es wird keine Bedeutung für die Schemakonstrukte und keine Constraints für die Verwendung von Kontextpfaden bestimmt. Der nächste Abschnitt behandelt diese sogenannten semantischen Bedingungen.

[6] Dort über das ausgezeichnete Prädikat $\mathcal{O}$ beziehungsweise *HU*. Dies wird als *grounding* bezeichnet.

4.2.3 Semantische Bedingungen und Axiome

Die im letzten Abschnitt gegebene Definition 4.14 legt zunächst die Abbildung von kontext-konstanten Mengen von $\mathcal{I}$-Regeln nach Mengen von Formeln in Hornlogik$_{wfs,IC}$ fest. Mit den in diesem Abschnitt aufgestellten Formeln werden drei Arten von semantischen Bedingungen festgelegt, nämlich die Bedingungen

- (1)-(4) zur Verwendbarkeit von Kontextpfaden,
- (5) zur Einschränkung von Kontextpfaden auf Taxonomien und
- (6)-(15) zur Interpretation von Schemaatomen.

Sie sind bezogen auf eine gegebene Menge M von kontext-konstanten $\mathcal{I}$-Regeln. Hier und in den folgenden Abschnitten bezeichnen fettgedruckte Buchstaben Metavariablen. Ferner gehören einzelne kleingeschriebene lateinische Buchstaben zu der zugrunde liegenden Variablenmenge und die restlichen Bezeichner zur Konstantenmenge.

4.2.3.1 Bedingungen für Kontextpfade

Die intendierte Semantik der Kontextpfade ist eine taxonomische Kategorisierung von Ressourcen. Sie sollen lediglich an Objektstelle in Beziehungsatomen stehen dürfen und nicht neue Klassen oder Eigenschaften repräsentieren. Eine (ausdrucksstärkere) Alternative wäre die Verwendung einer *sortierten Logik*, in der getypte Variablen den ‚Fluß' von Kontextpfaden durch die Argumente der Atome ‚steuern'. Wegen der der Wahl der unsortierten Axiomatisierungssprache Hornlogik$_{wfs,IC}$ und aufgrund der Tatsache, daß sich sortierte Logik auf unsortierte abbilden läßt, wird in dieser Arbeit darauf verzichtet. Die folgenden Constraints dienen dazu, die Stellen, an denen Kontextpfade in Atomen vorkommen dürfen, auf Objektstellen in Beziehungsatomen zu beschränken.[7]

(1) $statement(s,p,o,b) \wedge (\bigvee_{\mathbf{x}\in\{s,p,b\}} contextPath(\mathbf{x},y,z)) \rightarrow$

(2) $signature(x_1,\ldots,x_5) \wedge (\bigvee_{\mathbf{i}=1,\ldots,5} contextPath(x_\mathbf{i},y,z)) \rightarrow$

(3) $(subClassOf(x_1,\ldots,x_3) \vee instanceOf(x_1,\ldots,x_3)) \wedge$

[7] Es sei bemerkt, daß zur Erreichung der mit der Syntaxdefinition intendierten strengen Typisierung der Argumente von Atomen die Einführung von Sortierungsprädikaten (für alle Sorten) sinnvoll wäre. Hier werden exemplarisch (die ausdrucksschwächeren) Constraints für Kontextpfade dargestellt.

$$(\bigvee_{\mathbf{i}=1,\ldots,3} contextPath(x_{\mathbf{i}}, y, z)) \rightarrow$$

(4) $\bigwedge_{\mathbf{k}\in K} \left(predicate(x_1, x_2, \ldots, x_{\mathbf{k}}, b) \bigwedge_{\mathbf{l}=1,\ldots,\mathbf{k}} contextPath(x_{\mathbf{l}}, y, z) \rightarrow \right)$
wobei K die Menge der Stelligkeiten der in M vorkommenden P-Atome ist.

Das nachstehende K_M-induzierte Familie von Constraints erzwingt, daß sich Kontextpfade an den Taxonomien ausrichten müssen.

(5) $\bigwedge_{\mathbf{r_1}/\ldots/\mathbf{rn}\in K_M} ($
$statement(s, p, (\mathbf{r_1}/\ldots/\mathbf{r_n}), b) \wedge$
$((\bigvee_{\mathbf{i}=2,\ldots,n-1} naf\ subClassOf(\mathbf{r_i}, \mathbf{r_{i-1}}, b)) \vee$
$(naf\ subClassOf(\mathbf{r_n}, \mathbf{r_{n-1}}, b) \wedge naf\ instanceOf(\mathbf{r_n}, \mathbf{r_{n-1}}, b))) \rightarrow\ )$

Die letzten beiden Elemente des Pfades können sowohl in Spezialisierungs als auch in Instanzbeziehung stehen (siehe Bild 3.5 auf Seite 72).

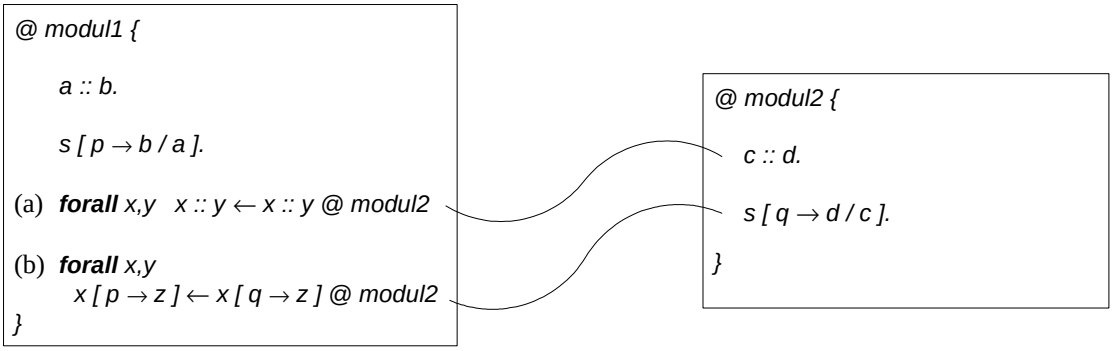

Bild 4.5: Taxonomie-Import

Von besonderer Wichtigkeit ist das Einbeziehen der Beziehungskontexte bei der Überprüfung der Pfade. Denn (5) drückt aus, daß, wenn die Verwendung eines (Begriffs-)Kontextpfades t in dem Beziehungskontext b stattfindet, die taxonomische Verträglichkeit genau in diesem Beziehungskontext b gelten muß. Eine globale Überprüfung der Taxonomie-Verträglichkeit ohne Einbeziehung der Beziehungskontexte würde einer möglichen Modularisierung der Regelmengen nicht gerecht, da jedes Modul seine eigenen Spezialisierungsbeziehungen definiert. Bild 4.5 zeigt ein Beispiel. Über die Regel (a) werden alle Spezialisierungsbeziehungen von *modul2* in *modul1* importiert. Regel (b) importiert alle *q*-Beziehungen aus *modul2* unter Umbenennung nach *p*. Daher gilt in *modul1* die Kategorisierung *s [p -> {b/a, d/c}]*. Aufgrund von

Regel (a) sind die beiden Pfade verträglich mit der Taxonomie. Ohne diese Regel würde zwar *s [p -> d/c] @ modul1* ableitbar sein, jedoch zur Verletzung von (5) führen, da *c :: d @ modul1* nicht ableitbar ist.

(P1)	$contextSubPath(x/y) \wedge subClassOf(y, x, b) \rightarrow taxonSub(x/y, b)$
(P2)	$contextSubPath(t/x/y) \wedge taxonSub(t/x, b) \wedge subClassOf(y, x, b) \rightarrow taxonSub(t/x/y, b)$
(P3)	$taxonSub(t, b) \rightarrow taxon(t, b)$
(P4)	$contextSubPath(x/y) \wedge instanceOf(y, x, b) \rightarrow taxon(x/y, b)$
(P5)	$contextPath(t/x/y, z, y) \wedge taxonSub(t/x, b) \wedge instanceOf(y, x, b) \rightarrow taxon(t/x/y, b)$
(P6)	$contextPath(t, x, y) \wedge statement(s, p, t, b) \wedge naf\ taxon(t, b) \rightarrow$

Fragment 4.4: Alternative Pfadvalidität

Eine Alternative für die Überprüfung der Pfade ist in Fragment 4.4 aufgeführt. Die Bedingungen sind deswegen nur als Alternative aufgezeigt, weil sie das Vorhandensein von Funktionssymbolen erzwingen, was bei den hier festgelegten semantischen Bedingungen zwecks Abbildbarkeit auf Datalog umgangen wird (siehe Bemerkung 4.22). Implikation (P1)-(P5) zählen alle gültigen Pfade auf. Während (P1)-(P3) die gültigen Pfade bezüglich der Spezialisierungsbeziehung bestimmt, definieren (P4) und (P5) diejenigen Pfade, an deren Ende zusätzlich die Instanzbeziehung stehen darf.

Bemerkung 4.15. Es sei bemerkt, daß der Zugriff auf die *grounding*-Prädikate in den Prämissen die Endlichkeit der Menge der folgerbaren Atome gewährleistet. Denn würde zum Beispiel in (P1) das *grounding* $contextSubPath(x/y)$ fehlen, so würde man aufgrund der Reflexivität von $subClassOf$ unendlich viele Fakten der Form $taxonSub(a/a/\ldots, b)$ folgern können.

4.2.3.2 Bedingungen für Schemakonstrukte

Die Interpretation der Signaturkonstrukte wird in den nachstehenden Formeln festgelegt.

$$(6) \quad statement(s, p, o, b) \wedge signature(p, domain, c, a, b) \wedge naf\ instanceOf(s, a, b) \rightarrow$$

(7) $flatTerm(o) \wedge statement(s,p,o,b) \wedge signature(p,range,c,a,b) \wedge$
$instanceOf(s,c,b) \wedge naf\ instanceOf(o,a,b) \rightarrow$

(8) $contextPath(o,x,y) \wedge statement(s,p,o,b) \wedge signature(p,range,c,a,b) \wedge$
$instanceOf(s,c,b) \wedge naf\ instanceOf(y,a,b) \rightarrow$

(9) $flatTerm(o) \wedge statement(s,p,o,b) \wedge signature(p,taxonRange,c,a,b) \wedge$
$instanceOf(s,c,b) \wedge naf\ subClassOf(o,a,b) \rightarrow$

(10) $contextPath(o,x,y) \wedge statement(s,p,o,b) \wedge$
$signature(p,taxonRange,c,a,b) \wedge$
$instanceOf(s,c,b) \wedge naf\ subClassOf(x,a,b) \rightarrow$

Constraints (6) und (7) stellt die klassische Interpretation der Definitions- und Wertebereichskonstrukte dar, so wie sie auch in RDF und F-Logic üblich ist. Für RDF gilt dies natürlich nur für den Fall, daß die Bedingungen auch wirklich als Constraints interpretiert werden.[8] Der einzige Unterschied besteht darin, daß hier eine Unterscheidung zwischen lokalen und globalen Eigenschaften ermöglicht wird. Prinzipiell wird eine Lokalisierung durch Einschränkung auf eine Quellklasse c für s in (7) erreicht. Die Globalität einer Eigenschaft kommt zustande durch Angabe der allgemeinsten Beschränkung in Definition 4.14.(7). Lokale Eigenschaftsdefinitionen sind in RDF nicht möglich. In F-Logic wiederum sind nur nichtoptionale lokale Eigenschaften ausdrückbar.

Constraints (8)-(10) betreffen die hier vorgenommenen Erweiterungen um Kontextpfade und taxonomische Kategorisierungen. Kontextpfade sind gültig bzgl. der Zielklasse, falls die in einen Kontext gestellte Ressource Instanz der Zielklasse ist. Bei taxonomischen Wertebereichen müssen sich Kontextpfade unterhalb der Taxonomie beginnend mit der Zielklassen einordnen lassen. Bild 4.6 zeigt den Zusammenhang. Im Fall der taxonomischen Kategorisierung muß das *erste* Element eines Kontextpfades oder eine einzelne Ressource eine *Spezialisierung der Zielklasse* sein. Im Fall der klassischen Wertebereichsangabe muß das *letzte* Element eines Kontextpfades oder eine einzelne Ressource eine *Instanz der Zielklasse* sein.

Die beiden folgenden Constraints behandeln in üblicher Weise die Kardinalität von Eigenschaftszuweisungen.[9]

(11) $signature(p,mandatory,c,true,b) \wedge instanceOf(s,c,b) \wedge$

[8] Zur unterschiedlichen Verwendung siehe Erörterung auf Seite 40.

[9] Das einfach zu formulierende Constraint, daß nicht gleichzeitig $true$ und $false$ in denselben Argumenten abgeleitet werden darf, wird hier als gegeben vorausgesetzt.

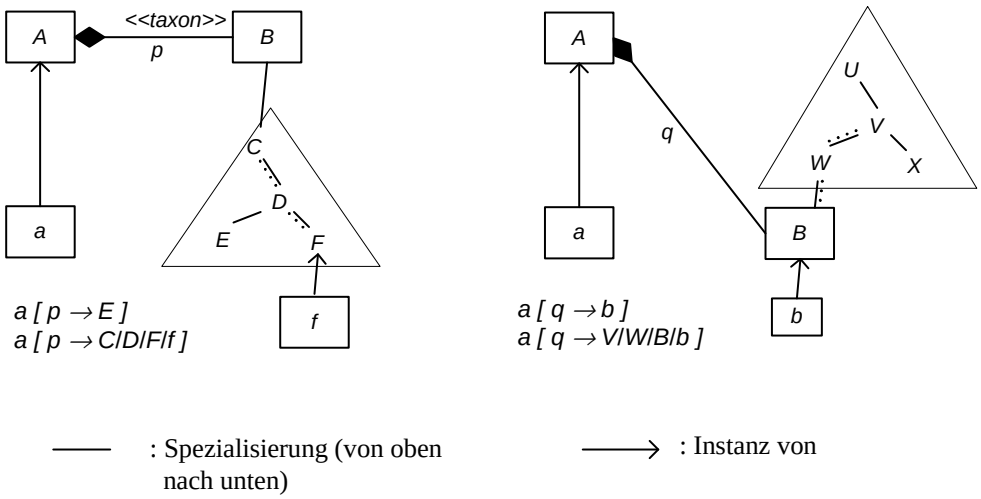

Bild 4.6: Signatur-Constraints und Kontextpfade

$$
\begin{aligned}
&naf\ statement(s,p,o,b) \rightarrow \\
(12)\quad &signature(p, functional, c, true, b) \wedge instanceOf(s,c,b) \wedge \\
&statement(s,p,o_1,b) \wedge statement(s,p,o_2,b) \wedge \\
&naf\ equal(o_1,o_2,b) \rightarrow
\end{aligned}
$$

Die folgenden Formeln beziehen sich auf die Spezialisierungs- und Instanzbeziehung

(13) $resource(x) \wedge resource(b) \rightarrow subClassOf(x,x,b)$

(14) $subClassOf(x,y,b) \wedge subClassOf(y,z,b) \rightarrow subClassOf(x,z,b)$

(15) $subClassOf(x,y,b) \wedge instanceOf(o,x,b) \rightarrow instanceOf(o,y,b)$

Die ersten beiden Implikationen geben die Reflexivität und Transitivität der Spezialisierungsbeziehung an. Die letzte als *subclass inclusion* bezeichnete Implikation definiert den klassischen auf der Mengenlehre basierenden Zusammenhang zwischen Instantiierung und Spezialisierung.

Die folgende Definition fasst nun die bisher aufgeführten semantischen Bedingungen zusammen.

Definition 4.16. ***(Semantische Bedingungen)*** Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln. Die mit AX_M bezeichnete Menge von *semantischen Bedingungen zu* $\mu(M)$ besteht aus der Expansion[10] der Formeln (1)-(15). □

4.2.3.3 Gleichheitsaxiome

Die Formel für das $equal$-Prädikat unter Definition 4.13 realisiert die in Abschnitt 3.3.5 behandelte lokale UNA derart, daß die Ungleichheit von x und y in einem Beziehungskontext b dann gelten soll, wenn $equal(x, y, b)$ nicht abgeleitet werden kann.[11] Dies ist bei syntaktischer Ungleichheit von x und y und unter der Voraussetzung, daß keine Regeln mit $equal$-Kopfatomen definiert wurden, der Fall. Soll aber die Gleichheit für syntaktisch ungleiche Individuen zugelassen werden, so müssen die folgenden Gleichheitsaxiome (Symmetrie, Transitivität und Substitutivität für Prädikate) zusätzlich gelten.

(16) $equal(x, y, b) \rightarrow equal(y, x, b)$

(17) $equal(x, y, b) \wedge equal(y, z, b) \rightarrow equal(x, z, b)$

(18) $\bigwedge_{\mathbf{p} \in P} \bigwedge_{\mathbf{k} \in S(\mathbf{p})}$
$equal(x_{\mathbf{1}}, y_{\mathbf{1}}, b) \wedge \ldots \wedge equal(x_{\mathbf{k\text{-}1}}, y_{\mathbf{k\text{-}1}}, b) \wedge \mathbf{p}(x_{\mathbf{1}}, \ldots, x_{\mathbf{k\text{-}1}}, b)$
$\rightarrow \mathbf{p}(y_{\mathbf{1}}, \ldots, y_{\mathbf{k\text{-}1}}, b)$
wobei P die Menge der in $\mu(M)$ vorkommenden Prädikate und $S(p)$ mit $p \in P$ die Menge der in $\mu(M)$ vorkommenden Stelligkeiten von p ist.

(19) $\bigwedge_{((\mathbf{r_1}/.../\mathbf{r_n}),(\mathbf{s_1}/.../\mathbf{s_n})) \in (K_M)^2} \big($
$equal(\mathbf{r_1}, \mathbf{s_1}, b) \wedge \ldots \wedge equal(\mathbf{r_n}, \mathbf{s_n}, b) \leftrightarrow equal((\mathbf{r_1}/\ldots/\mathbf{r_n}), (\mathbf{s_1}/\ldots/\mathbf{s_n}), b)\big)$

Definition 4.17. Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln. Die mit GAX_M bezeichnete Menge von *Gleichheitsaxiomen* zu $\mu(M)$ besteht aus der Expansion[12] der Formeln (16)-(19). □

Man beachte, das unter Hinzunahme von GAX_M die UNA erhalten bleibt, falls in M keine Regeln enthalten sind, die Gleichheitsatome im Kopf beinhalten.

[10] Mit Expansion ist hier die Auflösung der Metavariablen gemeint.

[11] Insofern steht das negative Atom $naf\, equal(x, y, b)$ unter der UNA für das Ungleichheits-Atom $inequal(x, y, b)$, was z.B. bei [EGM97] in gleicher Weise mit $\{(not\ Eq(x, y) \rightarrow Neq(x, z)), (true \rightarrow Eq(x, x))\}$ eingeführt wird.

[12] Mit Expansion ist hier die Auflösung der Metavariablen gemeint.

4.2.4 Axiomatisierung

4.2.4.1 Ontologiedeklarationen und DLP-Einbettungen

Aufbauend auf den Definitionen der letzten Abschnitte ist man nun in der Lage, die axiomatische Semantik für Ontologiedeklarationen festzulegen. Dies vollzieht sich in den folgenden Schritten. Zunächst werden Ontologiedeklarationen mit Hilfe der Transformationsvorschrift Δ nach $\mathcal{I}$-Regeln abgebildet. Anschließend wird das Ergebnis auf auf die Axiomatisierungssprache Hornlogik$_{wfs,IC}$ mit μ abgebildet. Das Resultat zusammen mit den in Hornlogik$_{wfs,IC}$ vorliegenden semantischen Bedingungen stellt die axiomatische Semantik von Ontologiedeklarationen dar. DLP-Einbettungen werden ohne die Hinzunahme der semantischen Bedingungen transformiert. Der Grund dafür ist, daß sowohl Kontextpfade in ihnen nicht formuliert werden können als auch daß sie von ihrem Wesen her keine Signaturen als Constraints festlegen.

Die axiomatische Semantik von Kollektionen, welche Ontologiedeklarationen und DLP-Einbettungen umfassen, wird erst nach gesonderter Betrachtung der Eigenheiten der Spezialisierungsbeziehung festgelegt. Die nachstehende Definition bestimmt nun die Semantik von Ontologiedeklarationen und DLP-Einbettungen aufbauend auf der Semantik von Regeln.

Definition 4.18. ***(Axiomatische Semantik von Regeln, Ontologiedeklarationen und DLP-Einbettungen)***

(i) Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln. Die *axiomatische Semantik* ϕ_{rules} *von* M ist definiert durch $\phi_{\text{rules}}(M) = \mu(M) \cup G_M \cup AX_M \cup GAX_M$.

(ii) Sei D eine Ontologiedeklaration. Die *axiomatische Semantik* ϕ *von* D ist definiert durch $\phi(D) = \phi_{\text{rules}}(\Delta(D))$.

(iii) Sei E eine DLP-Einbettung. Die *axiomatische Semantik* ϕ *von* E ist definiert durch $\phi(E) = \mu(\pi(E))$. □

Eine der wesentlichen Anforderungen an die hier vorgestellte Ontologie-Beschreibungssprache ist die Entscheidbarkeit von $\mathcal{I}$. Die Transformation eines Satzes der Ontologie-Beschreibungssprache in einen Satz der Axiomatisierungssprache sollte nicht dazu führen, daß Anfragen eine Nicht-Termination des Ableitungsverfahrens bewirken.[13] Die folgenden Lemmata stellen dies sicher.

[13] Als Ableitungsverfahren sei hier die SLG-Resolution [CW93] ausgewählt.

Lemma 4.19. Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln, D eine Ontologiedeklaration und E eine DLP-Einbettung. Dann enthalten $\phi_{\text{rules}}(M)$, $\phi(D)$ und $\phi(E)$ nur variablenfreie Terme.

Beweis: M enthält nur kontext-konstante Regeln. $M_1 = \Delta(D)$ und $M_2 = \pi(E)$ enthalten nur kontext-konstante Regeln nach Lemma 4.10 resp. 4.12. Zusammen mit der Variablenfreiheit von Kontextpfaden folgt, daß M, M_1, M_2 keine Terme mit Variablen enthalten. Weder G_{M_1}, AX_{M_1}, GAX_{M_1} noch das Ergebnis einer Anwendung von μ enthalten damit Terme mit Variablen nach Definition 4.13, 4.14 und somit auch $\phi_{\text{rules}}(M)$, $\phi(D) = \phi_{\text{rules}}(M_1)$ und $\phi(E) = \mu(M_2)$ □

Lemma 4.20. Ist F eine Hornlogik$_{wfs,IC}$-Formelmenge, die nur variablenfreie Terme enthält, so ist F entscheidbar.

Beweis: Für F gilt unmittelbar die *bounded term size property* und damit die Termination von Ableitungen [CW93]. □

Die Termination der bisher eingeführten Basiselemente ist durch das nachstehende Korollar, welches direkt aus Lemma 4.19 und 4.20 folgt, gegeben.

Korollar 4.21. ***(Entscheidbarkeit)*** Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln, D eine Ontologiedeklaration und E eine DLP-Einbettung. Dann sind $\phi_{\text{rules}}(M)$, $\phi(D)$ und $\phi(E)$ entscheidbar. □

Bemerkung 4.22. Aufgrund der Variablenfreiheit von Kontextpfaden und der Kontext-Konstanz von aus Ontologiedeklarationen und DLP-Einbettungen erzeugten $\mathcal{I}$-Regelmengen ist es möglich, diese auf Datalog$_{wfs,IC}$-Programme abzubilden. Dazu muß jeder Kontextpfad eineindeutig einer noch nicht vorkommenden Ressource zugeordnet werden, um die Abwesenheit von Funktionssymbolen zu gewährleisten. Die Sicherstellung der Datalog-safety ist per definitionem gegeben (Definition 4.4 auf Seite 95). Diese Vorschrift zusammen mit einem entsprechenden *grounding* der semantischen Bedingungen würde darüberhinaus die für Entscheidbarkeit sorgende DL-safety für den Fall gewährleisten, daß anstatt der DLP nach [Vol04] die Sprachklasse *Horn-*$\mathcal{SHIQ}$ nach [HMS05, KHSV05] für DLP-Einbettungen verwendet werden würde.

4.2.4.2 Bewahrung der DL-Semantik

DLP-Einbettungen erlauben die Integration von DLP-Programmen und der in dieser Arbeit vorgestellten Ontologie-Beschreibungssprache. Wie in den letzten Abschnitten dargestellt,

werden zu diesem Zwecke DLP-Programme mit π so kodiert, daß eine Passbarkeit zu den syntaktisch höherstufigen $\mathcal{I}$-Regelmengen gegeben ist. Eine anschließende Überführung mit μ gibt die axiomatische Semantik im Rahmen von $\mathcal{I}$ an. Damit wird aus der klassischen zweistufigen DL eine höherstufige DL, die es ermöglicht, die bei den F-Logic orientierten Sprachen selbstverständliche Unterstützung der Metamodellierung auch im DL-Kontext zu nutzen.

Eine der zentralen Eigenschaften, die nun gelten müssen, ist die Bewahrung der DL-Semantik. Zum Nachweis dieser Eigenschaft wird zunächst gezeigt, daß sich Ableitungen auf die kodierte Form übertragen lassen und umgekehrt (zur speziellen Kodierung von DLP-Programmen siehe Definition 4.11 und 4.14). Da DLP-Programme per definitionem definite logische Programme sind, wird dies für SLD-Widerlegungen gezeigt. Dabei wird im wesentlichen auf die Definitionen von Lloyd aufgebaut [Llo87].

Lemma 4.23. Sei D ein DLP-konformes Datalog-Programm, A eine Konjunktion von DLP-konformen Atomen und $id \in \mathbf{R}$. Eine Widerlegung von $\{A \rightarrow \mathit{false}\} \cup D$ existiert genau dann, wenn eine Widerlegung von $\{\tilde{\mu}_{id}(A \rightarrow \mathit{false})\} \cup \tilde{\mu}_{id}(D)$ existiert.

Beweis: (Induktion über die Länge der Widerlegung) Sei $A = p(s)$ und eine Widerlegung von $\{A \rightarrow \mathit{false}\} \cup D$ der Länge 1 gegeben, also

$$\left\langle (p(s) \rightarrow \mathit{false}), p(t), \theta \right\rangle \vdash \left\langle \mathit{false} \right\rangle ,$$

wobei $p(t) \in D$. Es gilt $(*)$:

$p(s)\theta = p(t)\theta$ gdw. $s\theta = t\theta$ gdw. $\mathit{instanceOf}(s, p, id)\theta = \mathit{instanceOf}(t, p, id)\theta$,
$\tilde{\mu}_{id}(p(s)) = \mu(\tilde{\pi}_{id}(p(s))) = \mu(s : p \ @ \ id) = \mathit{instanceOf}(s, p, id)$,
$\tilde{\mu}_{id}(p(t)) = \mathit{instanceOf}(t, p, id)$ und
$\tilde{\mu}_{id}(p(t)) \in \tilde{\mu}_{id}(D)$ gdw. $p(t) \in D$.

Somit existiert eine Widerlegung

$$\left\langle (\mathit{instanceOf}(s, p, id) \rightarrow \mathit{false}), \mathit{instanceOf}(t, p, id), \theta \right\rangle \vdash \left\langle \mathit{false} \right\rangle$$

mit $\mathit{instanceOf}(t, p, id) \in \tilde{\mu}_{id}(D)$, also eine Widerlegung von $\{\tilde{\mu}_{id}(A \rightarrow \mathit{false})\} \cup \tilde{\mu}_{id}(D)$. Der Beweis für $A = p(s_1, s_2)$, $\tilde{\mu}_{id}(A) = \mathit{statement}(s_1, p, s_2, id)$ verläuft analog. Die Rückrichtung folgt direkt aus den Gleichungen und Äquivalenzen $(*)$.

Sei nun eine Widerlegung der Länge n gegeben, also

$$\left\langle (X \wedge p(s) \rightarrow \mathit{false}), (Y \rightarrow p(t)), \theta_1 \right\rangle \vdash \left\langle ((X \wedge Y)\theta_1 \rightarrow \mathit{false}), C_2, \theta_2 \right\rangle \vdash \ldots \vdash \left\langle \mathit{false} \right\rangle ,$$

wobei X, Y Konjunktionen von DLP-konformen Atomen sind. Analog zur Argumentation für die Induktionsbasis folgt

$$\big\langle(\tilde{\mu}_{id}(X) \wedge instanceOf(s,p,id) \rightarrow false), (\tilde{\mu}_{id}(Y) \rightarrow instanceOf(t,p,id)), \theta_1\big\rangle \\ \vdash \big\langle((\tilde{\mu}_{id}(X) \wedge \tilde{\mu}_{id}(Y))\theta_1 \rightarrow false), \tilde{\mu}_{id}(C_2), \theta_2\big\rangle = S$$

und mit der Induktionsvoraussetzung für S schließlich die Behauptung. Der Induktionsschritt für den Fall des dreistelligen $statement$-Atoms verläuft analog. Die Rückrichtung folgt direkt aus $(*)$. □

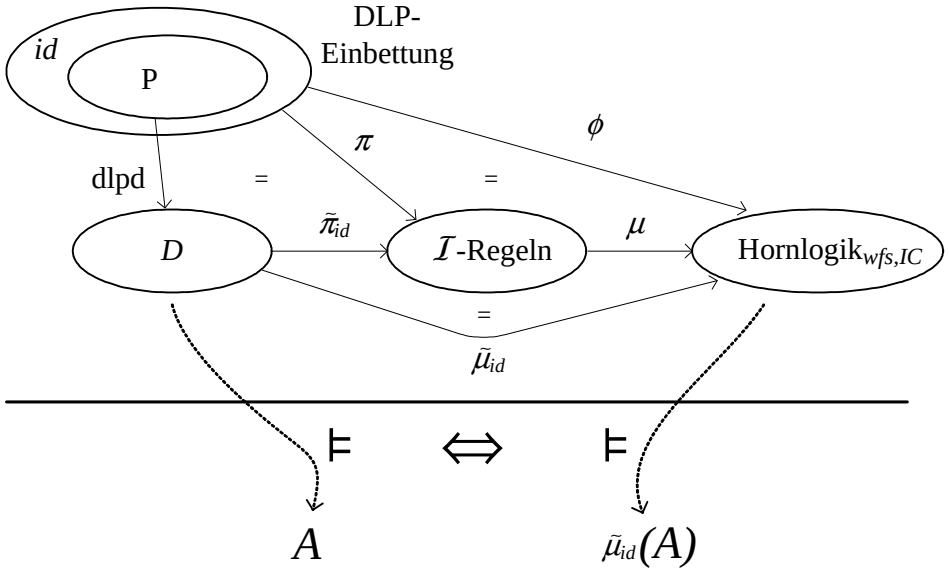

Bild 4.7: Bewahrung der DL-Semantik

Die DL-Semantik zu bewahren heißt, daß die aus einem DLP-Programm einer DLP-Einbettung folgerbaren Atome auch in kodierter Form aus dem Hornlogik$_{wfs,IC}$-Programm, das die axiomatische Semantik der Einbettung ist, folgen müssen, und umgekehrt. Bild 4.7 zeigt diesen Zusammenhang, der im nachstehenden Theorem formuliert wird.

Theorem 4.24. ***(Bewahrung der DL-Semantik)*** Sei $E = \textbf{\textit{dlp}}\ id\ \{\ P\ \}$ eine DLP-Einbettung, $D = \mathrm{dlpd}(P)$, A ein DLP-konformes Datalog-Atom. Es gilt

$$D \models A \quad \text{genau dann, wenn} \quad \phi(E) \models \tilde{\mu}_{id}(A)\ .$$

Beweis: Wegen der Korrektheit und Vollständigkeit des SLD-Kalkül gilt $D \models A$ gdw. eine Widerlegung von $\{A \rightarrow false\} \cup D$ existiert. Mit Lemma 4.23 ist dies äquivalent zur Existenz einer Widerlegung von $\{\tilde{\mu}_{id}(A \rightarrow false)\} \cup \tilde{\mu}_{id}(D)$. Mit $\phi(E) = \tilde{\mu}_{id}(D)$ und der Korrektheit und Vollständigkeit des SLD-Kalkül folgt die Behauptung. □

4.2.4.3 Besonderheit der Spezialisierungsbeziehung

In den vorangegangenen Abschnitten wurde die Axiomatisierung von Ontologiedeklarationen und DLP-Einbettungen getrennt behandelt. Die Fortsetzung der Transformation ϕ auf Kollektionen könnte nun in einfacher Weise als Vereinigung der Transformationen der Mitglieder einer Kollektion festgelegt werden, also schlicht

$$\phi(K) = \phi(X_1) \cup \ldots \cup \phi(X_n)$$

für eine Kollektion $K = (X_1, \ldots, X_n)$. Diese naive Semantik ist durchaus genügend, um zum einen das korrekte Zusammenwirken von Ontologiedeklarationen untereinander und zum anderen die Integration von DL und Regeln zu gewährleisten (siehe Abschnitt 4.1.3). Allerdings hat die naive Semantik den entscheidenden Nachteil, daß sie nicht vollständig die intendierte Semantik bezüglich der Spezialisierungsbeziehung abbildet, was im folgenden näher erläutert wird.[14]

Sowohl die DL-Sprachen wie DLP und OWL-DL als auch die Nicht-DL-Sprachen wie F-Logic, RDF und TRIPLE besitzen als Basiselement ein ausgezeichnetes Prädikat zur Angabe der Spezialisierungsbeziehung.[15] Allerdings ist ein wesentlicher Unterschied bei der klassischen Axiomatisierung festzustellen: während für die DL und damit für DLP-Einbettungen die Axiomatisierung

$$a \sqsubseteq b \qquad \xrightarrow{\phi} \qquad instanceOf(a, x) \rightarrow instanceOf(b, x)$$

eine Implikation ergibt, erzeugt die F-Logic-orientierte Axiomatisierung

$$a :: b \qquad \xrightarrow{\phi} \qquad subClassOf(a, b)$$

eine atomare Formel, wobei a, b Ressourcen sind, x eine Variable ist und vom Beziehungskontext an dieser Stelle abstrahiert wird. Während bei der DL die semantischen Eigenschaften der Spezialisierungsbeziehung implizit über die semantischen Eigenschaften der Folgerungsbeziehung in der Prädikatenlogik, wie z.B. Reflexivität und Transitivität, gegeben ist, müssen in den F-Logic-orientierten Sprachen explizit entsprechende Axiome aufgestellt werden (vergl. Formel (13) und (14) auf Seite 114).

[14] Es wird sich dabei auch zeigen, daß sogar die an den klassischen Vorbildern orientierte axiomatische Semantik ϕ für Ontologiedeklarationen als naiv bezeichnet werden muß.

[15] Zur besseren Unterscheidung der Sprachen wurde ja bisher dafür durchgängig das Symbol $\sqsubseteq$, *::* resp. *subClassOf* verwendet.

```
ontology auto {

    forall x
        x [ unterstützt -> Mobilität ]  ←  x :: Kraftfahrzeug.
}
dlp auto {

    PKW ⊑ LandKfz.
    LandKfz ⊑ Kraftfahrzeug.
}
```

Fragment 4.5:

Aus Sicht einer Integration, so wie sie auch in dieser Arbeit angestrebt wird, müßten aber die Ausdrücke $a :: b$ und $a \sqsubseteq b$ auf semantischer Ebene äquivalent sein. Betrachte man dazu in Fragment 4.5 zwei Kollektionsmitglieder, nämlich eine Ontologiedeklaration und eine DLP-Einbettung. Die intendierte Semantik der Integration der beiden Kollektionsmitglieder sollte nun die Spezialisierungbeziehung zwischen von *Kraftfahrzeug* zu *PKW* und damit die Aussage *PKW [unterstützt -> Mobilität] @ auto* gelten lassen. Die Axiomatisierung des Fragments ergibt die Hornlogik$_{wfs,IC}$-Formelmenge

$$\begin{aligned}&\{\ subClassOf(x, \textsf{Kraftfahrzeug}, \textsf{auto}) \rightarrow statement(x, \textsf{unterstützt}, \textsf{Mobilität}, \textsf{auto}),\\ &\quad instanceOf(\textsf{PKW}, x, \textsf{auto}) \rightarrow instanceOf(\textsf{LandKfz}, x, \textsf{auto})\\ &\quad instanceOf(\textsf{LandKfz}, x, \textsf{auto}) \rightarrow instanceOf(\textsf{Kraftfahrzeug}, x, \textsf{auto})\ \}\\ &\cup G_M \cup AX_M \cup GAX_M\ ,\end{aligned}$$

wobei x eine Variable ist und G_M, AX_M, GAX_M die entsprechenden zusätzlichen Bedingungen sind (vergl. Definition 4.18). Daraus ist aber keineswegs das gewünschte Atom

$$statement(\textsf{PKW}, \textsf{unterstützt}, \textsf{Mobilität}, \textsf{auto})$$

folgerbar.

Bemerkung 4.25. Dieses Problem tritt nicht nur in Zusammenhang mit der Hinzunahme von DL auf, sondern auch, wenn nur F-Logic-orientierte Einheiten wie Ontologiedeklaration verwendet werden. Denn derselbe Fall tritt ein, wenn man als zweites Kollektionsmitglied statt einer DLP-Einbettung eine Ontologiedeklaration verwenden würde, welche die zwei $instanceOf$-Regeln direkt mit dem Konstrukt *:* angeben würde (siehe Fußnote 14).

Der Grund für dieses Problem ist, daß es keine semantische Bedingung gibt, so daß aus der Implikation bzgl. der Instanzbeziehung die Spezialisierungsbeziehung folgt. Bezogen auf das Beispiel müßte die Bedingung die Folgerbarkeit von $subClassOf(\mathsf{PKW}, \mathsf{Kraftfahrzeug}, \mathsf{auto})$ aus

$$instanceOf(\mathsf{PKW}, x, \mathsf{auto}) \rightarrow instanceOf(\mathsf{Kraftfahrzeug}, x, \mathsf{auto})$$

ermöglichen. Übliche DL-Reasoner haben kein Problem damit, da sie aufgrund der fehlenden Regelkomponente diese Bedingung nicht während des Inferenzprozesses nutzen, sondern sie lediglich oberhalb der T-/A-Box in speziell ausgerichteten Inferenzverfahren berücksichtigen müssen. Die klassischen Ansätze aus Abschnitt 2.3 auf Seite 60, die DL und Regeln integrieren (also auch DLP), haben gar nicht ein Problem damit, da sie aufgrund der fehlenden syntaktischen Höherstufigkeit die Spezialisierungsbeziehung gar nicht erst als atomare Formel ausdrücken können.[16] Bei den F-Logic-orientierten Sprachen ist dieses Problem vorhanden, wird aber nicht berücksichtigt.

Zur Lösung dieses Problems, also die Ermittlung der Spezialisierungsbeziehungen auf der Basis der angegebenen Implikationen, die nicht mehr direkt in Hornlogik ausdrückbar ist, wird hier eine Verallgemeinerung des Verfahrens für den *subsumption check* von [Vol04] verwendet (vergl. Bemerkung 2.5 und Satz 2.6 auf Seite 66): der Test, ob x eine Spezialisierung von y im Kontext z bei einer gegebenen Hornformelmenge F ist, die aus einer Ontologiedeklaration oder DLP-Einbettung erzeugt wurde, wird vorgenommen mit Hilfe des Tests, ob $instanceOf(s, y, z)$ aus $F \cup \{instanceOf(s, x, z)\}$ folgt, wobei x, y aus der Menge der Grundterme aus F sind und z aus der Menge der Konstanten an der Stelle des Beziehungskontextes in den Regeln aus F ist. s ist eine noch nicht in F vorkommende Konstante. Darauf aufbauend ist dort ein Verfahren beschrieben, das die Menge der Spezialisierungsbeziehungen berechnet.[17] Allerdings ist dieses Verfahren, das hier mit $\mathrm{classif}(F)$ bezeichnet wird und für unseren Fall eine Menge von $subClassOf$-Grundatomen liefert, nur anwendbar, wenn keine nichtmonotone Negation verwendet wird.

Da es aber sein kann, daß mit $\mathrm{classif}$ erzeugte Spezialisierungsbeziehungen wiederum die Basis für neue Spezialisierungsbeziehungen sind, muß die Iteration

[16] Der Ansatz [Mot05] unterstützt zwar eine Höherstufigkeit, jedoch wird auf dieses Problem der Regelintegration nicht eingegangen.

[17] Auf spezielle Optimierungen des Verfahrens wird in [Vol04, Abschnitt 5.4] verwiesen. Da F nur Grundterme und Variablen beinhaltet, terminiert dieses.

$\mathrm{scext}(F) :=$
$X := \emptyset; F' := F$
while true do $\{ X := \mathrm{classif}(F'); $ ***if*** $X \subseteq F'$ ***then return*** F' ***else*** $F' := F' \cup X \}$

vorgenommen werden.

Das nachfolgende Lemma sichert die praktische Anwendbarkeit des auf $\mathrm{classif}$ basierenden Verfahrens scext, welches für die Definition der axiomatischen Semantik erforderlich ist.

Lemma 4.26. Sei eine Hornformelmenge F gegeben, die nur Grundterme und Variablen enthält. Dann terminiert $\mathrm{scext}(F)$.

Beweis: Da F nur Grundterme und Variablen enthält, ist die Menge S der bildbaren $subClassOf$-Grundatome endlich. Somit ist innerhalb der Iteration von scext die Menge F' beschränkt durch die endliche Menge $S \cup F$, womit folgt, daß es einen Iterationsschritt geben muß, in dem für die erzeugte Menge X von $subClassOf$-Grundatomen $X \subseteq F'$ gilt. Damit und wegen der Termination von $\mathrm{classif}$ ist die Termination von scext gewährleistet. □

4.2.4.4 Erweiterte Axiomatisierung

Wie in [KHSV05] dargelegt, ist die nichtmonotone Negation und damit die bei $\mathcal{I}$ genutzte *wfs*-Negation nicht verträglich mit der Semantik von DL. Jedoch ist, wie dort angeführt, eine Nutzung der nichtmonotonen Negation ‚oberhalb' von DL möglich, was heißen soll, daß keine Ableitung von DL-Atomen auf Regeln mit *wfs*-Negation basieren darf, wohl aber umgekehrt. Außerdem ist das im letzten Abschnitt eingeführte Verfahren scext zur Berechnung der Spezialisierungsbeziehungen nur ohne nichtmonotone Negation verwendbar. Daher muß noch eine spezielle Bedingung für die aus Ontologiedeklarationen und DLP-Einbettungen erzeugten Hornlogik$_{wfs,IC}$-Formelmengen eingehalten werden. Diese Bedingung muß sicherstellen, daß bei allen Ableitungen von DL-Atomen keine Hornformeln benutzt werden, die nichtmonotone Negation verwenden. Dies betrifft also Ableitungen bzgl. der Prädikate $instanceOf$ und $statement$. Welche Hornformeln nutzbar sind, ergibt sich aus der Passbarkeit zwischen Formelrümpfen und -köpfen.

Die sonst übliche Definition der Abhängigkeiten zwischen Hornformeln, bei der die Prädikate zur Überprüfung der Abhängigkeiten genutzt werden (z.B. Datalog-Stratifizierung), wäre für die Art der hier verwendeten Hornformeln zu grob, da wegen der geforderten syntaktischen Höherstufigkeit lediglich die in Definition 4.14 aufgeführten Prädikate zuzüglich einiger Basisprädikate genutzt werden. Daher basiert die nachstehende Definitionen auf der Unifizier-

barkeit von Hornformelköpfen.

Definition 4.27. ***(Abhängigkeitsgraph)*** Sei F eine Menge von Hornlogik$_{wfs,IC}$-Formeln. Eine Formel $A \in F$ heißt *abhängig von* einer Formel $B \in F$, falls ein (positives oder negatives) Rumpfatom a in A existiert, so daß der Kopf von B mit a unifizierbar ist. Die Abhängigkeit wird *positiv* resp. *negativ* genannt, falls a definit resp. nicht definit ist. Der *Abhängigkeitsgraph* von F wird wie folgt gebildet: jedes Element aus F ist ein Knoten und jede Abhängigkeit eine Kante. Entsprechend der Abhängigkeit wird eine Kante *positiv* oder *negativ* genannt. Ein Pfad heißt *negativ*, falls mindestens eine Kante in ihm negativ ist, ansonsten *positiv*. □

Mit Hilfe des Abhängigkeitsgraphen ist man nun in der Lage, die oben erläuterte Bedingung für Hornlogik$_{wfs,IC}$-Formelmengen zu formulieren, die sowohl die Anwendbarkeit von scext als auch die Verträglichkeit mit der DL-Semantik gewährleistet.

Definition 4.28. ***(DL-Verträglichkeit)*** Sei F eine Hornlogik$_{wfs,IC}$-Formelmenge. Sei $X = (instanceOf(u, v, w) \rightarrow false)$ und $Y = (statement(u, v, w, x) \rightarrow false)$, wobei u, v, w, x Variablen sind. F heißt *DL-verträglich*, falls in dem Abhängigkeitsgraphen von $F \cup \{X, Y\}$ kein negativer Pfad mit Anfangsknoten X oder Y existiert. □

Die nachstehende Definition erweitert die in Definition 4.18 auf Seite 116 eingeführte Axiomatisierung um die Anwendung von scext. Dabei erfolgt für DLP-Einbettungen keine Erweiterung, da in ihnen keine Atome für Spezialisierungsbeziehungen vorkommen.

Definition 4.29. ***(Erweiterte axiomatische Semantik von Regeln, Ontologiedeklarationen und DLP-Einbettungen)***

(i) Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln und $\phi_{\mathrm{rules}}(M)$ DL-verträglich. Die *erweiterte axiomatische Semantik* ϕ^*_{rules} *von* M ist definiert durch $\phi^*_{\mathrm{rules}}(M) = \mathrm{scext}(\phi_{\mathrm{rules}}(M))$.

(ii) Sei D eine Ontologiedeklaration und $\phi(D)$ DL-verträglich. Die *erweiterte axiomatische Semantik* ϕ^* *von* D ist definiert durch $\phi^*(D) = \mathrm{scext}(\phi(D))$.

(iii) Sei E eine DLP-Einbettung. Die *erweiterte axiomatische Semantik* ϕ^* *von* E ist gleich ϕ. □

Lemma 4.26 sorgt für die Erzeugbarkeit der die erweiterte axiomatische Semantik repräsentierende Formelmenge. Der Teil (i) für Regelmengen M würde natürlich nicht sinnvoll sein, wenn die durch ϕ_{rules} hinzukommenden Formelmengen G_M, AX_M und GAX_M an sich schon

nicht DL-verträglich sein würden. Das folgende Lemma stellt die DL-Verträglichkeit für diese Basismengen sicher.

Lemma 4.30. Sei M eine kontext-konstante Menge von $\mathcal{I}$-Regeln. Dann ist $G_M \cup AX_M \cup GAX_M$ DL-verträglich.

Beweis: $X = G_M \cup AX_M \cup GAX_M$ besteht nur aus definiten Formeln oder Constraints. Letztere können aber nur Anfangsknoten von Pfaden durch den Abhängigkeitsgraph sein. Somit kann kein negativer Pfad in dem Abhängigkeitsgraphen von X mit den Anfangsknoten X oder Y aus 4.28 existieren. □

Will man Ontologiedeklarationen und $\mathcal{I}$-Regelmengen unter der klassischen Semantik, vergleichbar mit der aus [ABdB$^+$05, LdBPF05] für Web-orientierte Sprachen und [DEFS99, FHK$^+$97, YKZ03] für F-Logic, verwenden, so genügt Definition 4.18. Die Semantik von Kollektionen gibt die nachstehende Definition an.

Definition 4.31. ***(Axiomatische Semantik von Kollektionen)***
Sei $K = (X_1, \ldots, X_n)$ eine Kollektion. Die *erweiterte axiomatische Semantik* ϕ^* *von* K ist definiert durch $\phi^*(K) = \mathrm{scext}(\phi(X_1) \cup \ldots \cup \phi(X_n))$. □

Wie in Abschnitt 3.3.4 auf Seite 89 gefordert, muß es im Kontext von Kollektionen von Ontologiedeklarationen und DLP-Einbettungen möglich sein, die logischen Zusammenhänge zwischen Ressourcen, seien damit Klassen, Attribute oder Kontexte intendiert, rein deklarativ zu definieren, ohne auf etwaige Fragestellungen der Terminierung Rücksicht zu nehmen. Der Sicherstellung dieser Eigenschaft dient der folgende Satz.

Satz 4.32. ***(Entscheidbarkeit für Kollektionen)*** Sei $K = (X_1, \ldots, X_n)$ eine Kollektion. Dann ist $\phi^*(K)$ entscheidbar.

Beweis: Nach Definition ist $\phi^*(K) = \mathrm{scext}(\phi(X_1) \cup \ldots \cup \phi(X_n))$. Wegen Lemma 4.19, der Tatsache, daß scext lediglich Grundatome hinzufügt, und Lemma 4.20 folgt die Behauptung. □

4.3 Vergleich mit bestehenden Ansätzen

Die bestehenden Ansätze, die zu der in dieser Arbeit vorgestellten Ontologie-Beschreibungssprache herangezogen werden, lassen sich in drei Gruppen aufteilen. Zum einen sind die auf F-Logic basierenden Ansätze zu nennen, welche ihren Ursprung in den durch [KLW95, KL89]

eingeführten Konzepten haben. Zu diesen Ansätzen gehören FLORA2 und TRIPLE [YKZ03, SD02], welche das Inferenzsystem XSB [SSW94] für Hornlogik als unterliegendes System besitzen, und die Ansätze FLORID und Ontobroker [FHK+97, DEFS99], welche eigens dafür entwickelte vorwärtsverkettende Inferenzkomponenten nutzen.

Als Vertreter der zweiten Gruppe lassen sich die an *description logic* [BCM+03] ausgerichteten Ansätze unter Hinzunahme von Regeln ansehen. Dazu gehören die Ansätze DLP, OWL/DL+rules und ORL aus [GHVD03, MSS04, HPS04]. Die Vertreter der dritten Gruppe stellen Adaptionen beziehungsweise Übernahmen der Sprachen der ersten beiden Gruppen dar. Sie sind Vorschläge für den WWW-Standard einer *Semantic Web*-Regelsprache und direkt ausgerichtet auf die Einbettbarkeit in die restlichen Standards des WWW. Die dritte Gruppe umfaßt SWRL[18] und WRL [HPSBT05, ABdB+05], wobei WRL von der F-Logic basierten Sprache WSML [LdBPF05], welche zur Spezifikation von Web-Diensten vorgesehen ist, abstammt.

Eine der Anforderungen an die in dieser Arbeit vorgestellten Ontologie-Beschreibungssprache ist es, eine Höherstufigkeit der Logik zu gewährleisten, d.h. die wesentlichen Elemente der konzeptuellen Modellierung wie Klassen, Attribute und Prädikate über diesen als *first class citizens* zu behandeln. Dazu wurde hier der Ansatz der syntaktischen Höherstufigkeit aus [KL89, KLW95, CKW93] als Basis gewählt, der von TRIPLE, FLORA2, FLORID, Ontobroker und WRL unterstützt wird. Diese Ansätze behandeln jedoch nicht das in Abschnitt 4.2.4.3 und 4.2.4.4 dargelegte Problem der Interpretation von Spezialisierungsbeziehungen. Von den DL-basierten Ansätzen schlägt nur [MSS04] eine entsprechende Erweiterung zur Höherstufigkeit in Zusammenhang mit der Metamodellierung, angelehnt an [CKW93], vor, wobei aber auch hier eine Erweiterung bzgl. der Spezialisierung nicht mit einbezogen wird.

Die Integration von DL und Regeln werden bei allen Ansätzen der zweiten und dritten Gruppe vorgenommen. DLP und OWL-DL+rules basieren dabei auf dem hier adaptierten Transformationsansatz, welche DL auf Hornlogik resp. disjunktives Datalog abbildet. Bezüglich der DL-Mächtigkeit wurde hier zugunsten der praktischen Anwendbarkeit (vergl. [HHK+05, HSS06]) eine Untermenge von DLP verwendet. Sowohl SWRL als auch WRL bieten eine Integration von DL und Regeln. Die DL-Mächtigkeit von SWRL übersteigt zwar die der hier eingeführten DLP-Einbettungen, allerdings fehlt dort der für eine Metamodellierung notwendige Übergang zur syntaktischen Höherstufigkeit. Die aus dem Bereich der Dienstemodellierung stammende WRL bietet eine DL-Einbettung in einem eingeschränkten Maße insofern, daß zwar eine Untermenge von OWL-DL über den Transformationsansatz eingebettet werden kann, jedoch

[18] Wie von den Autoren in [HPSBT05] bemerkt, ist wurde ORL in SWRL umbenannt.

eine Berücksichtigung des für die DL-Integration wichtigen Spezialisierungproblems nicht mit einbezogen wird.

Die Kontextualisierung von Beziehungssetzungen ist nur in den Ansätzen TRIPLE, FLORA2 und Ontobroker möglich. TRIPLE und Ontobroker lassen zusätzlich zu der Angabe von Konstanten als Kontext wie in FLORA2 beliebige Beziehungskontextterme zu, womit eine Parametrisierung von Regelmengen zum Zwecke der Transformation möglich wird. Der hier entwickelte Ansatz läßt den allgemeinen Fall von TRIPLE und Ontobroker zu, trennt jedoch die explizite Modularisierung im Sinne von FLORA2 von der für Transformationen benötigten Angabe von beliebigen Beziehungskontexttermen zugunsten der Praktikabilität und Entscheidbarkeit.

Keine der aus der Literatur bekannten Ansätze behandeln das hier eingeführte Konzept des taxonomischen Begriffskontextes und das entsprechende Konstrukt des Kontextpfades. Auch die Verwendung der Kontextpfade bei der Kategorisierung von Ressourcen und die Einbeziehung von entsprechenden Signaturkonstrukten und Constraints über Kontextpfaden ist bei keinem der Ansätze gegeben. Lediglich der Begriffskontext in Form des Namensraums wird bei allen Ansätzen unterstützt.

Die Möglichkeit der Vereinbarung von Ontologien ohne eine Kenntnis, wie diese mit Hilfe von Hornregeln entworfen werden können, unterstützt diejenige Zielgruppe, welche aus Anwendungsbereichen kommt, in denen keine prädikatenlogischen Vorkenntnisse vorauszusetzen sind. Bei den Ansätzen der ersten Gruppe steht auf syntaktischer Ebene ein regelorientierter Entwurf im Mittelpunkt. Die zur zweiten und dritten Gruppe gehörenden Ansätze OWL-DL, DLP und SWRL bieten spezielle Konstrukte gemäß der klassischen Syntax für DL-Ausdrücke. WRL bietet die von F-Logic stammenden und in Richtung der objektorientierten Programmierung modifizierten Konstrukte zur Vereinbarung von Ontologien an. Der hier an WRL ausgerichtete Ansatz bietet zusätzlich explizite Konstrukte für die Definition von Klassen mit Hilfe von klassendefinierenden Anfragen.

5 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurden die grundlegenden Konstrukte der Basissprachen F-Logic und *Description Logic* (DL) sowie der Web-orientierten Sprachen RDF, OWL und TRIPLE vorgestellt. Ferner wurden Integrationsansätze für DL und Regeln behandelt und die Semantik der Sprachen erläutert, wobei sowohl die modelltheoretisch orientierte Semantik umrissen als auch die axiomatische Semantik dargelegt wurde. Als Voraussetzung für die Konzeption einer Ontologie-Beschreibungssprache wurden Anforderungen an den Sprachumfang und ein Vergleich mit den bestehenden Ansätzen aufgestellt. Die sich daraus ergebenden Anforderungen an die Axiomatisierungssprache wurden erörtert.

Die folgenden Kernaspekte spielten eine wesentliche Rolle bei dem Entwurf einer integrativen Ontologie-Beschreibungssprache $\mathcal{I}$.

- syntaktische Höherstufigkeit
- DL-Einbettung
- Negation
- Entscheidbarkeit

Die syntaktische Höherstufigkeit von $\mathcal{I}$ wurde gemäß den Konzepten von F-Logic und TRIPLE eingeführt. Es erfolgte eine Einbettung der von ihrer originalen Definition her nicht höherstufigen DL in $\mathcal{I}$. Das Problem der Anpassung der semantischen Bedingungen der Spezialisierungsbeziehung an die semantischen Besonderheiten der Höherstufigkeit, welches in den bestehenden Ansätzen nicht behandelt wird, wurde dabei berücksichtigt. Die Bewahrung der DL-Semantik bei der Einbettung in $\mathcal{I}$ und dem damit verbundenen Übergang zur Höherstufigkeit wurde gezeigt.

Die nichtmonotone Negation unter der Interpretation der wohlfundierten Semantik wurde in $\mathcal{I}$ mit aufgenommen. Es erfolgte in diesem Zusammenhang die Aufstellung von notwendigen

Randbedingungen zur Sicherstellung der Verträglichkeit mit DL. Um die praktische Anwendbarkeit zu gewährleisten, ist die Entscheidbarkeit der Basiskomponenten von $\mathcal{I}$ eine der wesentlichen Anforderungen gewesen, welche im Rahmen der formalen Semantik gezeigt wurde.

Die folgenden syntaktisch höherwertigen Sprachelemente, die zum Teil nicht in den bestehenden Ansätzen behandelt werden, wurden im Rahmen der vorliegenden Arbeit eingeführt. Dies sind Sprachelemente zur

- Kontextualisierung,
- Definition von Ontologiesammlungen,
- Kategorisierung und
- Definition von Klassen über Anfragen.

Das mit TRIPLE eingeführte Konzept des Beziehungskontextes wurde für $\mathcal{I}$ und damit für DL-Einbettungen mit entsprechenden Randbedingungen übernommen. Es wurden Konstrukte entwickelt, welche eine Vereinbarung von Ontologien und ganzen Ontologiesammlungen im Stil der objektorientierten Modellierung erlauben.

Das Konzept des taxonomischen Kontextes, das in den bestehenden Ansätzen nicht behandelt wird, wurde eingeführt. Ein Konstrukt zur Angabe von taxonomieverträglichen Kontextpfaden gemäß dieses Konzeptes wurde vorgestellt. Dabei mussten entsprechende Erweiterungen der semantischen Bedingungen berücksichtigt werden. Die Kategorisierung von Modellelementen wurde um die Möglichkeit der taxonomischen Kategorisierung erweitert, welche eine Anpassung des Signaturkonzepts erforderte. Eine Erweiterung von Ontologiedeklarationen um syntaktisch höherwertige Sprachelemente zur Definition von Klassen ist vorgenommen worden. Dazu gehörten Sprachelemente, die eine Konstruktion von Klassen mit Hilfe von Anfragen erlauben. Schließlich erfolgte eine semantische Fundierung aller Erweiterungen über die Axiomatisierung der Sprachelemente mit Hilfe der Hornlogik, welche wegen ihrer Praktikabilität als Axiomatisierungssprache ausgewählt wurde.

Die folgenden Anknüpfungspunkte an die vorliegende Arbeit sind gegeben. Sie betreffen zum einen die Suche in Dokumentbeständen, nämlich die Informationsselektion auf der Basis von Kontextpfaden, insbesondere die unscharfe Suche über Kontextpfaden. Dies würde weiterführende Untersuchungen von Konzepten zu Ähnlichkeitsmaßen über Kontextpfade erfordern. Weitere Anknüpfungspunkte an die vorliegende Arbeit betreffen die Evolution von

Ontologien. Nachfolgende Arbeiten müßten Konzepte erarbeiten, welche die semantikerhaltende Transformation von $\mathcal{I}$-Regelmengen, induziert durch die Transformation von Ontologiedeklaration, behandeln. Schließlich ist noch die Entwicklung von Werkzeugen, welche die hier vorgestellte Ontologie-Beschreibungssprache implementieren, als mögliche Aufgabe für die Zukunft zu nennen.

Literaturverzeichnis

[ABdB+05] Jürgen Angele, Harold Boley, Jos de Bruijn, Dieter Fensel, Pascal Hitzler, Michael Kifer, Reto Krummenacher, Holger Lausen, Axel Polleres, and Rudi Studer. *Web Rule Language (WRL)*. W3C Member Submission. W3C Consortium, September 2005.

[AK00] Colin Atkinson and Thomas Kühne. Meta-Level Independent Modeling. In *International Workshop Model Engineering (in Conjunction with ECOOP'2000)*. Cannes, France, June 2000.

[BBLS04] Andreas Billig, Susanne Busse, Andreas Leicher, and Jörn Guy Süß. Platform Independent Model Transformation Based on Triple. In *Middleware 2004, ACM/IFIP/USENIX International Middleware Conference*, pages 493–512, 2004.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BD98] Gerhard Brewka and Jürgen Dix. Knowledge representation with logic programs. *Lecture Notes in Computer Science*, 1471, 1998.

[Bec04a] Sean Bechhofer. *OWL Web Ontology Language Parsing OWL in RDF/XML*. W3C Working Group Note. W3C Consortium, Januar 2004.

[Bec04b] Dave Beckett. *RDF/XML Syntax Specification (Revised)*. W3C Recommendation. W3C Consortium, Februar 2004.

[BG00] Dan Brickley and R.V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0*. W3C Candidate Recommendation. W3C Consortium, März 2000.

[BG04] Dan Brickley and R.V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. W3C Consortium, Februar 2004.

[BHGS01] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. In *Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence*, volume 2174 of *LNAI*, pages 396–408, Vienna, Sep 2001. Springer-Verlage.

[BHS03] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer, 2003. To appear.

[BLFIM98] T. Berners-Lee, R. Fielding, U. C. Irvine, and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax*. Request for Comments (RFC) 2396. Internet Engineering Task Force (IETF), August 1998.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):35–43, 2001.

[BNES03] Andreas Billig, Lutz Nentwig, Johannes Werner Erdmann, and Kurt Sandkuhl. Mecomp.net - Organizational, Sociological and Technological Aspects of a Community Network in the Field of Education and Employment. In *Euro PDP*, pages 507–516, 2003.

[Bor96] Alexander Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[BPM04] Paul V. Biron, Kaiser Permanente, and Ashok Malhotra. *XML Schema Part 2: Datatypes*. W3C Recommendation. W3C Consortium, October 2004.

[BS02] A. Billig and K. Sandkuhl. Match-Making based on Semantic Nets - The XML-based Approach of BaSeWeP. In R. Tolksdorf and R. Eckstein, editors, *XML Technologien f ur das SemanticWeb (XSW 2002)*, pages 39–52. Gesellschaft für Informatik, 2002.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *OWL Web Ontology Language Reference*. W3C Recommendation. W3C Consortium, Februar 2004.

[CFF+98] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 600–607. AAAI Press, Juli 1998.

[CK00] Wolfram Conen and Reinhold Klapsing. A Logical Interpretation of RDF. In *Linköping Electronic Articles in Computer and Information Science*, volume 5 (2000):nr 013 of *ISSN 1401-9841*. Linköping University Electronic Press, 2000.

[CKW93] Weidong Chen, Michael Kifer, and David Scott Warren. HILOG: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, 1993.

[CW93] Weidong Chen and David Scott Warren. Query Evaluation under the Well Founded Semantics. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC*, pages 168–179. ACM Press, 1993.

[dBLPF05] Jos de Bruijn, Rubén Lara, Axel Polleres, and Dieter Fensel. OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic Web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 623–632, New York, NY, USA, 2005. ACM Press.

[DBSA98] Stefan Decker, Dan Brickley, Janne Saarela, and Jürgen Angele. A query and inference service for rdf. In *QL'98 - Query Languages 1998*. World Wide Web Consortium (W3C), 1998.

[Dec02] Stefan Decker. *Semantic Web Methods for Knowledge Management*. Dissertation, Universität Karlsruhe, 2002.

[DEFS99] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In Robert Meersman, Zahir Tari, and Scott M. Stevens, editors, *DS-8*, volume 138 of *IFIP Conference Proceedings*, pages 351–369. Kluwer, 1999.

[DFvH+00] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge Representation on the Web. In *Proceedings of the 2000 Description Logic Workshop (DL 2000)*, pages 89–98, 2000.

[DLNS98] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating Datalog and Description Logics. *J. Intell. Inf. Syst.*, 10(3):227–252, 1998.

[DSB+05] Stefan Decker, Michael Sintek, Andreas Billig, Nicola Henze, Peter Dolog, Wolfgang Nejdl, Andreas Harth, Andreas Leicher, Susanne Busse, Joern Guy Suess, Zoltan Miklos, Jose-Luis Ambite, Matthew Weathers, Gustaf Neumann, and Uwe Zdun. TRIPLE - an RDF Rule Language with Context and Use Cases. In *W3C Workshop on Rule Languages for Interoperability*, Washington, D.C., USA, 2005.

[DSN02] S. Decker, M. Sintek, and W. Nejdl. The Model-Theoretic Semantics of TRIPLE. Technischer Bericht, Universität Hannover, November 2002. URL: http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2002/triple_semantics.pdf.

[EFT86] Heinz Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Einführung in die mathematische Logik*. Wissenschaftliche Buchgesellschaft Darmstadt, 1986.

[EGM97] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

[EMC+01] H. Ehrig, B. Mahr, F. Cornelius, M. Große-Rhode, and P. Zeitz. *Mathematisch-strukturelle Grundlagen der Informatik*. Springer, Berlin, 2001.

[FHK+97] Jürgen Frohn, Rainer Himmeröder, Paul-Thomas Kandzia, Georg Lausen, and Christian Schlepphorst. FLORID: A Prototype for F-Logic. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K*, page 583. IEEE Computer Society, 1997.

[FM01] R. Fikes and D. McGuiness. An axiomatic semantics for RDF, RDF Schema and DAML+OIL. Technical report, KSL, Stanford University, 2001.

[GB04] Jan Grant and Dave Beckett. *RDF Test Cases*. W3C Recommendation. W3C Consortium, Februar 2004.

[GEF+99] W. E. Grosso, H. Eriksson, R. W. Fergerson, J. H. Gennari, S. W. Tu, and M. A. Musen. Knowledge modelling at the millenium (the design and evolution of protege-2000). In *Proceedings of Knowledge Acqusition Workshop (KAW99)*, 1999.

[GH03] R.V. Guha and Patrick Hayes. *LBase: Semantics for Languages of the Semantic Web*. W3C Working Group Note. W3C Consortium, Oktober 2003.

[GHVD03] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: combining logic programs with description logic. In *WWW*, pages 48–57, 2003.

[GRS00] Günther Görz, C.-R. Rollinger, and J. Schneeberger, editors. *Handbuch der Künstlichen Intelligenz*. Oldenbourg Wissenschaftsverlag, 2000.

[HAMS05] Pascal Hitzler, Jürgen Angele, Boris Motik, and Rudi Studer. Bridging the Paradigm Gap with Rules for OWL. In *W3C Workshop on Rule Languages for Interoperability*, Washington, D.C., USA, 2005.

[Hay04] Patrick Hayes. *RDF Semantics*. W3C Recommendation. W3C Consortium, Februar 2004.

[Hef04] Jeff Heflin. *OWL Web Ontology Language Use Cases and Requirements*. W3C Recommendation. W3C Consortium, Februar 2004.

[HHK$^+$05] Pascal Hitzler, Peter Haase, Markus Krötzsch, York Sure, and Rudi Studer. DLP isn't so bad after all. In *Proceedings of the Workshop OWL - Experiences and Directions, Galway, Ireland*, NOV 2005.

[HM01] Patrick Hayes and Christopher Menzel. A semantics for the Knowledge Interchange Format. In *IJCAI 2001 Workshop on the IEEE Standard Upper Ontology*, 2001.

[HMS04] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In D. Dubois, C. Welty, and M.-A. Williams, editors, *Proceedings of the 9th International Conference on Knowledge Representation and Reasoning (KR2004)*, pages 152–162. AAAI Press, 2004.

[HMS05] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data Complexity of Reasoning in Very Expressive Description Logics. In *IJCAI*, pages 466–471, 2005.

[Hor02a] Ian Horrocks. DAML+OIL: a Description Logic for the Semantic Web. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 25(1):4–9, March 2002.

[Hor02b] Ian Horrocks. DAML+OIL: a Reason-able Web Ontology Language. In *Proc. of EDBT 2002*, number 2287 in Lecture Notes in Computer Science, pages 2–13. Springer, March 2002.

[Hor05] Ian Horrocks. OWL Rules, OK? In *W3C Workshop on Rule Languages for Interoperability*, Washington, D.C., USA, 2005.

[HPS04] Ian Horrocks and Peter F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004.

[HPSBT05] Ian Horrocks, Peter F. Patel-Schneider, Sean Bechhofer, and Dmitry Tsarkov. OWL rules: A proposal and prototype implementation. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):23–40, July 2005.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[HSS06] Pascal Hitzler, York Sure, and Rudi Studer. Description Logic Programs: A Practical Choice For the Modelling of Ontologies. In *Principles and Practices of Semantic Web Reasoning*, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2006.

[HTdSM05] Sandro Hawke, Said Tabet, and Christian de Sainte Marie. Rule Language Standardization / Report. In *W3C Workshop on Rule Languages for Interoperability*, Washington, D.C., USA, 2005. URL: http://www.w3.org/2004/12/rules-ws/report/.

[Ins94] National Information Standards Institute. Guidelines for the construction, format, and management of monolingual thesauri. *NISO Press*, page 69p, 1994. (ANSI/NISO Z39.19-1993).

[KC04] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. W3C Consortium, Februar 2004.

[KdBBF05] Michael Kifer, Jos de Bruijn, Harold Boley, and Dieter Fensel. A Realistic Architecture for the Semantic Web. In Asaf Adi, Suzette Stoutenburg, and Said Tabet, editors, *RuleML*, volume 3791 of *Lecture Notes in Computer Science*, pages 17–29. Springer, 2005.

[KHSV05] Markus Krötzsch, Pascal Hitzler, Michael Sintek, and Denny Vrandecic. Expressive OWL Reasoning with Logic Programs. Technischer Bericht, Universität Karlsruhe, November 2005.

[KL89] Michael Kifer and Georg Lausen. F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. In *SIGMOD Conference*, pages 134–146, 1989.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of ACM*, 42(4):741–843, 1995.

[Kut94] Ralf Kutsche. *A type-oriented approach to the specification and formal semantics of a distributed, heterogeneous object system*. Dissertation, Technische Universitt Berlin, 1994.

[Las98] Ora Lassila. Web metadata: A matter of semantics. *ieee-ic*, 2(4):30–37, 1998.

[LdBPF05] Holger Lausen, Jos de Bruijn, Axel Polleres, and Dieter Fensel. WSML - a Language Framework for Semantic Web Services. In *W3C Workshop on Rule Languages for Interoperability*, Washington, D.C., USA, 2005.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming*. Berlin, Springer, 2nd edition, 1987.

[LR96] Alon Y. Levy and Marie-Christine Rousset. CARIN: A Representation Language Combining Horn Rules and Description Logics. In *ECAI*, pages 323–327, 1996.

[MM04] Frank Manola and Eric Miller. *RDF Primer*. W3C Recommendation. W3C Consortium, Februar 2004.

[Mot05] Boris Motik. On the Properties of Metamodeling in OWL. In *International Semantic Web Conference*, pages 548–562, 2005.

[MSS04] Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. In *International Semantic Web Conference*, pages 549–563, 2004.

[Ode95] James Odell. Meta-modeling. *OOPSLA'95 Workshop on Metamodeling in OO*, 1995.

[PH01] Jeff Z. Pan and Ian Horrocks. Metamodeling architecture of web ontology languages. In *Proc. of the 2001 Int. Semantic Web Working Symposium (SWWS 2001)*, pages 131–149, 2001.

[PSF02] Peter F. Patel-Schneider and Dieter Fensel. Layering the Semantic Web: Problems and Directions. In *Proceedings of the First International Semantic Web Conference ISWC*, volume 2348 of *LNCS*, pages 16–29. Springer Verlag, Juni 2002.

[PSH06] Peter F. Patel-Schneider and Ian Horrocks. Position paper: a comparison of two modelling paradigms in the Semantic Web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 3–12, New York, NY, USA, 2006. ACM Press.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Recommendation. W3C Consortium, Februar 2004.

[SD01] Michael Sintek and Stefan Decker. TRIPLE - An RDF Query, Inference, and Transformation Language. In *Proceedings of the 14th International Conference of Applications of Prolog INAP2001*, Tokyo, Japan, October 2001.

[SD02] Michael Sintek and Stefan Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *Proceedings of International Semantic Web Conference ISWC 2002*. Lecture Notes in Computer Science, Bd. 2342, Springer, 2002.

[SM01] Steffen Staab and Alexander Maedche. Knowledge portals: Ontologies at work. *AI Magazine*, 22(2):63–75, 2001.

[Sow00] John F. Sowa. *Knowledge Representation - Logical, Philosophical, and Computational Foundations*. Brooks/Cole (Thomson Learning), 2000.

[SS04] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.

[SSW94] Konstantinos F. Sagonas, Terrance Swift, and David Scott Warren. XSB as an Efficient Deductive Database Engine. In Richard T. Snodgrass and Marianne Winslett, editors, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994*, pages 442–453. ACM Press, 1994.

[vDKV00] Arie van Deursen, Paul Klint, and Joost Visser. Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices*, 35(6):26–36, 2000.

[vGRS91] Allen van Gelder, Kenneth Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[Vol04] Raphael Volz. *Web Ontology Reasoning with Logic Databases*. Dissertation, Universität Karlsruhe, 2004.

[Vra05] Denny Vrandecic. Explicit knowledge engineering patterns with macros. In Chris Welty and Aldo Gangemi, editors, *Proceedings of the Ontology Patterns for the Semantic Web Workshop at the ISWC 2005*, Galway, Ireland, NOV 2005.

[WSWS01] B. J. Wielinga, A. Th. Schreiber, J. Wielemaker, and J. A. C. Sandberg. From thesaurus to ontology. In *Proceedings of the international conference on Knowledge capture*, pages 194–201. ACM Press, 2001.

[YK00] Guizhen Yang and Michael Kifer. FLORA: Implementing an Efficient DOOD System Using a Tabling Logic Engine. In *Computational Logic*, pages 1078–1093, 2000.

[YK02] Guizhen Yang and Michael Kifer. On the Semantics of Anonymous Identity and Reification. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*. Springer-Verlag, 2002.

[YKZ03] Guizhen Yang, Michael Kifer, and Chang Zhao. Flora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *CoopIS/DOA/ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 671–688. Springer, 2003.