

H. Stuckenschmidt

INFORMATIK IM FOKUS

# Ontologien

Konzepte, Technologien und Anwendungen

2. Auflage

 Springer

# Informatik im Fokus

---

*Herausgeber:*

Prof. Dr. O. P. Günther

Prof. Dr. W. Karl

Prof. Dr. R. Lienhart

Prof. Dr. K. Zeppenfeld

# Informatik im Fokus

---

Weitere Titel der Reihe Informatik im Fokus:  
<http://www.springer.com/series/7871>

Heiner Stuckenschmidt

# Ontologien

Konzepte, Technologien  
und Anwendungen

2. Auflage

 Springer

Prof. Dr. Heiner Stuckenschmidt  
Universität Mannheim  
Institut für Informatik  
A5, 6  
68159 Mannheim  
Deutschland  
heiner@informatik.uni-mannheim.de

*Herausgeber*

Prof. Dr. O. P. Günther  
Humboldt Universität zu Berlin

Prof. Dr. R. Lienhart  
Universität Augsburg

Prof. Dr. W. Karl  
Universität Karlsruhe (TH)

Prof. Dr. K. Zeppenfeld  
Hochschule Hamm-Lippstadt

ISSN 1865-4452

e-ISSN 1865-4460

ISBN 978-3-642-05403-7

e-ISBN 978-3-642-05404-4

DOI 10.1007/978-3-642-05404-4

Springer Heidelberg Dordrecht London New York

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

© Springer-Verlag Berlin Heidelberg 2009, 2011

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

*Einbandentwurf:* KuenkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# Vorwort

Ontologien sind ein Thema, welches sich in der Informatik wachsender Beliebtheit erfreut, seit sich Mitte der achtziger Jahre erstmals große Forschungsprojekte wie Cyc und Wordnet mit der formalen Erfassung von menschlichem Wissen beschäftigten. Inzwischen liefert eine Suche nach dem englischen Plural “ontologies” bei Google beinahe 18,8 Millionen Treffer (Stand 6. September 2010). Diese unerwartete Popularität verdanken Ontologien vor allem neueren Entwicklungen im Bereich des Semantic Web. Der wesentliche Beitrag des Semantic Web besteht hierbei weniger in grundlegenden Erkenntnissen, sondern vielmehr in der Bereitstellung von Technologien, durch welche die Verwendung von Ontologien in Informationssystemen erst möglich gemacht wird. Obwohl dies natürlich einen wesentlichen und vielleicht sogar den entscheidenden Beitrag zum Erfolg von Ontologien als etablierte Technologie darstellt, sollte nicht vergessen werden, dass die grundlegenden Konzepte und Ideen dieser Technologie zumeist schon älter sind und ihren Ursprung in unterschiedlichen Bereichen, wie etwa der Philosophie und der Linguistik haben. Interessanterweise bildeten Arbeiten

im Bereich des Semantic Web zwar gewissermaßen einen Katalysator für die Entwicklung von Ontologie-Technologie, das Web selbst eignet sich jedoch, wie sich inzwischen herausgestellt hat, nur bedingt als überzeugender Anwendungsfall für Ontologien. Solche Anwendungsfälle finden sich in Bereichen, in denen zwar Web-Technologien verwendet werden, diese jedoch nicht das Web im eigentlichen Sinne darstellen. Beispiele erfolgreicher Anwendungen gibt es etwa im Bereich Digitaler Bibliotheken in der Medizin, der automatischen Generierung von Internetportalen aus Daten sowie der Integration von Daten aus unterschiedlichen Anwendungssystemen innerhalb eines Unternehmens.

Das vorliegende Buch ist ein Versuch, im Spannungsfeld zwischen grundlegenden Konzepten aus Philosophie und Linguistik, dem Stand der Technik im Bereich unterstützender Technologien aus der Semantic Web Forschung und vielversprechenden Anwendungsgebieten einen Überblick über die Rolle von Ontologien in der Informationsverarbeitung zu bieten, ohne hierbei Gefahr zu laufen, in wenigen Jahren überholt zu sein. Das Buch richtet sich hiermit an Leser, die ein grundlegendes Interesse an Ontologien als Teil moderner Informationstechnologie haben und sich einen Überblick über das Gebiet verschaffen wollen. Im Rahmen einzelner Themen werden zwar immer wieder Verweise auf spezielle Technologien, wie Sprachstandards zur Darstellung und Infrastrukturen zur Verarbeitung von Ontologien, gegeben, dieses Buch versteht sich jedoch nicht als Anleitung zur Verwendung spezieller Technologien. Vielmehr soll dem Leser Wissen vermittelt werden, welches die Verwendung von Sprachen wie OWL vereinfacht und bestimmte Entscheidungen bei der Entwicklung von Sprachen wie RDF und OWL nachvollziehbar macht. Ausführliche Informationen zur Verwendung dieser populären Sprachen finden sich unter anderem in dem kürzlich im Springer Verlag erschienenen Lehrbuch "Semantic Web"

von Hitzler und anderen, welches an dieser Stelle ausdrücklich empfohlen wird. Wir werden nicht ganz umhin kommen, an bestimmten Stellen konkrete Beispiele zu verwenden, die naturgemäß den aktuellen Stand der Technik widerspiegeln und damit keinen Anspruch auf Allgemeingültigkeit besitzen. Wann immer solche Beispiele verwendet werden, wird auf diese Tatsache hingewiesen. Zudem wird, wenn immer möglich, versucht, von konkreten Ansätzen zu abstrahieren und allgemeine Richtlinien zumindest ergänzend zu erwähnen. Dies gilt insbesondere für die beschriebenen Beispielanwendungen. Im Hinblick auf die Beschreibung konkreter Ontologien ist die Auswahl auf solche Ontologien gefallen, die zumindest in einer bestimmten Community bereits etabliert sind und als De-facto-Standard gelten können.

Der Aufbau des Buches orientiert sich an den drei oben genannten Aspekten, die im Zusammenhang mit Ontologien betrachtet werden müssen: grundlegende Konzepte, unterstützende Technologien und vielversprechende Anwendungen.

**Teil I: Konzepte** Der erste Teil des Buches beschäftigt sich mit grundlegenden Konzepten der Darstellung von Semantik, die ihren Ursprung zum einen in der Philosophie und zum anderen in der Linguistik haben. Entsprechende Theorien werden kurz beschrieben, ohne dabei den eigentlichen Fokus der Informationstechnologie aus dem Auge zu verlieren. Schwerpunkt ist dann die Beschreibung von unterschiedlichen Ansatzpunkten zur Übertragung der generellen Konzepte aus Philosophie und Linguistik auf eine Darstellung von Semantik in der Informationstechnologie. Um dem Leser ein Gefühl für unterschiedliche Ausprägungen dieser Darstellung zu vermitteln, werden in diesem Teil entsprechende Repräsentationen anhand konkreter Beispiele gezeigt.



**Teil II: Technologien** Der zweite Teil des Buches widmet sich Technologien, welche die Erstellung, Verwaltung und Verwendung von Ontologien unterstützen. Der Schwerpunkt liegt hier zum einen im Bereich logischer Formalismen zur Beschreibung von Ontologien. Hierbei geht es um grundlegende Sprachklassen mit unterschiedlichen Eigenschaften im Hinblick auf Ausdrucksmächtigkeit und Möglichkeiten, logisches Schließen zu unterstützen. Außerdem werden die Beziehungen dieser Sprachklassen zu konkreten Ontologiesprachen wie RDF, F-Logic und OWL aufgezeigt. Zum anderen behandelt dieser Teil des Buches Vorgehensmodelle und Technologien, welche die Erstellung von Ontologien unterstützen, wobei zwischen der manuellen Erstellung mit Hilfe entsprechender Editoren und der halbautomatischen Erzeugung von Ontologien aus Texten unterschieden wird.

**Teil III: Anwendungen** Im dritten und letzten Teil dieses Buches wird der Nutzen von Ontologien im Kontext moderner Informationssysteme thematisiert. Hierzu werden typische Anwendungsfälle für Ontologien beschrieben und jeweils mit konkreten Beispielen erfolgreicher Anwendungen oder Produkte ergänzt. Hierbei kommen wiederum nicht nur Anwendungen zu ihrem Recht, die primär aus Entwicklungen im Bereich des Semantic Web hervorgegangen sind. Vielmehr soll gezeigt werden, dass Ontologien eine wichtige Rolle in ganz unterschiedlichen Anwendungsbereichen, wie der Datenintegration und der Informationssuche, spielen können.

Dieses Buch beinhaltet Erkenntnisse aus Arbeiten in unterschiedlichen Projekten und Themen, in deren Kontext ich ganz unterschiedliche Sichtweisen auf Ontologien kennengelernt habe. Naturgemäß wurde die hier dargestellte Sichtweise durch verschiedene Forscher geprägt, die an den entsprechenden Arbeiten beteiligt waren. Diese alle aufzuzählen, würde den Rah-

men sprengen. Erwähnt seien jedoch die folgenden Forscher, die für mich jeweils für eine ganz bestimmte Sichtweise auf das Thema Ontologien stehen:

- Barry Smith und Nicola Guarino mit ihren Arbeiten zu grundsätzlichen Fragen formaler Ontologien in der Tradition philosophischer Arbeiten zu dem Thema.
- Das EMTREE-Team bei Elsevier, von dem ich gelernt habe, dass es rund um das Konzept eines Thesaurus eine ganz eigene Community mit völlig anderen Sichtweisen auf Semantik gibt.
- Frank van Harmelen und Ian Horrocks, die Ontologien in der Tradition von Wissensrepräsentation und logischem Schließen sehen und Ausgangspunkt meiner eigenen Arbeiten in diesem Bereich waren.
- Alan Rector und Mark Musen als Vertreter einer angewandten Sicht, die Ontologien als Mittel zur Lösung realer Probleme begreifen, die sich aus den Anforderungen konkreter Anwendungsbereiche, in diesem Fall der Medizin, ergeben.

Einen weiteren wesentlichen, wenn auch indirekten Anteil zu diesem Buch hat die Deutsche Forschungsgemeinschaft, die mir durch ein Emmy-Noether Stipendium die notwendige finanzielle Unterstützung gewährt hat, sowie meine Mitarbeiterinnen und Mitarbeiter, die mich in meiner Arbeit so gut unterstützten, dass neben dem Alltagsgeschäft noch Zeit blieb, ein solches Buch in Angriff zu nehmen.

Mannheim, April 2010

*Heiner Stuckenschmidt*

# Inhaltsverzeichnis

## Teil I Konzepte

<b>1</b>	<b>Symbole, Objekte und Konzepte</b> .....	5
1.1	Konzepte und das Universalienproblem .....	8
1.2	Objekte und Konzepte .....	11
1.3	Symbole und Konzepte .....	16
1.4	Anmerkungen zu Begrifflichkeiten .....	22
<b>2</b>	<b>Repräsentation von Bedeutung</b> .....	25
2.1	Semantische Netze .....	26
2.2	Logik .....	34
<b>3</b>	<b>Beispiele</b> .....	53
3.1	WordNet – ein semantisches Lexikon .....	55
3.2	UMLS – Unified Medical Language System ...	65
3.2.1	Der Metathesaurus .....	67
3.2.2	Das semantische Netz .....	74
3.3	Die Suggested Upper Merged Ontology .....	77
3.3.1	SUO-KIF und logische Definitionen ...	80

3.3.2	SUMO als Basis-Ontologie .....	83
3.3.3	Verbindung zu WordNet .....	88

## Teil II Technologien

<b>4</b>	<b>Ontologiesprachen</b> .....	95
4.1	Logik-Programme .....	99
4.1.1	RDF-Schema als Regelsystem .....	102
4.1.2	F-Logic .....	115
4.1.3	Fazit .....	126
4.2	Beschreibungslogik .....	127
4.2.1	Grundlagen von Beschreibungslogiken .....	128
4.2.2	OWL als Beschreibungslogik .....	146
<b>5</b>	<b>Erstellen von Ontologien</b> .....	155
5.1	Modellierung von Ontologien .....	157
5.1.1	Vorgehen .....	158
5.1.2	Beschreibungsmuster .....	172
5.2	Ontologie-Editoren .....	184
5.2.1	Visualisierung von Ontologien .....	186
5.2.2	Manipulation von Definitionen .....	192
5.2.3	Inferenzunterstützung .....	198

## Teil III Anwendungen

<b>6</b>	<b>Datenintegration</b> .....	211
6.1	Heterogenität von Informationsquellen .....	213
6.1.1	Strukturelle Heterogenität .....	214
6.1.2	Semantische Heterogenität .....	217
6.2	Ontologien für die Datenintegration .....	220
6.2.1	Neutrale Darstellung von Domänenstrukturen .....	221

6.2.2	Explizite Darstellung von Annahmen .....	225
6.2.3	Konsistenzprüfung .....	227
6.3	Beispiel Ontobroker .....	229
<b>7</b>	<b>Informationssuche .....</b>	<b>233</b>
7.1	Klassische Dokumentensuche .....	234
7.1.1	Relevanz von Dokumenten .....	235
7.1.2	Das Vektorraum-Modell .....	239
7.1.3	Probleme des klassischen Modells .....	242
7.2	Thesaurus-basierte Suche .....	243
7.2.1	Thesaurus-basierte Verschlagwortung .....	244
7.2.2	Anfrage-Normalisierung und -expansion .....	250
7.3	Beispiel: Das DOPE-System .....	253
<b>8</b>	<b>Zusammenfassung und Literatur .....</b>	<b>259</b>
	Literaturverzeichnis .....	267
	<b>Sachverzeichnis .....</b>	<b>271</b>

# **Teil I**

## **Konzepte**

In diesem ersten Teil des Buches werden grundlegende Konzepte eingeführt, die notwendig sind, um die Rolle von Ontologien in Informationssystemen zu verstehen. Hierzu holen wir etwas weiter aus und beziehen uns auf Erkenntnisse aus der Philosophie und der Linguistik, die sich mit der Entstehung, Beschreibung und Kommunikation von Bedeutung beschäftigen. Insbesondere werden wir uns in Kapitel 1 kurz mit den Grundgedanken der Ontologie als philosophischer Disziplin und ihrem Versuch, Objekte der realen und gedachten Welt in Kategorien zu unterteilen, sowie deren Eigenschaften und Abhängigkeiten zu analysieren, und der Semantik als Teilbereich der Linguistik, die versucht, die Bedeutung von Worten zu erfassen, beschäftigen. Die hier diskutierten Grundüberlegungen führen uns direkt zu unterschiedlichen Ansätzen der Repräsentation von Bedeutung, wie wir sie von Ontologien in der Informatik kennen. Kapitel 2 beschäftigt sich mit entsprechenden Ideen. Wir unterscheiden hierbei zwischen linguistisch inspirierten Ansätzen, die versuchen, Bedeutung über Worte und spezielle Relationen zwischen Worten, wie etwa Synonymie, darzustellen, und Ansätzen, die zumeist auf der Verwendung logischer Definition als Träger von Bedeutung beruhen. Diese Unterscheidung schlägt sich bis heute im Stand der Forschung im Bereich der Informatik nieder, da es bisher nicht ausreichend Versuche gibt, diese beiden grundlegenden Formen der Darstellung von Bedeutung miteinander zu verbinden. In Kapitel 3 werden typische Beispiele von Ontologien gezeigt, die jeweils die eine oder andere Form der Beschreibung von Bedeutung verwenden. Dabei werden auch Unterschiede diskutiert, die sich aus dem Fokus der semantischen Beschreibungen in einer Ontologie ergeben. Insbesondere können wir zwischen Ontologien unterscheiden, die möglichst umfassend versuchen, unabhängig von einem bestimmten Anwendungsfeld die Bedeutung allgemeiner Begrifflichkeiten zu erklären. Demgegenüber stehen Ontologien, die speziell auf ein

bestimmtes Anwendungsgebiet, etwa die Medizin zugeschnitten sind und deren Ziel es unter anderem ist, bestehende Fachbegriffe zu beschreiben. Die in Kapitel 3 gezeigten Beispiele realer Ontologien bilden auch die Grundlage für die Beschreibung typischer Anwendungen von Ontologien in Teil III dieses Buches.



# Kapitel 1

## Symbole, Objekte und Konzepte

Um die potenzielle Bedeutung von Ontologien in der Informationsverarbeitung zu verstehen, muss man sich klarmachen, was die grundlegenden Aufgaben und Probleme der Informationsverarbeitung sind: nämlich bestimmte Ausschnitte der realen Welt in eine geeignete Darstellungsform zu überführen, die mit Hilfe des Computers manipuliert werden kann, um mit dieser Veränderungen in der realen Welt abzubilden. Dies können wir am Beispiel des Electronic Banking genauer betrachten. Ein System in diesem Bereich muss bestimmte Objekte der Welt – Personen, Organisationen, Konten, Aktien, Geldbeträge usw. – abbilden. Diese Objekte können physikalische Objekte sein, wie etwa Personen, oder aber abstrakte Objekte, wie Konten. Häufig ist hierbei der Übergang zwischen konkreten und abstrakten Objekten fließend. Eine Aktie zum Beispiel kann ein physikalisches Objekt, nämlich ein Anteilschein in Form eines Dokuments, oder aber ein abstrakter Anteil an einem Unternehmen sein. Operationen, die auf den im System enthaltenen Repräsentationen von Personen, Konten usw. ausgeführt werden, spiegeln hierbei Veränderungen der realen Welt wider. Die Überweisung eines

Geldbetrags, dargestellt durch die Subtraktion eines bestimmten Betrags von einer Zahl im System, sowie die Addition des gleichen Betrags auf eine andere Zahl<sup>1</sup> hat hierbei eine konkrete Veränderung der Besitzverhältnisse in der realen Welt zur Folge.

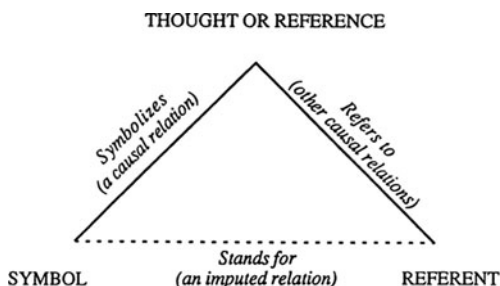
Da, wie das Beispiel zeigt, Informationsverarbeitung unmittelbaren Einfluss auf die reale Welt haben kann, ist die korrekte und angemessene Abbildung der realen Welt in das entsprechende Programm ein zentrales Problem. Die Definition einer solchen Abbildung wird hierbei dadurch erschwert, dass es in der Regel eine Vielzahl unterschiedlicher Möglichkeiten gibt, diese Abbildung vorzunehmen. Welche Darstellung die „beste“ ist, lässt sich oft nicht sagen und hängt von unterschiedlichen Faktoren ab: dem Ausschnitt der betrachteten Welt, der Verfügbarkeit von Informationen, von der zu erfüllenden Aufgabe, von technischen Möglichkeiten usw. Diese Fülle an Faktoren führt dazu, dass Systeme, die ganz ähnliche Aufgaben erfüllen, sehr unterschiedliche Darstellungen der realen Welt verwenden. Unterschiede zwischen verwendeten Darstellungen können rein struktureller Natur sein, wie etwa die getrennte Speicherung von Vor- und Nachnamen versus der gemeinsamen Speicherung in einem einzigen Datenfeld. Mögliche Unterschiede können aber auch viel grundlegenderer Natur sein. Eine Frage ist zum Beispiel, ob Organisationen und Personen als Inhaber von Konten einheitlich oder in unterschiedlichen Repräsentationen dargestellt werden. Erfolgt eine einheitliche Darstellung, schließt sich die Frage an, ob Namen von Organisationen und Personen gleich behandelt werden können (was ist der Vorname eines Unternehmens?) und was die Gemeinsamkeiten von Personen und Organisationen sind, die eine solche Gleichbehandlung rechtfertigen. Aus rechtlicher Sicht sind beide juristische Personen. Dennoch bestehen Unterschiede in der Behandlung von Privat- und

---

<sup>1</sup> Der Vorgang bei einer Umbuchung ist natürlich viel komplexer, was aber an dieser Stelle nichts zur Sache tut.

Firmenkunden, die sich nicht nur aus strategischen Überlegungen der entsprechenden Bank, sondern auch aus gesetzlichen Vorgaben ergeben (Stichwort Beratungspflicht). Schon an diesem einfachen Beispiel zeigen sich die Schwierigkeiten, denen sich die Informationsverarbeitung tagtäglich stellen muss. Hinzu kommt der Trend der Vernetzung von Informationen – Daten über Konten werden nicht nur bei der entsprechenden Bank, sondern auch bei der Schufa und neuerdings von diversen Behörden geführt. Verwenden die Systeme Darstellungen, die von denen der entsprechenden Bank abweichen, wird der Austausch von Informationen erschwert und erfordert ein genaueres Verständnis der jeweils gewählten Darstellungsformen.

Wir können demnach feststellen, dass die Abbildung realer Objekte auf eine entsprechende Darstellung auf der damit verbundenen Erwartungen bezüglich der Natur der wiedergegebenen Objekte beruht. Dieser komplexe Zusammenhang wurde bereits Anfang des Jahrhunderts im Bereich der Semiotik, der Lehre von Zeichen und ihrer Bedeutung, untersucht. Entsprechende Überlegungen finden sich schon bei Aristoteles, eine explizite Beschreibung der Beziehung zwischen einer



**Abb. 1.1** Das semiotische Dreieck

Darstellung (in Form eines Symbols), dem realen Objekt und den damit verbundenen Erwartungen findet sich bei Odgen und Richardson in Form des sogenannten semiotischen Dreiecks (vgl. Abb. 1.1). Ausgangspunkt der Betrachtung sind hierbei Symbole in einer bestimmten Darstellungsform. Diese Symbole beziehen sich auf Objekte in der realen Welt, die in der Abbildung als Referenten bezeichnet werden. In unserem Beispiel wäre die Kundennummer ein solches Symbol, welches sich auf den entsprechenden Kunden, sei es eine Person oder eine Organisation, bezieht. Als drittes Element führen Odgen und Richardson die Erwartungen ein, die der Betrachter mit dem Objekt bzw. dem Symbol verbindet. Diese werden hier als „Thought or Reference“ bezeichnet. In neueren Arbeiten wird diese Bezeichnung häufig durch den Begriff des Konzeptes ersetzt, eine Bezeichnung, die auch im Bereich der Informationsverarbeitung gebräuchlich ist, um Klassen von Objekten und deren typische Eigenschaften zu beschreiben. Die Rolle der Ontologien in der Informationsverarbeitung ist nun, grob gesagt, die Formalisierung dieses dritten Elements, welches die Verbindung zwischen Symbolen einer informationstechnischen Darstellung und den Erwartungen an das hierdurch dargestellte Objekt herstellt. Um diese Rolle deutlicher zu verstehen, ist eine genauere Betrachtung dieses dritten Elements sowie dessen Relationen zu Objekten auf der einen und Symbolen auf der anderen Seite erforderlich.

## 1.1 Konzepte und das Universalienproblem

Um die Relation zwischen Symbolen, Objekten der realen Welt und Konzepten zu verstehen, ist ein kleiner Exkurs in den Bereich der philosophischen Ontologie notwendig. Die Untersuchung der Natur von Konzepten und deren Rolle bei der

Beschreibung der Welt ist eine zentrale Fragestellung der Ontologie, häufig als „Universalienproblem“ bezeichnet. Um dies zu veranschaulichen, betrachten wir natürlichsprachliche Aussagen wie die folgenden:

„Die Parkbank ist grün.“

„Herr Meier ist ein Kunde.“

Versucht man einen Bezug zwischen diesen Sätzen und der realen Welt herzustellen, um deren Bedeutung zu erfassen, so stellt man fest, dass dies nicht ohne Weiteres möglich ist. Die Interpretation der Satzteile „Herr Meier“ sowie „Die Parkbank“ ist relativ einfach, da wir diese auf konkrete Objekte in der Welt, nämlich die entsprechende Person bzw. das entsprechende Sitzmöbel abbilden können. Auch die Interpretation der Sätze als Ganzes ist leicht möglich, indem diesen ein Wahrheitswert zugeordnet wird. Wir betrachten die reale Welt und entscheiden, ob die entsprechende Aussage wahr ist, ob also die bezeichnete Bank tatsächlich grün bzw. Herr Meier tatsächlich ein Kunde ist. Schwieriger gestaltet sich die korrekte Abbildung der Eigenschaften der Parkbank bzw. der Eigenschaften von Herrn Meier auf die reale Welt. In beiden Fällen gibt es kein konkretes Objekt, welches Ziel der Abbildung ist, da es in der realen Welt weder das Objekt „grün“ noch das Objekt „Kunde“ gibt. Es handelt sich hierbei vielmehr um eine allgemeine Eigenschaft, die verschiedene Objekte gemeinsam haben können. Solche gemeinsamen Eigenschaften werden auch Universalien genannt, da sie kein Individuum, sondern eine universellere Beschreibung darstellen. Konkrete Objekte können hierbei als Beispiele für diese universellen Beschreibungen dienen, etwa Herr Meier als Beispiel für einen Kunden. Während es unstrittig ist, dass universelle Beschreibungen in der Sprache ihren festen Platz haben, herrscht in der Philosophie Uneinigkeit darüber, ob Universalien auch in der realen Welt existieren oder ob die entsprechenden Bezeichnungen stets Mengen von Objekten bezeichnen.

In den meisten Fällen ist die Interpretation von Universalien als Mengen von Objekten sinnvoll. Die Universalie „Kunde“ zum Beispiel ist durch die Menge aller Objekte, für welche die Aussage „...ist ein Kunde“ wahr ist, angemessen repräsentiert. Wie wir später sehen werden, findet sich diese Behandlung von Universalien als Objektmengen auch in entsprechenden Repräsentationssprachen wieder. Es gibt jedoch auch Fälle, in denen die Universalie wie ein Objekt behandelt wird. Ein Beispiel ist der folgende Satz:

„Die Armut ist angestiegen.“

Die Universalie „Armut“ wird hier wie ein Objekt verwendet. Insbesondere wird eine Aussage über die Armut gemacht, der ein Wahrheitswert zugeordnet werden kann, ohne dass es möglich ist, diese auf ein konkretes Objekt abzubilden. Auch gibt es nicht wirklich eine Menge von Armutsobjekten, die gemeinsam die Menge der Armut bilden könnten, dennoch lässt sich die Wahrheit der Aussage überprüfen. Es ist also sinnvoll, auch Universalien als mögliche Referenten im Sinne der Terminologie von Abb. 1.1 zu betrachten, die nicht ohne Weiteres als Menge von Objekten angesehen werden können. Dies bedeutet also, dass Symbole im Prinzip für konkrete Objekte, Mengen von Objekten, aber auch für abstrakte Begriffe wie eben die oben genannte Armut stehen können. In letzteren beiden Fällen sprechen wir auch von Konzepten, die in Ermangelung eines konkreten Referenzobjektes nicht immer eindeutig zu fassen sind. Eine sinnvolle Verwendung des entsprechenden Symbols ist daher nur deshalb möglich, weil, wie oben angedeutet, mit dem entsprechenden Konzept eine gewisse Erwartung verbunden ist, die sich in der Regel aus der Erfahrung bzw. der Sozialisierung des Betrachters sowie häufig auch aus dem Kontext der Verwendung eines Begriffes ergibt.

## 1.2 Objekte und Konzepte

Eine weitere Fragestellung, mit der sich die Philosophie beschäftigt, ist der Gedanke, welche Arten von Objekten es in der realen Welt gibt. Hierbei werden Objekte in der größten allgemein denkbaren Form betrachtet. Diese werden auch als Entitäten bezeichnet und umfassen obiger Argumentation folgend sowohl konkrete Objekte wie Personen als auch abstrakte Objekte (Universalien) wie Organisationen oder Konten aus unserem Beispiel. Die Betrachtungen umfassen hierbei zum einen die Frage, was allen Objekten gemeinsam ist, also im Grunde, was Existenz eigentlich bedeutet, zum anderen aber auch die Frage, was Objekte voneinander unterscheidet. Diese Unterscheidung von Objektarten ist eng verbunden mit dem dritten Element des semiotischen Dreiecks, welches die Erwartungen an die Natur eines dargestellten Objektes repräsentiert. Diese Erwartungen ergeben sich direkt aus den typischen Eigenschaften eines Objektes, welches es von anderen Arten von Objekten unterscheidet. So erwartet man von einer Person, die ein Konto besitzt, dass sie auch einen Vornamen hat. Von einer Firma als Inhaber eines Kontos erwartet man hingegen zum Beispiel einen Eintrag ins Handelsregister. Von diesem wiederum erwartet man, dass er die entsprechende Firma eindeutig identifiziert, was man von dem Vornamen einer Person nicht erwarten würde. Die eindeutige Identifizierbarkeit über einen Eintrag ins Handelsregister ist also eine Eigenschaft, die ein Objekt von der Art Unternehmen von einem Objekt der Art Person unterscheidet. Während die Frage nach den Gemeinsamkeiten aller existierenden Objekte sich im Kontext der Informationsverarbeitung nicht stellt – es sollen ja stets Objekte abgebildet werden, von deren Existenz wir wissen oder ausgehen –, ist die Frage, was Objekte und damit unsere Erwartungen an sie unterscheidet, wesentlich, da sie direkten Einfluss auf die Gestaltung der Abbildung

der realen Welt in eine informationstechnische Darstellung hat. Beim Entwurf einer Datenbank mit Konten und Kontoinhabern ist es, wie man leicht einsehen kann, von entscheidender Bedeutung, ob und wie ein Kontoinhaber eindeutig identifiziert werden kann. Im Falle von Unternehmen könnte dies demnach über den Eintrag ins Handelsregister erfolgen, im Falle von Privatpersonen ist eine andere Form der Identifizierung notwendig. Eine hiermit eng verbundene Frage, die auch im Rahmen der klassischen Ontologie eine Rolle spielt, ist die nach der Veränderlichkeit bestimmter Eigenschaften. Das Geburtsdatum einer Person zum Beispiel wird sich nicht mehr ändern. Es kann, eine korrekte Erfassung vorausgesetzt, für alle Zeit als gültig angesehen werden. Ganz anders verhält es sich mit dem Namen eines Unternehmens, der sich durchaus ändern kann, ohne dass dies das Unternehmen an sich in Frage stellt.

Im Gegensatz zu solchen doch recht konkreten Überlegungen zu Objekten und ihren Eigenschaften, beschäftigt sich die Philosophie traditionell eher mit grundlegenden Unterscheidungen, wie der zwischen materiellen und immateriellen Gegenständen, und versucht, sogenannte Kategoriensysteme zu entwickeln, die Objekte aufgrund ihrer Eigenschaften in unterschiedliche Kategorien einteilen. Kategoriensysteme sollen hierbei in der Philosophie in der Regel umfassend sein, d.h., Kategoriensysteme sollen die Menge aller denkbaren realen Objekte beschreiben und sich nicht auf ein bestimmtes Anwendungsgebiet, wie etwa das Electronic Banking in unserem Beispiel, beschränken. Außerdem ergibt sich durch das erklärte Ziel der Unterscheidung von Objekten die Notwendigkeit, Kategorien so zu definieren, dass eine eindeutige Zuordnung von Objekten zu Kategorien besteht.

Abbildung 1.2 zeigt unterschiedliche Darstellungen des von Porphyry vorgeschlagenen Kategoriensystems für Objekte, welches auf Aristoteles zurückgeht. Obwohl dieses System inzwischen überholt ist, hilft es doch, die grundlegenden Ideen eines



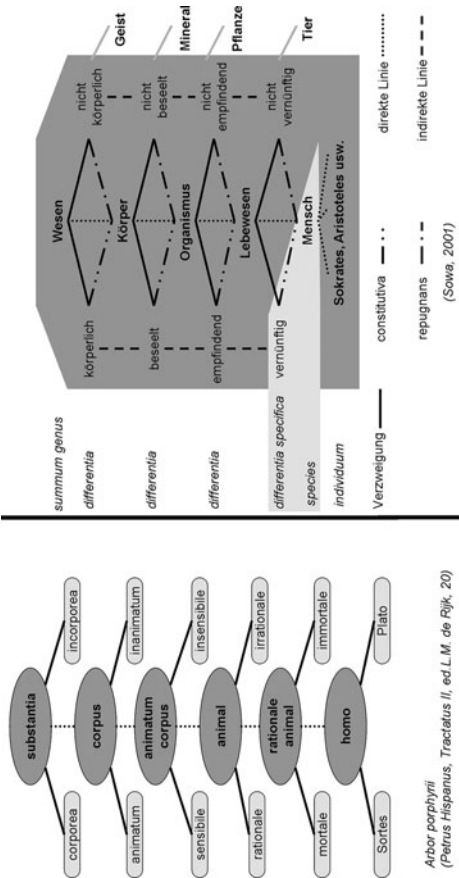


Abb. 1.2 Porphyrys Kategoriensystem für „Substanz“ nach Petrus Hispanus und Sowa

Kategoriensystems genauer zu betrachten. Insbesondere zeigt sich an dieser Darstellung sehr schön das Prinzip von Kategoriensystemen, Objekte anhand charakteristischer Eigenschaften zu klassifizieren. Im Kategoriensystem von Porphyry werden Objekte der Kategorie „Substanz“ aufgrund bestimmter differenzierender Eigenschaften in weitere Unterkategorien aufgeteilt. Die erste dieser als „differentiae“ bezeichneten Eigenschaften ist hier die Körperlichkeit. Substanzen, die nicht körperlich sind, fallen in die Kategorie „Geist“, körperliche Substanzen werden anhand weiterer Eigenschaften klassifiziert. Die Körperlichkeit dient hierbei als eindeutiges Kriterium und dient sowohl der Zuordnung von Objekten zu einer Klasse (ist eine Substanz nicht körperlich, fällt sie automatisch in die Kategorie Geist), als auch dem Ausschluss von Objekten von einer Kategorie (ist etwas nicht körperlich, so kann es nicht zu der Kategorie der Körper gehören). Umgekehrt erlaubt die Zugehörigkeit eines Objektes zu einer Kategorie auch Rückschlüsse auf (einige von) dessen Eigenschaften. So besitzt jedes Objekt der Kategorie „Geist“ notwendigerweise die Eigenschaft, nicht körperlich zu sein. Die „differentiae“ beschreiben demnach notwendige und hinreichende Bedingungen für die Zugehörigkeit zu einer bestimmten Kategorie. Ein weiteres Phänomen, welches sich gut anhand des Kategoriensystems in Abb. 1.2 erklären lässt, ist die Vererbung von Eigenschaften an Unterkategorien. Betrachten wir die Kategorie „Mensch“ so stellen wir fest, dass Objekte dieser Kategorie automatisch auch in den Kategorien Lebewesen, Organismus usw. enthalten sind. Diese nach unserer Intuition selbstverständliche Tatsache ergibt sich im Kategoriensystem zwangsläufig aus der oben beschriebenen Definition von Kategorien über charakteristische Eigenschaften. Wurde ein Objekt als Mensch klassifiziert, so ist dies stets Folge einer Entscheidungskette, in der die vorhandenen „differentiae“ überprüft und das Objekt den entsprechenden Kategorien

zugewiesen wird. Im Falle des Objektes „Aristoteles“ wird zunächst die Körperlichkeit überprüft. Aufgrund des positiven Ergebnisses dieser Prüfung wird Aristoteles als Körper eingestuft. Im nächsten Schritt folgt die Überprüfung der Beseeltheit, welche Aristoteles als Organismus klassifiziert, usw. Dies bedeutet aber auch, dass jedes Objekt, das schließlich der Kategorie Mensch zugeordnet wird, bereits die Tests aller übergeordneten Kategorien bestanden hat und dementsprechend ebenfalls die charakteristischen Eigenschaften dieser Kategorien erfüllt. Die Verwendung charakteristischer Eigenschaften zur Unterscheidung von Objekten ist ein Prinzip, welches sich direkt auf unser Beispiel übertragen lässt. Um entscheiden zu können, ob es notwendig ist, zwischen Personen und Organisationen zu unterscheiden, sind zunächst charakteristische Eigenschaften zu bestimmen. Anschließend kann überprüft werden, ob Personen und Organisationen sich im Bezug auf diese Eigenschaften unterscheiden. In unserem Fall könnte eine maßgebliche Eigenschaft die eindeutige Identifizierbarkeit über den Namen sein.

Die Betrachtung der Prinzipien, die einem Kategoriensystem zugrunde liegen, führt uns zu zwei wesentlichen Ansätzen, Kategorien zu beschreiben: zum einen über die Menge der Objekte, die zu der entsprechenden Kategorie gehören (extensionale Beschreibung), oder aber über die Eigenschaften, die allen Objekten der entsprechenden Kategorie gemeinsam sind (intensionale Beschreibung). Beide Arten der Beschreibung besitzen Vor- und Nachteile. Wollen wir beispielsweise das Konzept „Mensch“ beschreiben, so ist es relativ einfach, dies anhand der im System verwendeten Eigenschaften zu tun: Menschen sind körperlich, beseelt, empfindend und vernünftig. Eine Beschreibung der Kategorie ist wesentlich schwerer, da hierfür die mehr als fünf Milliarden Menschen einzeln aufgezählt werden müssten. Dabei kommt erschwerend hinzu, dass sich diese Menge ständig ändert. Auf der anderen Seite ist die Beschreibung der

Kategorie „Kreditkartentyp“ wohl in Form einer Aufzählung vorhandener Kartentypen einfacher zu bewerkstelligen als durch eine Beschreibung der charakteristischen Eigenschaften.

Das Dilemma, welches stets mit der Möglichkeit der Definition von Konzepten verbunden ist, besteht darin, dass es beinahe unendlich viele Möglichkeiten der Unterscheidung und somit eine nahezu unübersichtliche Anzahl potenzieller Konzepte gibt. Um eine Überfrachtung zu vermeiden, folgt man in der Ontologie dem Prinzip von Ockhams Rasiermesser (*Occams Razor*). Dieses Prinzip besagt im Wesentlichen, dass nur notwendige Unterscheidungen getroffen werden sollen. Während im Allgemeinen schwer festzulegen ist, wann eine Unterscheidung notwendig ist und wann nicht, fällt dies im Kontext einer konkreten Aufgabenstellung wie der formalen Beschreibung einer Anwendung, zum Beispiel dem Online Banking, leichter, da man anhand der geforderten Funktionalität entscheiden kann, welche Unterscheidungen notwendig sind und welche nicht. Diese Tatsache wurde ja zu Beginn dieses Kapitels bereits anhand des Beispiels Privatkunden versus Firmenkunden kurz angesprochen. Wie wir in [Kapitel 5](#) sehen werden, stellt das Prinzip der Notwendigkeit von Unterscheidungen auch eine wichtige Basis für strukturierte Vorgehensweisen bei der Erstellung von Ontologien dar.

### 1.3 Symbole und Konzepte

Die parallele Betrachtung unseres Beispiels und des Kategoriensystems von Porphyry zeigt uns bereits ein weiteres grundlegendes Problem, mit welchem wir bei der Abbildung der Realität zu kämpfen haben, nämlich der Ambiguität von Symbolen, in diesem Fall von Worten, die wir zur Beschreibung von Objekten

und Kategorien verwenden. In unserem Beispiel sprachen wir über Personen. Im Kategoriensystem taucht der Begriff „Mensch“ auf. Obwohl diese beiden Begriffe unterschiedlich sind, würden wir doch in der Regel davon ausgehen, dass sie die gleiche Kategorie beschreiben, nämlich die der körperlichen, beseelten, empfindenden und vernünftigen Substanzen, um es mit Porphyry auszudrücken. Da wir uns bei der Beschreibung der realen Welt nicht nur in der Literatur, sondern auch in der Informationstechnik so gut wie immer Elementen der natürlichen Sprache bedienen, und sei es nur als Bezeichnungen für bestimmte Datenstrukturen, ist es sinnvoll, sich mit dem Bereich der linguistischen Semantik auseinanderzusetzen, welche sich genauer mit der Relation zwischen Begriffen und deren Bedeutung – weitestgehend im Sinne des Kategorienbegriffes aus der Philosophie – beschäftigt.

In der Linguistik wird im Allgemeinen davon ausgegangen, dass die Fähigkeit des Menschen über Sprache zu kommunizieren auf einer abstrakten Grammatik beruht. Diese Grammatik besteht aus einem Lexikon, welches alle Wörter der jeweiligen Sprache enthält, sowie einer strukturellen Komponente, die festlegt, in welcher Weise Worte aus dem Lexikon zu sinnvollen Sätzen kombiniert werden können. Das Lexikon enthält hierbei neben Informationen über Schreibweise und Aussprache von Worten auch Informationen über deren Bedeutung und bildet damit die Grundlage für das Sprachverstehen. Das Lexikon ist hierbei nicht bloß eine Ansammlung von Begriffen. Der Grund hierfür ist die Tatsache, dass es keine eindeutige Beziehung zwischen Begriffen und deren Bedeutung im Sinne der oben erwähnten Konzepte gibt. Vielmehr kann der gleiche Begriff mehrere Bedeutungen haben. Auf der anderen Seite können mehrere unterschiedliche Begriffe das gleiche Konzept repräsentieren. So hat der Begriff „bank“ in der englischen Sprache mindestens zehn unterschiedliche Bedeutungen. Neben der Bank im

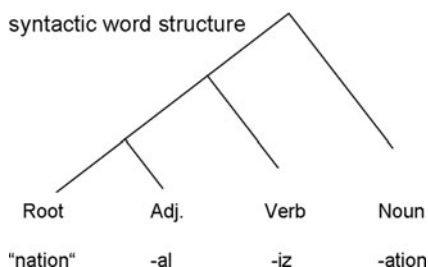
Sinne eines Geldinstituts steht der Begriff „bank“ unter anderem auch für das Bankgebäude oder für den Vorgang des Einzahlens auf ein Bankkonto. Diese Bedeutungen haben zwar alle etwas mit Geldgeschäften zu tun, wodurch sich die Verwendung des gleichen Begriffs erklärt, aus der Sicht der philosophischen Ontologie bezeichnet der Begriff jedoch völlig unterschiedliche Konzepte. Vergleicht man etwa die „Bank“ im Sinne des Geldinstituts mit „Bank“ im Sinne des Bankgebäudes, so wird schnell klar, dass diese ganz unterschiedliche charakteristische Eigenschaften aufweisen. So steht das Bankgebäude an einem eindeutigen Ort und hat eine exakt bestimmte Ausdehnung, die Bank als Organisation hingegen ist ein abstraktes Gebilde, dessen räumliche Position sich nicht bestimmen lässt. Der Begriff „to bank“ wiederum bezeichnet eine Tätigkeit und unterscheidet sich hierdurch von den anderen.

Um diesen Unterschieden Rechnung zu tragen, besitzen semantische Lexika in der Linguistik eine komplexe Struktur, die eine eindeutige(re) Zuordnung von Begriffen zu Konzepten erlauben soll. Eine erste grundlegende Strukturierung besteht in der Verwendung unterschiedlicher Worttypen. Die gängigsten dieser Worttypen sind Substantive und Verben sowie Adjektive und Adverbien. Im Kontext von Ontologien sind hierbei vor allem Substantive von Bedeutung, da diese häufig für bestimmte Kategorien von Objekten stehen, wie man bereits am Kategoriensystem von Porphyry sehen konnte. Da Substantive in vielen Sprachen in unterschiedlichen Formen vorkommen können, je nachdem, in welchem Zusammenhang der Begriff genannt wird, bilden nicht alle Vorkommen von Substantiven eine geeignete Grundlage für die Identifizierung der Bedeutung. So kann der Begriff „Betrag“ auch in den Formen „Beträge“ (Plural) oder „Betrags“ (Genitiv) auftreten. Als Antwort auf dieses Problem, welches im Kontext von Verben in noch verschärfter Form auftritt, wird in der Linguistik von sogenannten „Lexemen“

ausgegangen. Lexeme sind Grundformen von Worten, die unabhängig von grammatikalischen Variationen sind. Im obigen Beispiel haben wir es mit dem Lexem BETRAG zu tun. Alle genannten Worte sind Ableitungen dieses Lexems.

Unglücklicherweise löst die Verwendung von Lexemen anstelle von Worten nicht das Problem der Zuordnung von Begriffen zu Bedeutungen im Sinne der Ontologie. Die Ursachen hierfür liegen zum einen in der Sprachmorphologie, insbesondere die Möglichkeit der Bildung von komplexen Begriffen durch die Modifikation eines Lexems durch ergänzende Wortteile, die unter Umständen die Bedeutung des Lexems grundlegend verändern können. Zum anderen besteht auch zwischen Lexemen und Kategorien im Sinne der Ontologie kein eindeutiger Zusammenhang, wie schon das Beispiel der „Bank“ zeigt. Im Folgenden betrachten wir diese beiden Ursachen etwas genauer.

Die Morphologie beschäftigt sich grob gesagt mit der Zusammensetzung von Worten aus einzelnen Wortbestandteilen, die auch Morpheme genannt werden. Die oben genannten Lexeme sind hierbei eine spezielle Art von Morphemen, welche den Ausgangspunkt für die Bildung komplexer Begriffe bilden. Abbildung 1.3 zeigt am Beispiel des englischen Begriffs



**Abb. 1.3** Morphologische Variationen eines Grundbegriffs

„nationalization“, wie ein komplexer Begriff auf der Basis eines Lexems – in diesem Fall NATION – gebildet werden kann. Hierbei ist zu beachten, dass sich durch die schrittweise Modifikation sowohl der Worttyp als auch die Bedeutung des Begriffes ändern kann: Durch die Endung „-al“ wird aus dem Substantiv „nation“ das Adjektiv „national“, aus dem durch die Endung „-ize“ wiederum das Verb „nationalize“ wird. Ersetzt man hier den letzten Buchstaben durch die Endung „-ation“ erhält man wiederum ein Substantiv, welches zwar auf dem gleichen Lexem beruht wie „nation“, jedoch eine komplett andere Bedeutung hat. So bezeichnet „nationalization“ einen Prozess, wohingegen „nation“ eine Ansammlung von Menschen bezeichnet. Noch deutlicher wird der Zusammenhang zwischen Morphologie und Bedeutung, wenn wir uns den Begriff „Gleichheit“ ansehen. Durch die Vorsilbe „Un-“ können wir hier die Bedeutung umkehren, indem wir den Begriff „Ungleichheit“ erzeugen.

In der Linguistik wird das Problem der Mehrdeutigkeit traditionell durch die Verwendung spezieller Relationen zwischen Wörtern gelöst, ohne explizit auf deren Bedeutung einzugehen. Die Relationen geben lediglich an, wie sich die Bedeutungen der entsprechenden Wörter zueinander verhalten. So spricht man von Synonymen, wenn zwei unterschiedliche Wörter dieselbe Bedeutung haben („Bank“ und „Geldinstitut“), Antonymie bezeichnet den Fall, dass Wörter entgegengesetzte Bedeutung besitzen („Gleichheit“ und „Ungleichheit“). Steht ein Wort für eine Kategorie, welche die eines anderen Wortes umfasst, so sprechen wir von einem „Hypernym“ aus der Sicht des speziellen bzw. einem „Hyponym“ aus der Sicht des allgemeineren Wortes. Besitzt ein Wort mehrere unterschiedliche Bedeutungen, so bezeichnen wir dieses als „Homonym“. Die Definition dieser Relationen zeigt hierbei interessante Parallelen zu der oben beschriebenen Bildung von Kategorien in der Ontologie. In der Ontologie wurden Objekte und deren Eigenschaften betrachtet.



Konnten Objekte anhand einer bestimmten Anzahl charakteristischer Eigenschaften nicht unterschieden werden, so gehörten sie der gleichen Kategorie an. In der linguistischen Semantik werden anstelle von Objekten Worte und anstelle charakteristischer Eigenschaften natürlichsprachliche Aussagen betrachtet. So werden zwei Worte genau dann als Synonyme aufgefasst, wenn sie im Kontext beliebiger Aussagen nicht unterscheidbar und daher beliebig austauschbar sind, ohne den Wahrheitsgehalt zu verändern. Mit Hilfe des Prinzips der linguistischen Inferenz lassen sich auch die anderen genannten Relationen definieren. Ersetzt man einen Begriff durch ein Hyponym, so impliziert die ursprüngliche Aussage die neu entstandene, nicht aber anders herum. Im Falle von Hypernymen ist es gerade umgekehrt: Die neu entstandene Aussage impliziert stets die ursprüngliche. Bei Antonymen impliziert die ursprüngliche Aussage die Negation der abgeleiteten und anders herum. Homonyme erkennt man daran, dass Aussagen, die das Homonym enthalten, in bestimmten Situationen unterschiedliche Wahrheitswerte annehmen können. Betrachten wir etwa die Aussage „Er ging zur Bank hinüber“ so kann diese Aussage für die Interpretation von „Bank“ als Sitzmöbel wahr sein, während die Aussage für die Interpretation von „Bank“ als „Bankgebäude“ falsch sein kann.

Diese Definition der Bedeutung von Begriffen auf der Basis linguistischer Inferenz entspricht im Wesentlichen dem Prinzip der Kategorisierung von Objekten, wie sie im letzten Abschnitt beschrieben wurde, vorausgesetzt man betrachtet eine genügend große und repräsentative Menge charakteristischer Eigenschaften. Deckt sich diese Menge mit den unterschiedlichen Erwartungen, die wir an das Verhalten eines Objektes in einem bestimmten Kontext haben, so führt dies dazu, dass wir bestimmte Objekte auch im Kontext einer Aussage nicht unterscheiden können und somit Aussagen über entsprechende Objekte stets den gleichen Wahrheitswert zugewiesen bekommen.

## 1.4 Anmerkungen zu Begrifflichkeiten

Wir haben es bisher vermieden, eine explizite Definition des Begriffs „Ontologie“ anzugeben. Es existieren zwar eine Reihe solcher Definitionen, diese umfassen jedoch entweder nur eine bestimmte Sichtweise auf Ontologien oder sind so allgemein gehalten, dass sie kaum geeignet sind, Ontologien besser zu verstehen. Als Beispiel sei die wohl meistzitierte Definition von Gruber angeführt. Dieser definiert eine Ontologie als eine „explizite Spezifikation einer Konzeptualisierung“. Um tatsächlich zu verstehen, was eine Ontologie ausmacht, müsste man hier noch die Begriffe Spezifikation und Konzeptualisierung genauer definieren und festlegen, wann eine Spezifikation als explizit gelten kann. Versuche, diese Definition genauer zu fassen, führen auf der anderen Seite dazu, dass bestimmte Arten von Ontologien ausgeschlossen werden. So verfeinern Studer und andere die oben angegebene Definition, indem sie eine Ontologie als eine „formale, explizite Spezifikation einer gemeinsam genutzten Konzeptualisierung“ bezeichnen. Diese Definition schließt jedoch die meisten aus der Linguistik stammenden Ansätze aus, da diese nicht im mathematischen Sinne formal sind. Auch die gemeinsame Nutzung, die häufig als Kriterium für Ontologien angegeben wird, schließt bestimmte Modelle aus. So wäre ein Modell, das wir als Ontologie bezeichnen würden, in dem Moment, in dem es nur noch von einer Person genutzt wird, keine Ontologie mehr. Um solche Probleme zu vermeiden, legen wir uns in diesem Buch nicht auf eine bestimmte Definition des Begriffs Ontologie fest, sondern versuchen einen Überblick über Ideen aus unterschiedlichen Disziplinen zu geben, die einen Beitrag zur Entwicklung von Konzepten, Technologien und Anwendungen von Modellen geleistet haben, die wir im Allgemeinen als Ontologien bezeichnen würden.

Dieser Ansatz bringt allerdings ein Problem mit sich. Da wir Ideen aus unterschiedlichen Bereichen beschreiben, haben wir es häufig mit unterschiedlichen Bezeichnungen für die gleichen oder zumindest sehr ähnliche Konzepte zu tun. Wir haben uns dagegen entschieden, eine Vereinheitlichung dieser unterschiedlichen Terminologien vorzunehmen, da dies zwar die Lektüre dieses Buches vereinfachen würde, es jedoch erschweren würde, Verbindungen zur entsprechenden Fachliteratur herzustellen, die in [Kapitel 8](#) angegeben ist. Da dieses Buch lediglich einen Einstieg in das Thema bieten kann, scheint dieser Nachteil schwerer zu wiegen als der Vorteil einer einheitlicheren Darstellung. Um allzu große Verwirrung zu vermeiden, werden im Folgenden bestimmte wichtige Begriffe und ihre verschiedenen Ausprägungen, die sich in unterschiedlichen Teilen dieses Buches finden, diskutiert.

Da die Beschreibung von Objekten der realen Welt das erklärte Ziel von Ontologien ist, spielen diese eine wichtige Rolle. Im Zusammenhang mit Kategoriensystemen haben wir bereits erwähnt, dass solche Objekte in ihrer allgemeinsten Form, besonders im Kontext der Philosophie auch als „Entitäten“ bezeichnet werden. In der Informatik und der formalen Logik, die vor allem in den [Kapitel 2](#) und [4](#) thematisiert werden, werden diese Objekte häufig auch als „Instanzen“ bezeichnet, und zwar dann, wenn Objekte einer bestimmten Kategorie zugeordnet sind. Wird eine Kategorie als Menge von Objekten aufgefasst, ist manchmal auch von „Elementen“ die Rede.

Kategorien selbst besitzen als zentrales Thema der Ontologie ebenfalls eine Reihe unterschiedlicher Bezeichnungen. Während in der Künstlichen Intelligenz oft von „Konzepten“ die Rede ist, hat sich in der Informatik der Begriff des „Klasse“ durchgesetzt. In der Linguistik ist oft von „Begriffen“ die Rede, wenn Kategorien beschrieben werden sollen. Hier gibt es zusätzlich die

Unterscheidung zwischen „Begriffen“ und „Worten“, welche den Begriff beschreiben können, wobei „Begriff“ eher den gleichen Status besitzt wie Klassen oder Konzepte. Worte als Repräsentanten von Konzepten werden in den übrigen Bereichen selten thematisiert.

Ähnlich verhält es sich mit der Beschreibung charakteristischer Eigenschaften von Objekten, die zu einer bestimmten Kategorie gehören. Diese werden ja, wie oben beschrieben, in der Philosophie als „differentiae“ bezeichnet. In den anderen relevanten Disziplinen wird häufig von Eigenschaften und Attributen gesprochen. Diese wiederum werden oft durch „Relationen“ dargestellt. Zusätzlich werden in bestimmten Gebieten spezielle Bezeichnungen verwendet. So wird vor Allem im Kontext von Beschreibungslogiken und semantischen Netzen oft von „Slots“ gesprochen. Im Umfeld des Semantic Web hat sich die englische Bezeichnung „Properties“ für charakteristische Eigenschaften durchgesetzt. Für die Vererbungsrelation als wichtigste Beziehung zwischen Konzepten finden sich ebenfalls eine Reihe unterschiedlicher Bezeichnungen. Einige wurden bereits im vorangegangenen Kapitel erwähnt. So spricht die Linguistik von Hyponymie bzw. Hypernymie, um die Beziehung zwischen Ober- und Unterbegriffen zu beschreiben. In semantischen Netzen, die in [Kapitel 2](#) beschrieben werden, wird die Bezeichnung „a kind-of“ Relation verwendet, die auch „ako“ abgekürzt wird. Eine spezielle Form der Vererbungsbeziehung, die in der Regel eine Teilmengenbeziehung zwischen den Instanzen zweier Konzepte ist, ist die sogenannte Subsumptionsbeziehung. Der Begriff der Subsumption wird hierbei auch als terminus technicus im Kontext der Beschreibungslogik bezeichnet und stellt dort die logische Implikation zwischen zwei Konzeptdefinitionen dar, welche als Vererbungsrelation interpretiert wird.

## Kapitel 2

# Repräsentation von Bedeutung

Die im vorangegangenen Kapitel vorgestellten Methoden aus der Philosophie und der Linguistik zur Erfassung von Semantik zielten primär darauf ab, Prinzipien zu verstehen. Ihr Ansatz war demnach eher analytisch. In der Informationsverarbeitung ist das Ziel ein anderes. Hier geht es nicht in erster Linie um das Verstehen von Prinzipien, sondern vielmehr um deren Anwendung, um ein gegebenes Problem zu lösen. Dieser eher konstruktive Ansatz bedingt eine formālere Erfassung von Bedeutung als bisher. Insbesondere ist es notwendig, die im letzten Abschnitt diskutierten Prinzipien derart formal abzubilden, dass eine zumindest teilweise Automatisierung der semantischen Analyse möglich wird. Die Entwicklung solcher Formalismen ist Inhalt des Teilgebietes der Künstlichen Intelligenz, welches als Wissensrepräsentation bezeichnet wird. Hier werden Formalismen aus dem Bereich der diskreten Mathematik verwendet, um menschliches Wissen über den Zustand der Welt zu formalisieren und daraus Schlussfolgerungen zu ziehen, die menschliche Problemlösungskompetenz nachbilden können.

Insbesondere werden grundlegende Ansätze zur Formalisierung von Bedeutung vorgestellt, die jeweils in enger Beziehung zu den im letzten Kapitel beschriebenen Prinzipien stehen. Dies sind sogenannte semantische Netze. Sie verwenden Graphentheoretische Konzepte zur Beschreibung von Beziehungen zwischen Begriffen und lehnen sich hierbei stark an die oben beschriebene Darstellung von Wortbedeutungen in der Linguistik an. Als wohl populärster Ansatz zur Darstellung von Wissen und Bedeutung wird zudem die formale Logik und das logische Schließen dargestellt. Es wird gezeigt, dass sich die im letzten Kapitel vorgestellten Mechanismen, insbesondere die Einordnung von Objekten sowie die Überprüfung semantischer Relationen auf der Grundlage linguistischer Inferenz, durch logisches Schließen abbilden lassen.

Die dargestellten Ansätze bilden die Grundlage moderner Ontologiesprachen, wie der Web Ontology Language OWL, in der wir, wie in [Abschnitt 4.2.2](#) beschrieben, die Anordnung von Konzepten in einem Verband, die Verbindung von Begriffen durch Relationen sowie die Verwendung logischen Schließens als Hauptinferenzmechanismus wiederfinden. Zunächst wollen wir jedoch diese Prinzipien getrennt betrachten.

## 2.1 Semantische Netze

Eine relativ intuitive Art der Darstellung von Bedeutung sind die sogenannten semantischen Netze. Diese sind Graphen, die Begriffe und ihre Relationen zueinander wiedergeben. Motiviert werden diese durch die oben beschriebene Darstellung von Semantik in der Linguistik, die sich spezieller Relationen bedient, um das Verhältnis von Begriffen zueinander zu definieren und hierdurch implizit deren Bedeutung zu erfassen. Im Unterschied zur oben beschriebenen lexikalischen Semantik werden in

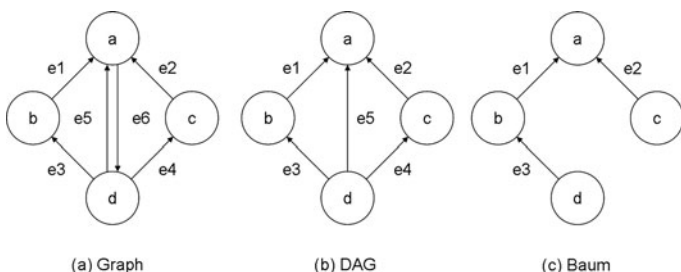
semantischen Netzen häufig jedoch nicht nur spezielle semantische Relationen wie Synonymie oder Hypernymie, sondern beliebige Relationen zwischen Begriffen angewandt. Bevor wir uns der Verwendung von semantischen Netzen zuwenden, wollen wir uns jedoch zunächst einige Grundbegriffe der Graphentheorie anschauen, die helfen, semantische Netze zu formalisieren.

### Exkurs: Graphentheorie

Ein (gerichteter Multi-)Graph ist ein Tupel  $G = (N, E, l, L, s, t)$ . Die Menge  $N$  bezeichnet hierbei eine Menge der Knoten,  $E$  die Menge der Kanten. Das Element  $l : N \cup E \rightarrow L$  bezeichnet eine Funktion, die jedem Knoten und jeder Kante einen Namen aus der Menge  $L$  zuweist. Ferner sind  $s : E \rightarrow N$  und  $t : E \rightarrow N$  Funktionen, die den Ursprungs- bzw. den Zielknoten einer Kante referenzieren. Hierbei bezeichnen wir für jede Kante  $e$  den Knoten  $t(e)$  als Nachfolger des Knotens  $s(e)$ . Ein (gerichteter) Pfad von einem Knoten  $n_1$  zu einem Knoten  $n_2$  in einem Graphen  $G$  ist eine Sequenz von Kanten  $(e_1, \dots, e_n)$ , so dass  $s(e_1) = n_1$  und  $t(e_n) = n_2$  sowie  $t(e_i) = s(e_{i+1})$  für  $i = 1, \dots, n-1$ . Es gibt eine Reihe von Spezialfällen von Graphen, denen aus theoretischen, aber auch aus praktischen Gründen besondere Bedeutung zukommt. Insbesondere sind dies Bäume und kreisfreie Graphen. Ein Graph im Sinne der oben genannten Definition ist ein Baum, falls alle Knoten  $n \in N$  genau einen Nachfolger besitzen. Ein Graph heißt kreisfrei, wenn es keinen Knoten gibt, der einen Pfad zu sich selbst hat. Kreisfreie gerichtete Graphen werden auch als DAG (Englisch: directed acyclic graph) genannt.

Abbildung 2.1 zeigt mehrere Beispielgraphen. In der oben eingeführten formalen Notation besteht der Graph in Abb. 2.1(a) aus der Knotenmenge  $N_{(a)} = \{a, b, c, d\}$  sowie der Kantenmen-

ge  $E_{(a)} = \{e_1, \dots, e_6\}$  mit  $t(e_1) = t(e_2) = t(e_5) = a$ ,  $t(e_3) = b$ ,  $t(e_4) = c$  und  $t(e_6) = d$  sowie  $s(e_1) = b$ ,  $s(e_2) = c$ ,  $s(e_3) = s(e_4) = s(e_5) = d$ . Die Funktion  $l$  sei an dieser Stelle erst einmal außer Acht gelassen. Anhand der oben eingeführten Definitionen können wir außerdem feststellen, dass der Graph weder kreisfrei (es gibt einen Pfad von  $a$  nach  $a$ , nämlich  $(e_6, e_3, e_1)$ ) noch ein Baum ist (der Knoten  $d$  hat mehr als einen Nachfolger, nämlich  $a$ ,  $b$  und  $c$ ). Entfernen wir allerdings die Kante  $e_6$  aus dem Graphen, so erhalten wir den in Abb. 2.1(b) dargestellten kreisfreien Graphen. Durch das Entfernen der Kante fallen alle zuvor vorhandenen Kreise weg. Entfernen wir zusätzlich die Kanten  $e_5$  und  $e_4$ , wird aus dem kreisfreien Graphen ein Baum, in dem jeder Knoten nur einen einzigen Nachfolger hat.



**Abb. 2.1** Beispielgraphen

Aus Sicht der Informatik hat die Verwendung von Graphen als Mittel zur Repräsentation von Bedeutung den Vorteil, dass sich Graphen auf der Grundlage effizienter Datenstrukturen sehr gut im Rechner darstellen lassen und eine Vielzahl von Algorithmen zur Anwendung von Graphen bekannt sind, die bei der Verarbeitung von konzeptuellem Wissen in Form von Graphen verwendet werden können.

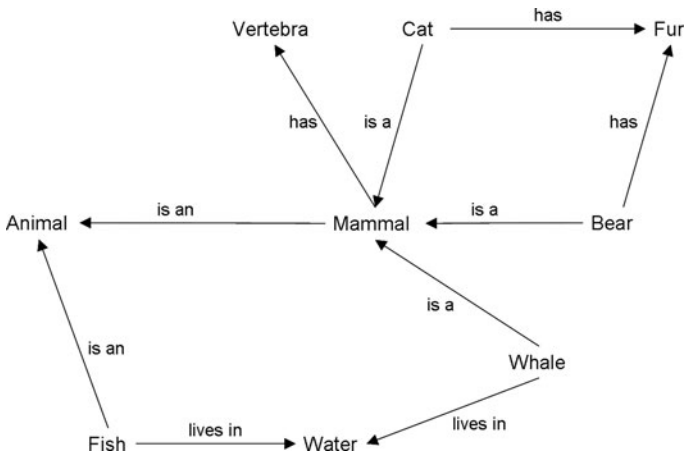


## Semantische Netze

Die Darstellung von ontologischem Wissen auf der Grundlage von Graphen geht auf Ross Quillian zurück. Dieser beschäftigte sich in den sechziger Jahren mit Modellen des menschlichen Gedächtnisses bei der Verarbeitung von Sprache und deren Bedeutung. Die Hypothese war hierbei, dass Wissen über Worte und ihre Bedeutung als Netzwerk (gerichtete Graphen mit Kantenbewertungen) von Begriffen dargestellt wird. Begriffe stellen hierbei die Knoten des Netzwerkes dar. Die Bewertung von Kanten zwischen den entsprechenden Knoten und die Entfernung von Knoten im Netzwerk repräsentieren hierbei, wie stark die Begriffe miteinander assoziiert werden, da sie in einer semantischen Beziehung zueinander stehen. Der Begriff „Bank“ wird beispielsweise stärker mit dem Begriff „Geld“ assoziiert, als mit dem Begriff „Schuh“. Viele Gedächtnismodelle stellen hierbei nicht dar, wodurch die Assoziation zwischen Begriffen begründet ist. Semantische Netze zeichnen sich nun dadurch aus, dass sie zwischen unterschiedlichen Typen von Kanten im Netzwerk unterscheiden. Durch diese Kantentypen wird die Art der Beziehung zwischen Begriffen charakterisiert und hierdurch die Assoziation erklärt. Abbildung 2.2 zeigt das Beispiel eines einfachen semantischen Netzes, in dem die Beziehungen zwischen Begriffen aus dem Tierreich dargestellt werden.

Wenn wir das Netz in der Abbildung mit dem in [Kapitel 1](#) beschriebenen Aufbau von Kategoriensystemen vergleichen, so können wir gewisse Parallelen feststellen. Die Knoten des Netzes stellen insbesondere Kategorien von Objekten dar. Zum Beispiel repräsentiert der mit ANIMAL bezeichnete Knoten die Kategorie aller Tiere, der mit MAMMAL bezeichnete die aller Säugetiere.

Zusätzlich spezifiziert das Netz charakteristische Eigenschaften der entsprechenden Kategorien. In einem semantischen Netz



**Abb. 2.2** Graph-Darstellung eines einfachen semantischen Netzes

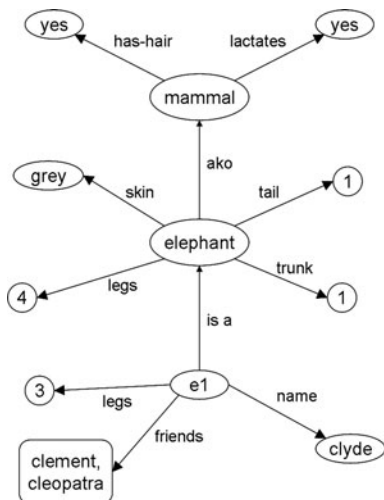
werden diese Eigenschaften, die allen Objekten einer Kategorie gemeinsam sind, durch Relationen zu anderen Elementen des Netzes dargestellt. So wird der Besitz von Wirbeln, welche die charakteristische Eigenschaft aller Säugetiere ist, durch die mit „has“ bezeichnete Verbindung zu der Kategorie aller Wirbel repräsentiert. Sie wird durch den mit VERTEBRA bezeichneten Knoten dargestellt. Auf ähnliche Weise werden charakteristische Eigenschaften der anderen beschriebenen Kategorien (Katzen und Bären besitzen ein Fell, Fische und Wale leben im Wasser) wiedergegeben. Ein wesentlicher Unterschied zu den im Abschnitt über Kategoriensysteme beschriebenen „differentiae“ ist, dass die in einem semantischen Netz gezeigten Eigenschaften einer Kategorie in den seltensten Fällen eine vollständige Beschreibung der hinreichenden Bedingungen darstellt, die es erlauben, ein Objekt eindeutig der entsprechenden Kategorie zuzuordnen. Die beschriebenen Eigenschaf-

ten sind vielmehr als notwendige Bedingungen in dem Sinn zu verstehen, dass alle Objekte, die der entsprechenden Kategorie angehören, die entsprechende Eigenschaft aufweisen müssen. Diese Unterscheidung zwischen hinreichenden, eine Kategorie definierenden und notwendigen Eigenschaften wird auch im Zusammenhang mit modernen Ontologiesprachen ein wichtiges Thema sein.

Ein weiteres Element, welches wir im Zusammenhang mit Kategoriensystemen kennengelernt haben, nämlich das Prinzip von Unter- und Oberkategorien, findet sich in semantischen Netzen wieder. Dieses wird durch mit IS-A bezeichnete Kanten zwischen Knoten, die Kategorien darstellen, bezeichnet. So können wir dem semantischen Netz entnehmen, dass die Kategorie der Säugetiere eine Unterkategorie der Kategorie aller Tiere darstellt und somit alle Säugetiere automatisch auch Tiere sind.

Neben der Darstellung von Kategorien und charakteristischen Eigenschaften finden sich in semantischen Netzen häufig auch Knoten, die konkrete Objekte oder auch Werte bestimmter Eigenschaften repräsentieren, sowie Relationen zwischen diesen unterschiedlichen Arten von Knoten. Beispiele hierfür enthält das in Abb. 2.3 dargestellte Netz.

Das Netz beschreibt den dreibeinigen Elefanten Clyde, der durch den mit e1 bezeichneten Knoten dargestellt wird. Durch Verbindungen zu anderen Knoten im Netz, die ebenfalls konkrete Objekte darstellen, werden seine tatsächlichen Eigenschaften, wie der Name „Clyde“ und die Freundschaft zu Clement und Cleopatra, beschrieben. Ein gewisses Problem stellt die Darstellung der Tatsache dar, dass Clyde nur drei Beine hat. Eine naheliegende Wiedergabe im Sinne des bisher Betrachteten wäre, für jedes von Clydes Beinen einen Objektknoten in das Netz einzufügen und diesen mit Clyde zu verbinden. Diese Darstellung lässt jedoch lediglich den Schluss zu, dass Clyde mindestens drei Beine hat, nämlich die dargestellten. Es ist nicht



**Abb. 2.3** Ein semantisches Netz mit konkreten Objekten

klar, ob es nicht auch noch ein viertes Bein gibt, welches nur nicht explizit erwähnt wird. Die Frage, ob es ein solches weiteres Bein geben kann, ist ebenfalls zentral bei der Beschreibung von konzeptuellem Wissen in Form von Ontologien. Es gibt hier zwei unterschiedliche Sichtweisen: Verwendet man die sogenannte „Closed-World“ Annahme, kann davon ausgegangen werden, dass alle relevanten Informationen im Modell dargestellt sind. In diesem Fall würde man davon ausgehen, dass Clyde genau drei Beine hat. Die andere häufig verwendete Annahme ist die „Open-World“ Annahme. Bei dieser geht man davon aus, dass die Tatsache, dass eine bestimmte Aussage nicht im Modell enthalten ist, nicht automatisch bedeutet, dass diese nicht gilt. Es bleibt vielmehr offen, ob die entsprechende Aussage gilt oder nicht. In unserem Fall hieße das, dass wir lediglich mit

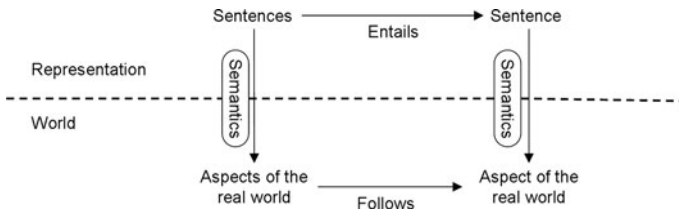
Bestimmtheit sagen können, dass Clyde mindestens drei Beine hat, es könnten aber auch vier oder mehr sein. In unserem speziellen Fall kann dieses Problem umgangen werden, indem die Anzahl der Beine explizit durch die Zahl „3“, die in diesem Fall ebenfalls als ein Objekt angesehen wird, wiedergegeben wird.

Neben der beschriebenen Darstellung konkreter Objekte enthält das semantische Netz außerdem auch die Beschreibung von Kategorien, wie die Kategorie aller Elefanten und die Kategorie aller Säugetiere. Die Verbindung zwischen Clyde und der Kategorie aller Elefanten wird hierbei durch eine IS-A Beziehung repräsentiert. Aus ontologischer Sicht ist dies ein Problem, da die Beziehung zwischen zwei Kategorien eine grundlegend andere Bedeutung hat als die zwischen einer Kategorie und ihren Objekten. Im ersten Fall entspricht die IS-A Beziehung einer Teilmengenrelation zwischen den jeweiligen Objektmengen, im zweiten Fall bezeichnet diese die Mitgliedschaft eines Objektes in der Menge aller Objekte, die dieser Kategorie angehören. Um diesem Unterschied Rechnung zu tragen, wird die Beziehung zwischen Unter- und Oberkategorie hier mit AKO (a kind of) bezeichnet und somit klar von der IS-A (is a) Relation unterschieden. Das Beispiel von Clyde zeigt allerdings auch, dass die Interpretation der IS-A und AKO Beziehungen mitunter problematisch sein können. Wenn wir uns die charakteristischen Eigenschaften der Elefanten-Kategorie ansehen, stellen wir fest, dass hier unter anderem festgelegt ist, dass Elefanten im Allgemeinen vier Beine haben. Nach der oben beschriebenen Interpretation dieser Beschreibung als notwendige Bedingung für alle Elefanten bedeutet dies, dass jedes Objekt der Kategorie Elefant vier Beine haben muss. Gleichzeitig ist festgelegt, dass Clyde, der ja nur drei Beine hat, dieser Kategorie angehört. Dies bedeutet nun, dass das Modell einen Widerspruch enthält, den wir nicht ohne Weiteres auflösen können. Ähnliche Konflikte können auch zwischen den Beschreibungen unterschiedlicher Kategorien

auftreten, da ja Unterkategorien alle Eigenschaften ihrer Oberkategorien erben, so dass es zu Widersprüchen zwischen diesen vererbten und den direkt für die Unterkategorie definierten Eigenschaften kommen kann. Eine Lösung für das Problem fehlender Eindeutigkeit bietet die im folgenden Kapitel beschriebene Verwendung formaler Logik als Grundlage für die Beschreibung von terminologischem Wissen. Logik wird hierbei als Alternative bzw. Ergänzung zu der hier beschriebenen Darstellung in Form eines semantischen Netzes verwendet.

## 2.2 Logik

Die formale Logik hat wie die Ontologie ihren Ursprung in der klassischen Philosophie. Während sich die Ontologie, wie in [Kapitel 1](#) beschrieben, mit der Natur von Objekten der realen Welt beschäftigt, wurde die formale Logik als Mittel zur Beschreibung des menschlichen Denkens und Schlussfolgerns untersucht. Zu diesem Zweck verfügt die formale Logik über Sprachkonstrukte zur Beschreibung eines konkreten Zustands der Welt sowie Schlussfolgerungsregeln zum Ableiten von Wissen über die Welt, welches dem Beobachter nicht direkt zugänglich ist. [Abbildung 2.4](#) veranschaulicht diesen Zusammenhang: Logische Theorien bestehen aus Mengen von Formeln (Syntax), welche bestimmte Aspekte der realen Welt beschreiben. Welche diese sind, bestimmt die Abbildung der Formeln in ein mathematisches Modell der realen Welt (Semantik). Das Prinzip der Logik ist nun, aus bestehenden Formeln, welche einen Zustand der Welt beschreiben, syntaktisch neue Formeln abzuleiten. Entsprechende Methoden sind so gestaltet, dass – die korrekte Beschreibung der Welt vorausgesetzt – abgeleitete Formeln ebenfalls wahre Aussagen über die Welt darstellen, da diese Aussagen intuitiv folgen.



**Abb. 2.4** Prinzip der logischen Formalisierung

Gemeinsames Interesse von Ontologie und Logik ist die Beschreibung der Welt. Während sich die Ontologie hierbei auf die inhaltlichen Aspekte konzentriert, liegt der Beitrag der formalen Logik auf der Bereitstellung einer Sprache zur Formalisierung dieser Beschreibung. Auf den ersten Blick bilden Ontologie und Logik daher eine perfekte Kombination zur Darstellung von Bedeutung, da formale Logik die notwendigen Werkzeuge liefert, um etwa die notwendigen und hinreichenden Bedingungen eines Kategoriensystems zu formalisieren. Wie wir später sehen werden, gibt es jedoch auch eine Reihe von Problemen bezüglich der Verwendung von Logik zur formalen Beschreibung von Ontologien. Dennoch stellt die formale Logik heute eine der wichtigsten Methoden zur Repräsentation von Ontologien dar. Ihre Beliebtheit ist hierbei vor allem darauf zurückzuführen, dass es aufgrund der erwähnten Schlussfolgerungsregeln möglich ist, aus Ontologien implizites Wissen über die Welt abzuleiten sowie automatisch zu überprüfen, ob ein bestimmter Zustand der Welt dem in der Ontologie beschriebenen Wissen entspricht. Diese Möglichkeiten werden wir in [Teil II](#) genauer betrachten. Bevor wir im Folgenden einige grundlegende Prinzipien der Verwendung von Logik zur Repräsentation von Ontologien und semantischen Netzen diskutieren, soll an dieser Stelle Syntax und Semantik der Prädikatenlogik erster Stufe als gebräuchlichste

Form der Logik in Erinnerung gerufen werden. Eine umfassende Behandlung würde den Rahmen dieses Buches sprengen, doch als eine sehr gute und detaillierte Einführung sei dem Leser das Lehrbuch „Logik für Informatiker“ von Uwe Schöning empfohlen [Schöning, 1995].

### Exkurs: Prädikatenlogik erster Stufe

Grundlegender Bestandteil prädikatenlogischer Formeln sind Terme. Terme sind hierbei abstrakte Beschreibungen von Objekten der realen Welt mit Hilfe von Variablen und Funktionen. Sie werden hierbei wie folgt rekursiv definiert:

- Jede Variable  $x_i$ ,  $i = 1, \dots, n$  ist ein Term.
- Sei  $f_i$  ein  $k$ -stelliges Funktionssymbol und  $t_1, \dots, t_k$  Terme, dann ist auch  $f_i(t_1, \dots, t_k)$  ein Term. Funktionssymbole der Stelligkeit null werden als Konstanten bezeichnet und als  $c_i$  dargestellt.

Beispiele von Termen sind *clyde* (Konstante), *name(e1)* (einstellige Funktion angewendet auf eine Konstante), *color(skin(e1))* (einstellige Funktion angewendet auf das Ergebnis der Anwendung einer einstelligen Funktion auf eine Konstante) usw.

Atomare Formeln entstehen durch die Anwendung von Prädikaten auf Terme und stehen für einfache Aussagen über den Zustand der Welt, die wahr oder falsch sein können. Häufig ist es hierbei sinnvoll, die Gleichheit von Termen als spezielles Prädikat zu erlauben:

- Sei  $P_i$  ein Prädikatensymbol der Stelligkeit  $k$  und  $t_1, \dots, t_k$  Terme, dann ist  $P_i(t_1, \dots, t_k)$  eine atomare Formel.
- Seien  $t_1$  und  $t_2$  Terme, dann ist  $t_1 = t_2$  eine atomare Formel.



Beispiele atomarer Formeln sind  $Elephant(e1)$  (Anwendung eines einstelligen Prädikats auf eine Konstante),  $is - grey(color(skin(e1)))$  (Anwendung eines einstelligen Prädikats auf einen komplexen Term),  $friends(e1, clement)$  (Anwendung eines zweistelligen Prädikats auf zwei Konstanten) oder  $name(e1) = clyde$  (Gleichheit).

Auf der Grundlage atomarer Formeln lassen sich nun auf folgende Weise allgemeine Formeln bilden, die komplexe Aussagen über den Zustand der Welt darstellen:

- Jede atomare Formel  $F_i$  ist eine Formel.
- Sei  $F$  eine Formel, dann ist auch  $\neg F$  eine Formel (Negation).
- Seien  $F_1$  und  $F_2$  Formeln, dann sind auch  $F_1 \wedge F_2$  (Konjunktion),  $F_1 \vee F_2$  (Disjunktion) und  $F_1 \rightarrow F_2$  (Implikation) Formeln.
- Sei  $x_i$  eine Variable und  $F$  eine Formel, dann sind auch  $\forall x_i F$  (universelle Quantifizierung) und  $\exists x_i F$  (existentielle Quantifizierung) Formeln.

Auf der Basis dieser syntaktischen Regeln lassen sich beinahe beliebig komplizierte Aussagen über die Welt formulieren. Beispiele sind  $\exists x friends(clyde, x) \wedge name(x) = clement$  oder  $\forall x \neg Elephant(x) \vee color(skin(x)) = grey$ .

Die Semantik einer prädikatenlogischen Formel ergibt sich durch die Abbildung der in der Formel vorkommenden Terme und Prädikate in eine mathematische Struktur, das sogenannte Universum, das mögliche Zustände der realen Welt beschreibt. Das Universum  $U$  ist hierbei eine nicht leere Menge von Objekten der realen Welt, die potenziell in bestimmten Relationen zueinander stehen können. Jede prädikatenlogische Formel stellt nun eine unvollständige Beschreibung der möglichen Relationen zwischen den Objekten des Universums dar. Je nachdem, ob die beschriebenen Relationen tatsächlich bestehen, ist eine Formel

wahr oder falsch. Die Abbildung von Formeln in das Universum erfolgt durch eine Interpretationsabbildung  $I$ , die wie folgt definiert ist:

- $I$  weist jedem  $k$ -stelligen Prädikatsymbol  $P_i$  in der Formel eine  $k$ -stellige Relation über den Elementen von  $U$  zu. Diese Relation wird als  $P_i^I$  bezeichnet.
- $I$  weist jedem  $k$ -stelligen Funktionssymbol ( $k \geq 0$ )  $f_i$  in der Formel eine  $k$ -stellige Funktion über den Elementen von  $U$  zu. Diese Relation wird als  $f_i^I$  bezeichnet.
- $I$  weist jeder Variablen  $x_i$  und jeder Konstanten  $c_i$  ein Element von  $U$  zu. Dieses Objekt wird als  $x_i^I$  bzw.  $c_i^I$  bezeichnet.
- $I$  weist jedem komplexen Term  $t$  einen Wert  $I(t)$  zu, der wie folgt definiert ist:
  - Ist  $t$  eine Variable  $x_i$ , so gilt  $I(t) = x_i^I$
  - Ist  $t$  eine Konstante  $c_i$ , so gilt  $I(t) = c_i^I$
  - Ist  $t = f_i(t_1, \dots, t_k)$ , so gilt  $I(t) = f_i^I(t_1^I, \dots, t_k^I)$ .

Auf diese Weise ist sichergestellt, dass jeder Term in einer Formel auf ein Element von  $U$  verweist, welches unter Umständen das Ergebnis der Anwendung einer Funktion auf Elementen von  $U$  ist. Um den Wahrheitswert einer Formel zu ermitteln, müssen wir jetzt nur noch überprüfen, ob die in der Formel beschriebenen Relationen auf Termen tatsächlich zwischen den Werten dieser Terme in  $U$  bestehen. Formal wird der Wahrheitswert einer Formel über eine Abbildung  $\mathcal{A}$  definiert, die jeder Formel den Wert 0 (falsch) oder 1 (wahr) zuordnet. Wir beginnen bei der Definition zunächst mit atomaren Formeln.

- Sei  $F$  eine atomare Formel der Form  $P(t_1, \dots, t_k)$ , so gilt:  $\mathcal{A}(F) = 1$  falls  $(t_1^I, \dots, t_k^I) \in P^I$  und sonst 0. Das heißt, die Formel ist genau dann wahr, wenn die Werte der Terme  $t_1, \dots, t_k$  tatsächlich in der Relation  $P^I$  zueinander stehen.

- Sei  $F$  eine atomare Formel der Form  $t_1 = t_2$ , so gilt:  $\mathcal{A}(F) = 1$  falls  $t_1^I = t_2^I$  und sonst 0. Das heißt, die Formel ist genau dann wahr, wenn die Werte der Terme  $t_1$  und  $t_2$  tatsächlich das gleiche Element aus  $U$  darstellen.

Die Formel  $friends(e1, clement)$  ist also genau dann wahr, wenn die Elemente, auf die  $e1$  und  $clement$  abgebildet werden, sich tatsächlich in der Relation  $friends$  befinden und die entsprechenden Lebewesen tatsächlich befreundet sind. Die Formel  $name(e1) = clyde$  ist wahr, wenn der Wert der Funktion  $name$  angewendet auf  $e1$  den Wert  $clyde$  ergibt.

Die Wahrheitswerte für nicht atomare Formeln lassen sich nun auf folgende Weise aus den Wahrheitswerten der enthaltenen atomaren Formeln ableiten:

- Sei  $F$  eine Formel der Form  $\neg G$ , so gilt  $\mathcal{A}(G) = 1$  falls  $\mathcal{A}(F) = 0$ , sonst 0. Das heißt, die Negation einer Formel ist genau dann wahr, wenn die negierte Formel falsch ist und umgekehrt.
- Sei  $F$  eine Formel der Form  $F_1 \vee F_2$ , so gilt  $\mathcal{A}(F) = 1$  falls  $\mathcal{A}(F_1) = 1$  oder  $\mathcal{A}(F_2) = 1$ , sonst 0. Das heißt, die Konjunktion zweier Formeln ist genau dann wahr, wenn beide Formeln wahr sind.
- Sei  $F$  eine Formel der Form  $F_1 \wedge F_2$ , so gilt  $\mathcal{A}(F) = 1$  falls  $\mathcal{A}(F_1) = 1$  und  $\mathcal{A}(F_2) = 1$ , sonst 0. Das heißt, die Disjunktion zweier Formeln ist genau dann wahr, wenn eine der Formeln wahr sind.
- Sei  $F$  eine Formel der Form  $F_1 \rightarrow F_2$ , so gilt  $\mathcal{A}(F) = 1$  falls  $\mathcal{A}(F_2) = 1$  oder  $\mathcal{A}(F_1) = 0$ , sonst 0.

Etwas komplizierter ist die Bestimmung des Wahrheitswertes für quantifizierte Formeln. Um deren Wahrheitswert zu bestimmen, ist es notwendig, unterschiedliche mögliche Abbildungen von Variablen auf Elemente von  $U$  zu betrachten. Insbesondere gilt:

- Sei  $F$  eine Formel der Form  $\forall xG$ , so gilt  $\mathcal{A}(F) = 1$ , falls  $\mathcal{A}(G) = 1$  für alle möglichen Werte  $x^I$  aus  $U$ , sonst 0. Eine allquantifizierte Formel ist also genau dann wahr, wenn die quantifizierte Formel für alle möglichen Interpretationen der entsprechenden Variablen wahr ist.
- Sei  $F$  eine Formel der Form  $\exists xG$ , so gilt  $\mathcal{A}(F) = 1$ , falls es einen Wert  $x^I$  aus  $U$  gibt, so dass  $\mathcal{A}(G) = 1$ , sonst 0. Eine allquantifizierte Formel ist also genau dann wahr, wenn es eine mögliche Interpretation der Variablen  $x$  gibt, welche die quantifizierte Formel wahr macht.

Die Formel  $\exists x \text{friends}(\text{clyde}, x) \wedge \text{name}(x) = \text{clement}$  ist also dann wahr, wenn es ein Element aus  $U$  gibt, für welches, wenn  $x$  auf es abgebildet wird, beide der Teilformeln wahr sind. Zum einen muss sich dieses Element in der friends-Relation zu dem Element, welches clyde darstellt, befinden. Zum anderen muss die Anwendung der Namensfunktion auf dieses Element den Wert von clement ergeben. Die Formel  $\forall x \neg \text{Elephant}(x) \vee \text{color}(\text{skin}(x)) = \text{grey}$  auf der anderen Seite ist dann wahr, wenn für alle möglichen Werte der Variablen  $x$  mindestens eine der beiden Teilformeln wahr ist. Für alle Interpretationen von  $x$  muss der Wert des Terms  $\text{color}(\text{skin}(x))$  dem Wert „grey“ entsprechen, oder aber das Element darf nicht in der Relation „Elephant“ enthalten sein.

## Formalisierung semantischer Netze

Wie bereits kurz erwähnt, wurde Logik schon sehr früh als Grundlage für die Formalisierung vorgeschlagen, um Uneindeutigkeiten im Hinblick auf die korrekte Interpretation eines Netzes und auf die aus einem Netz ableitbaren Informationen zu beseitigen. Hierzu ist eine geeignete Übersetzung von Netzwerkstruk-

turen, wie den im letzten Abschnitt gezeigten, in Mengen logischer Formeln notwendig. Es ist offensichtlich, dass hierfür unterschiedliche Möglichkeiten bestehen. Im Folgenden werden einige dieser Möglichkeiten vorgestellt und deren Vor- und Nachteile diskutiert. Hierbei werden gezielt Ansätze betrachtet, die sich im Kern auch in aktuellen Ontologiesprachen wie RDF-Schema und OWL wiederfinden.

Eine sehr direkte Möglichkeit, semantische Netze in logische Formeln zu übersetzen, basiert auf der direkten Korrespondenz zwischen Graphen und Relationen, die sich wiederum als Prädikate ausdrücken lassen. Die Idee ist hierbei, jede Kante im Netzwerk, die ja jeweils die Instanz einer Relation darstellt, in eine atomare Formel zu übersetzen. Die Label der durch eine Kante verbundenen Knoten bilden hierbei die Terme der atomaren Formel, das Label der Kante wird zum Namen des Prädikats. Die erzeugten Formeln werden durch Konjunktion verbunden, da alle Relationen gelten müssen. Eine entsprechende Übersetzung des semantischen Netzes aus Abb. 2.2 würde somit folgendes Resultat ergeben:

$$\begin{aligned}
 & is - a(mammal, animal) \wedge is - a(fish, animal) \wedge \\
 & is - a(cat, mammal) \wedge \\
 & is - a(bear, mammal) \wedge is - a(whale, mammal) \wedge \\
 & has(mammal, vertebra) \wedge has(cat, fur) \wedge has(bear, fur) \wedge \\
 & lives - in(fish, water) \wedge lives - in(whale, water) \quad (2.1)
 \end{aligned}$$

Neben dieser recht einfachen Darstellung des im Netz enthaltenen Wissens liegt der eigentliche Vorteil der Verwendung von Logik in der Möglichkeit, ableitbares Wissen explizit zu spezifizieren. Zu diesem Zweck werden gewünschte Inferenzen in Form logischer Formeln spezifiziert. Die entsprechenden Formeln beschreiben hierbei allgemeingültige Zusammenhänge,

die sich auf die verwendeten Prädikate beziehen und unabhängig von konkretem Wissen gelten. Für das Beispiel aus Abb. 2.2 könnte man die folgenden allgemeinen Ableitungsregeln angeben:

$$\begin{aligned} \forall x, y, z \text{ is} - a(x, y) \wedge \text{is} - a(y, z) &\rightarrow \text{is} - a(x, z) \\ \forall x, y, z \text{ is} - a(x, y) \wedge \text{has}(y, z) &\rightarrow \text{has}(x, z) \\ \forall x, y, z \text{ is} - a(x, y) \wedge \text{lives} - \text{in}(y, z) &\rightarrow \text{lives} - \text{in}(x, z) \quad (2.2) \end{aligned}$$

Mit Hilfe dieser Formeln und der Formel, die aus der direkten Übersetzung des Netzes entstanden ist, kann man nun auf der Grundlage der formalen Semantik der Prädikatenlogik eindeutig bestimmen, welche weiteren Aussagen aus dem Modell folgen:

Die erste Regel besagt, dass die „is-a“ Relation transitiv ist. In unserem Beispiel können wir mit Hilfe dieser Regel etwa folgern, dass Katzen und Bären nicht nur Säugetiere, sondern auch Tiere sind. Insbesondere können wir, wenn wir die Variablen  $x$ ,  $y$  und  $z$  durch die Konstanten *cat* (bzw. *bear*), *mammal* und *animal* ersetzen, zeigen, dass die folgenden atomaren Formeln wahr sind:  $\text{is} - a(\text{bear}, \text{animal})$ ,  $\text{is} - a(\text{cat}, \text{animal})$ . Auf gleiche Weise kann auch abgeleitet werden, dass Wale Tiere sind ( $\text{is} - a(\text{whale}, \text{animal})$ ). Die Ableitungsregel ist wichtig, da die „is-a“ Beziehung eine wichtige Rolle bei der Ableitung neuer Informationen spielt, wie die beiden anderen Ableitungsregeln zeigen. Diese besagen, dass die durch die Relationen *has* und *lives - in* beschriebenen Eigenschaften an Unterklassen vererbt werden. Mit Hilfe dieser Regeln können wir im Beispiel ableiten, dass Wale, Katzen und Bären alle Wirbel besitzen: Durch die Ersetzung der Variablen  $x$ ,  $y$  und  $z$  durch „whale“ (bzw. „cat“ oder „bear“), „mammal“ und „vertebra“ in der zweiten Regel können wir zeigen, dass die Aussagen  $\text{has}(\text{whale}, \text{vertebra})$ ,  $\text{has}(\text{cat}, \text{vertebra})$  und  $\text{has}(\text{bear}, \text{vertebra})$  wahr sind.

Eine ebenso wichtige Frage, die sich auf der Grundlage der Übersetzung in Prädikatenlogik beantworten lässt, ist die Frage danach, was nicht gilt. In dem semantischen Netz wird zum Beispiel nichts darüber gesagt, ob Fische Wirbel besitzen. Man kann hierüber unterschiedliche Auffassungen haben. Eine mögliche Interpretation ist, dass Fische keine Wirbel haben, da dies ansonsten erwähnt wäre. Diese Interpretation wird auch als „Closed World Assumption“ bezeichnet. Demgegenüber steht die Auffassung, dass man aus der Nichterwähnung lediglich schließen kann, dass nichts darüber bekannt ist, ob Fische Wirbel besitzen oder nicht. Nach dieser Auffassung, die auch als „Open World Assumption“ bezeichnet wird, kann nur dann angenommen werden, dass Fische keine Wirbel besitzen, wenn diese Aussage widerlegt werden kann. In der oben beschriebenen Übersetzung in Prädikatenlogik ist es nicht möglich diese Aussage zu widerlegen, also zu zeigen, dass die Aussage *has(fish, vertebra)* falsch ist. Die Verwendung von Prädikatenlogik als Mechanismus zur Formalisierung semantischer Netze führt demnach automatisch dazu, dass die Open World Assumption gewählt wird.

Ein wesentlicher Aspekt der Formalisierung von Ableitungsregeln, der an dieser Stelle thematisiert werden muss, ist die Frage, ob Ableitungsregeln als Teil einer speziellen Ontologie anzusehen sind, oder ob sie einen allgemeingültigen Status einnehmen und für jede beliebige Ontologie gelten sollen. Es hat sich herausgestellt, dass diese Frage nicht eindeutig zu beantworten ist und man zwischen allgemeingültigen Regeln, die meist bereits in den Formalismus zur Darstellung von Ontologien integriert sind, und speziellen Regeln unterscheiden muss, die nur im Kontext eines bestimmten Modells sinnvoll sind. Dies zeigt sich schon an den oben genannten Beispielregeln. Wie im Abschnitt über semantische Netze besprochen, kann die Verwendung der „is-a“ Relation zur Beschreibung der Subklassenbeziehung ein allgemeingültiges Vorgehen sein. Zudem besteht breiter Konsens

darüber, dass diese Relation transitiv ist (vgl. [Kapitel 1](#)). Aus diesem Grund kann die erste Regel unabhängig davon verwendet werden, ob die Ontologie Wissen aus dem Tierreich oder über Bankkunden beschreibt. Im Fall der zweiten und dritten Regel ist dies weniger eindeutig. Diese Regeln beziehen sich neben der „is-a“ auf eine weitere Relation (im ersten Fall die *has* im zweiten die „lives-in“ Relation). Eine allgemeine Verwendung dieser Regeln hängt demnach davon ab, ob diese Relationen im Kontext einer bestimmten Ontologie sinnvoll sind. Im Fall der zweiten Regel könnte man argumentieren, dass die Relation so allgemein ist, dass sie in unterschiedlichsten Ontologien sinnvoll sein kann. Bei der dritten ist dies nicht gegeben, da die Relation *lives – in* zum Beispiel im Kontext einer Ontologie über Gendefekte wohl nicht sinnvoll wäre. Da eine Entscheidung, welche Relationen und somit welche Ableitungsregeln eine allgemeine Bedeutung haben, oft schwer zu treffen ist, wird normalerweise nur eine sehr begrenzte Menge von Ableitungsregeln verwendet und stattdessen werden Mechanismen zur Verfügung gestellt, die es erlauben, anwendungsspezifische Ableitungsregeln zu definieren.

## Logik-basierte Ontologien

Die oben beschriebene direkte Übersetzung semantischer Netze in eine Logik-basierte Darstellung stößt schnell an ihre Grenzen, wenn komplexere Zusammenhänge als die in [Abb. 2.2](#) dargestellt werden sollen. Entsprechende Probleme zeigen sich bereits, wenn wir versuchen, das semantische Netz aus [Abb. 2.3](#) auf die oben beschriebene Weise zu formalisieren. Eine solche Übersetzung würde zum Beispiel die folgenden Aussagen enthalten:

$$is - a(clyde, elephant) \wedge ako(elephant, mammal)$$



An diesem Beispiel kann man sehen, dass durch die Übersetzung konkrete Objekte (*clyde*) und Kategorien (*elephant*) in die gleiche Zielstruktur, in diesem Fall Konstanten, übersetzt werden, obwohl die Unterscheidung zwischen Objekten und Kategorien, wie in [Kapitel 1](#) beschrieben, eine fundamentale ist. Insbesondere soll ja die Beschreibung der Eigenschaften einer Kategorie, wie in diesem Fall „Elefant“, aussagen, dass diese Eigenschaften für alle Objekte dieser Kategorie gelten. Diese intendierte Aussage lässt sich durch die direkte Übersetzung der Kanten im Netzwerk in Prädikate nicht angemessen ausdrücken. Es hat sich gezeigt, dass es zur Beschreibung komplexer Kategorien von Vorteil ist, andere Formen der Abbildung auf Prädikatenlogik zu verwenden, welche die explizite Unterscheidung zwischen Objekten und Kategorien zulässt. Die Prinzipien dieser alternativen Darstellung sind folgende:

- Objekte und konkrete Werte werden durch logische Konstanten dargestellt.
- Kategorien und Relationen werden durch logische Prädikate dargestellt. Kategorien sind hierbei einstellige, Relationen zwischen Objekten zweistellige Prädikate.

Diese alternative Form der Abbildung terminologischer Informationen in Prädikatenlogik ermöglicht es jetzt auch, die zentralen Relationen „is a“ und „a kind of“, also die Zugehörigkeit von Objekten zu Klassen sowie die Unterklassen-Beziehung direkt in Konstrukte der Prädikatenlogik zu übersetzen:

- Die Zugehörigkeit eines Objektes zu einer Klasse wird durch die Anwendung des einstelligen Prädikates, welches die Klasse repräsentiert, auf die Konstante, welche das Objekt darstellt, ausgedrückt (*Elephant(clyde)*).
- Die Subklassen-Beziehung wird durch eine Implikation zwischen den entsprechenden einstelligen Prädikaten repräsentiert ( $\forall x \text{ Elephant}(x) \rightarrow \text{Mammal}(x)$ ).

Eine entsprechende Formalisierung des in Abb. 2.2 dargestellten Wissens über Ober- und Unterkategorien, welches schon im vorangegangenen Kapitel als Beispiel diente, hätte die folgende Form:

$$\begin{aligned}
 \forall x \text{ Mammal}(x) &\rightarrow \text{Animal}(x) \wedge \\
 \forall x \text{ Fish}(x) &\rightarrow \text{Animal}(x) \wedge \\
 \forall x \text{ Cat}(x) &\rightarrow \text{Mammal}(x) \wedge \\
 \forall x \text{ Bear}(x) &\rightarrow \text{Mammal}(x) \wedge \\
 \forall x \text{ Whale}(x) &\rightarrow \text{Mammal}(x)
 \end{aligned} \tag{2.3}$$

Die Verwendung der Implikation ersetzt hierbei vollständig die Ableitungsregel 2.2, die zuvor noch explizit angegeben werden musste.

Eine Frage, die sich nun stellt, ist die Darstellung der Informationen über die charakteristischen Eigenschaften der beschriebenen Klassen sowie der entsprechenden Ableitungsregeln. Betrachten wir hierfür zunächst die Aussage, dass alle Säugetiere Wirbel haben. Prinzipiell lassen sich solche charakteristischen Eigenschaften von Konzepten ebenfalls durch Implikationen darstellen:

$$\forall x C(x) \rightarrow \phi(x)$$

Eine solche Formel legt fest, dass die durch die Formel  $\phi$  repräsentierte Eigenschaft eine notwendige Bedingung für Objekte ist, welche zum Konzept  $C$  gehören: Alle Objekte vom Typ  $C$  haben Eigenschaft  $\phi$ . Hinreichende Eigenschaften können durch eine Umkehrung der Implikation beschrieben werden:

$$\forall x \phi(x) \rightarrow C(x)$$

Diese Formel bestimmt, dass alle Objekte, welche die Eigenschaft  $\phi$  besitzen, automatisch dem Typ  $C$  zugeordnet werden.

Auf diese Art ist es möglich, Klassenzugehörigkeiten bestimmter Objekte aufgrund ihrer Eigenschaften abzuleiten. Klassen besitzen häufig Eigenschaften, die sowohl notwendig als auch hinreichend sind. Diese Situation wird durch die folgende Formel beschrieben, welche eine Kurzform der Konjunktion der beiden oben stehenden Formeltypen ist:

$$\forall x C(x) \leftrightarrow \phi(x)$$

In unserem Beispiel wäre der Besitz von Wirbeln zwar eine notwendige, jedoch keine hinreichende Bedingung für eine Zugehörigkeit zur Klasse der Säugetiere, da zwar alle Säugetiere, aber auch Reptilien über Wirbel verfügen. Auf der anderen Seite ist der Besitz von Wirbeln (wie der Name schon sagt) eine hinreichende und in diesem Fall auch notwendige Eigenschaft in Bezug auf die Zugehörigkeit zur Klasse der Wirbeltiere. Fälle, in denen eine Eigenschaft hinreichend, aber nicht notwendig ist, kommen seltener vor, sind aber nicht ausgeschlossen. Ein Beispiel ist die Zugehörigkeit eines konkreten Jahres zur Klasse der Nicht-Schaltjahre. So sind Jahre, die nicht durch 4 ohne Rest teilbar sind, keine Schaltjahre. Da jedoch Jahre, die durch 100, jedoch nicht durch 400 ohne Rest teilbar sind, ebenfalls keine Schaltjahre sind, ist dies keine notwendige Bedingung für Nicht-Schaltjahre.

Die korrekte Darstellung der notwendigen Eigenschaft von Säugetieren, Wirbel zu besitzen, ist also die Formel:  $\forall x Mammal(x) \rightarrow \phi(x)$ , wobei  $\phi(x)$  den Besitz von Wirbeln beschreibt. Im Folgenden wollen wir nun die logische Definition von Eigenschaften genauer betrachten. Eine naive Beschreibung wäre die folgende:  $has(x, vertebra)$ . Diese Darstellung mischt jedoch wiederum Klassen und Objekte, da *vertebra* wiederum ein Konzept ist, welches alle möglichen Wirbel beschreibt. Eine korrekte Beschreibung ist also:

$$\forall x Mammal(x) \rightarrow \exists y has(x, y) \wedge Vertebra(y) \quad (2.4)$$

Diese Definition besagt nun, dass alle Säugetiere (mindestens einen) Wirbel besitzen, was der intendierten Bedeutung dieser Eigenschaft entspricht. Wir können jedoch feststellen, dass diese Interpretation von Eigenschaften selbst in unserem Beispiel nicht immer die einzig mögliche ist. Dies zeigt die Betrachtung der ebenfalls beschriebenen Eigenschaft von Fischen, im Wasser zu leben. Eine analoge Übersetzung dieser Eigenschaft führt zwar zu einem korrekten Ergebnis: Alle Fische leben in irgendeiner Form von Wasser. Eine andere Interpretation der im semantischen Netz dargestellten Eigenschaft ist jedoch, dass Fische ausschließlich im Wasser leben. Diese Interpretation kann auf folgende Weise dargestellt werden:

$$\forall x \text{ Fish}(x) \rightarrow (\forall y \text{ lives} - \text{in}(x, y) \rightarrow \text{Water}(y)) \quad (2.5)$$

Diese Formel besagt, dass alles, worin Fische leben, Wasser sein muss. In der Praxis ist es nicht immer ganz leicht, zwischen diesen beiden möglichen Formalisierungen von Konzepteneigenschaften zu unterscheiden und die richtige Formalisierung zu wählen. Beide Varianten werden jedoch regelmäßig verwendet und sind fester Bestandteil von Ontologiesprachen wie OWL.

Komplizierter wird die Formalisierung, wenn man es wie in dem Beispiel aus Abb. 2.3 mit Kardinalitätseinschränkungen und konkreten Werten zu tun hat. So beinhaltet das semantische Netz hier die Informationen, dass Säugetiere säugen und Haare besitzen. Diese Eigenschaften unterscheiden sich von den oben beschriebenen dadurch, dass keine Aussagen über den Typ des Zielobjektes gemacht wird. Wir können dies dadurch ausdrücken, dass die Existenz von bestimmten Objekten gefordert wird, ohne dass diese näher beschrieben werden:

$$\forall x \text{ Mammal}(x) \rightarrow \exists y \text{ lactates}(x, y) \quad (2.6)$$

$$\forall x \text{ Mammal}(x) \rightarrow \exists y \text{ has} - \text{hair}(x, y) \quad (2.7)$$

Eine weitere Herausforderung bilden die im Beispiel beschriebenen Eigenschaften des Konzeptes „Elephant“, welches aussagt, dass Elefanten vier Beine, einen Rüssel und einen Schwanz haben. Die Formalisierung dieser Eigenschaften erfordert die Spezifikation bestimmter Anzahlen von Objekten. Diese ist möglich, wenn wir die oben beschriebene einfache Prädikatenlogik um spezielle Prädikate  $=$  und  $\neq$  erweitern, welche die Gleichheit bzw. Ungleichheit von Variablen bezeichnen. Insbesondere ist die atomare Formel  $x = y$  genau dann wahr, wenn  $x$  und  $y$  mit dem gleichen Objekt instantiiert werden (also  $x^I = y^I$ ). Analog ist  $x \neq y$  genau dann wahr, wenn  $x^I \neq y^I$  gilt. Unter Verwendung dieser speziellen Prädikate lässt sich die Tatsache, dass ein Elefant mindestens 4 Beine hat, wie folgt beschreiben:

$$\begin{aligned} \forall x \, Elephant(x) \rightarrow \exists y_1, y_2, y_3, y_4 \\ & legs(x, y_1) \wedge legs(x, y_2) \wedge legs(x, y_3) \\ & \wedge legs(x, y_4) \wedge y_1 \neq y_2 \wedge y_1 \neq y_3 \\ & \wedge y_1 \neq y_4 \wedge y_2 \neq y_3 \\ & \wedge y_2 \neq y_4 \wedge y_3 \neq y_4 \end{aligned}$$

Da die Größe von Formeln dieser Art mit der Anzahl der notwendigen Variablen exponentiell zunimmt, wird in der Regel die folgende verkürzte Schreibweise verwendet, mit der wir ausdrücken können, dass Elefanten mindestens 4 Beine haben:

$$\forall x \, Elephant(x) \rightarrow \exists^{\geq 4} y \, legs(x, y) \quad (2.8)$$

Zusätzlich können wir unter Verwendung von Gleichheit und Ungleichheit die Tatsache formalisieren, dass ein Elefant höchstens 4 Beine hat. Die entsprechende Formel in Prädikatenlogik ist folgende:

$$\begin{aligned}
\forall x \textit{ Elephant}(x) \rightarrow \exists y_1, y_2, y_3, y_4, y_5 \\
& \textit{legs}(x, y_1) \wedge \textit{legs}(x, y_2) \wedge \textit{legs}(x, y_3) \\
& \wedge \textit{legs}(x, y_4) \wedge \textit{legs}(x, y_5) \rightarrow \\
& y_1 = y_2 \vee y_1 = y_3 \vee y_1 = y_4 \vee y_1 = y_5 \\
& \vee y_2 = y_3 \vee y_2 = y_4 \vee y_2 = y_5 \vee y_3 = y_4 \\
& \vee y_3 = y_5 \vee y_4 = y_5
\end{aligned}$$

Da hier ebenfalls die Größe der Formel exponentiell zunimmt, gibt es auch für diesen Fall eine abkürzende Schreibweise:

$$\forall x \textit{ Elephant}(x) \rightarrow \exists^{\leq 4} y \textit{ legs}(x, y) \quad (2.9)$$

Die Kombination dieser beiden Aussagen, die der gewünschten Aussage über die genaue Anzahl von Beinen entspricht, wird hierbei wie folgt abgekürzt:

$$\forall x \textit{ Elephant}(x) \rightarrow \exists^{\leq 4} y \textit{ legs}(x, y) \quad (2.10)$$

Mit Hilfe dieser Form der Beschreibung von Eigenschaften können nun auch die beiden anderen Eigenschaften, die sich auf die Anzahl von Rüsseln und Schwänzen beziehen, formalisiert werden:

$$\forall x \textit{ Elephant}(x) \rightarrow \exists^{\leq 1} y \textit{ trunk}(x, y) \quad (2.11)$$

$$\forall x \textit{ Elephant}(x) \rightarrow \exists^{\leq 1} y \textit{ tail}(x, y) \quad (2.12)$$

Eine letzte Variante der Formalisierung von Eigenschaften ist die Festlegung konkreter Eigenschaftswerte. Ein Beispiel hierfür findet sich auch in der Beschreibung der Elefanten-Klasse. So könnten die unterschiedlichen möglichen Hautfarben in Form konkreter Werte modelliert und in der Prädikatenlogik wie

Objekte als Konstanten dargestellt werden. Die entsprechende Beschreibung dieser Eigenschaft kann in Form der folgenden prädikatenlogischen Formel erfolgen:

$$\forall x \textit{ Elephant}(x) \rightarrow \textit{skin}(x, \textit{grey}) \quad (2.13)$$

Zusammengenommen ergibt sich eine vollständige logische Beschreibung der Objekte vom Typ Elefant durch die Formeln 2.6 bis 2.13 sowie die Implikation  $\forall x \textit{ Elephant}(x) \rightarrow \textit{Mammal}(x)$ .

## Kapitel 3

# Beispiele

In den vorangegangenen Kapiteln haben wir uns mit grundlegenden Konzepten und Ansätzen zur Repräsentation von ontologischem Wissen beschäftigt. Bevor wir im folgenden Teil dieses Buches auf konkrete Technologien eingehen, welche die Erstellung und Nutzung von Ontologien unterstützen, wollen wir uns zunächst einigen Beispielen bestehender Ontologien zuwenden. Diese Beispiele stellen die Verbindung zwischen den beschriebenen Prinzipien sowie den konkreten Anwendungen von Ontologien her, die im letzten Teil dieses Buches beschrieben werden. Die Beispiele wichtiger Ontologien zeigen, dass sowohl das Prinzip der semantischen Netze, als auch die Logik-basierte Darstellung von ontologischem Wissen in der Realität Anwendung findet.

Die Auswahl der Beispiele zeigt auch, dass es unterschiedliche Typen von Ontologien gibt. Der Typ einer Ontologie bestimmt sich in der Regel durch die Art von Wissen, die in Form von Kategorien und deren Eigenschaften dargestellt wird. Hierbei wird vor allem unterschieden, ob das dargestellte Wissen



allgemeiner Natur ist, also im Sinne klassischer Kategoriensysteme grundlegende Konzepte der Welt beschreibt, oder aber ob es eine spezielle Wissensdomäne, etwa die Medizin, beschreibt. Ontologien des ersten Typs werden auch als „Upper-level“, „Top-level“ oder „Foundational Ontologies“ bezeichnet. Wir nennen diese Art von Ontologien im Folgenden allgemeine Ontologien. Im Gegensatz dazu werden Ontologien, die ein spezielles Wissensgebiet abdecken, als Domänen-Ontologien bezeichnet. Die folgende Tabelle 3.1 enthält eine Reihe von Beispielen für bekannte Ontologien und deren Einordnung in das genannte Schema. Drei dieser Ontologien werden im Folgenden näher betrachtet.

**Tabelle 3.1** Beispiele für bekannte Ontologien

	allgemeines Wissen	medizinisches Wissen
semantisches Netz	Wordnet ( <a href="#">Kapitel 3.1</a> )	UMLS ( <a href="#">Kapitel 3.2</a> )
logisches Modell	SUMO ( <a href="#">Kapitel 3.3</a> )	GALEN

Gelegentlich werden auch noch weitere Typen von Ontologien genannt. So ist zuweilen von sogenannten „Application-“ (Anwendungs-) und „Task-“ (Aufgaben-) Ontologien die Rede. Diese Typen beziehen sich zum Teil auf die Rolle, welche die entsprechend bezeichneten Ontologien in einem System spielen. Anwendungsontologien werden hierbei Modelle genannt, deren Ziel weniger eine neutrale Beschreibung eines Wissensgebietes ist, sondern die spezielle Aspekte der Anwendung beschreiben. Im Bereich der Medizin etwa würde eine Domänen-Ontologie Konzepte wie unterschiedliche Typen von Krankheiten und Medikamenten darstellen, eine Anwendungs-Ontologie, die in einem Diagnose-Unterstützungssystem Verwendung finden

könnte, würde hingegen Konzepte wie Symptome, Ursache und Diagnose enthalten. Diese anwendungsspezifischen Konzepte müssten dann wiederum durch Elemente der Domänen-Ontologie instanziiert werden. In diesem Fall wären die in der Domänen-Ontologie enthaltenen Krankheiten mögliche Diagnosen. Aufgaben-Ontologien sind hierbei spezielle Anwendungs-Ontologien, die Definitionen enthalten, mit deren Hilfe eine gegebene Aufgabenstellung beschrieben werden kann. Im Fall einer medizinischen Diagnose wäre die Aufgabenstellung zum Beispiel, auf der Grundlage beobachteter Symptome Hypothesen über mögliche Diagnosen zu erzeugen und auf der Grundlage dieser Hypothesen, Behandlungsvorschläge zu unterbreiten. Die Begriffe „Symptome“, „Hypothesen“ und „Behandlungsmöglichkeiten“ wären hierbei typischerweise in einer entsprechenden Aufgaben-Ontologie definiert. Wie im Fall der Anwendungs-Ontologien müsste auch hier wieder eine Verbindung zu einer geeigneten Domänen-Ontologie hergestellt werden, indem z.B. Behandlungsmöglichkeiten durch unterschiedliche Arten von Medikamenten instanziiert werden.

### **3.1 WordNet – ein semantisches Lexikon**

Das wohl bekannteste Beispiel eines semantischen Netzes ist die lexikalische Datenbank WordNet. Ziel von WordNet ist die Definition der Bedeutung von Begriffen in der englischen Sprache in Form eines semantischen Netzes. Hierbei geht es nicht primär um die formale Definition von Begriffen, sondern vielmehr darum, eine eindeutige Beziehung zwischen Erscheinungsformen von Begriffen in Form von Wörtern und deren intendierten Bedeutungen herzustellen. WordNet folgt also dem in

**Kapitel 1** beschriebenen Prinzip der linguistischen Semantik. Grundannahme ist die Existenz einer sogenannten lexikalischen Matrix, welche die komplexe Relation zwischen Wortformen, also Worten, wie sie in Sprache und Texten auftauchen, und ihren Wortbedeutungen darstellt. Die folgende Abbildung illustriert die Rolle dieser lexikalischen Matrix anhand einer Reihe von englischen Substantiven.

	move	impress	strike	motion	movement	work	go	run	test
s1	1	1	1	0	0	0	0	0	0
s2	1	0	0	1	1	0	0	0	0
s3	1	0	0	0	0	0	1	1	0
s4	0	0	0	0	0	1	1	1	0
s5	0	0	0	0	0	0	0	1	1
...									

**Abb. 3.1** Ausschnitt aus der lexikalischen Matrix

In der Abbildung werden die englischen Begriffen „move“, „impress“, „strike“, „motion“, „movement“, „go“, „work“, „run“ und „test“ beschrieben, die jeweils eine Spalte der Matrix einnehmen. Die Zeilen bilden eine Reihe unterschiedlicher möglicher Bedeutungen dieser Begriffe, die als  $s_1$  bis  $s_5$  bezeichnet werden. Die Einträge in der Matrix weisen Begriffen mögliche Bedeutungen zu. Wie man anhand des Beispiels sehen kann, ist diese Relation in beiden Richtungen uneindeutig. So kann die Bedeutung  $s_5$ , die im Englischen sinngemäß einen Testlauf beschreibt, sowohl durch den Begriff „test“ als auch den Begriff „run“ beschrieben werden. Auf der anderen Seite kann der Begriff „run“ nicht nur im Sinne eines Testlaufs, sondern auch im Sinne einer Bewegung, also der Bedeutung  $s_4$  interpretiert werden, die wiederum sowohl durch den Begriff „run“ als auch

durch die Begriffe „go“ und „move“ beschrieben wird. WordNet hat sich nun zum Ziel gesetzt, die unterschiedlichen möglichen Bedeutungen englischer Begriffe wie „move“ und „run“ in Form eines semantischen Netzwerkes darzustellen. Insbesondere werden unterschiedliche mögliche Bedeutungen eines Begriffes durch semantische Relationen zu anderen Begriffen beschrieben. Ein einzelner Begriff wie „run“ taucht hierbei mehrmals im Netz auf. Jedes Vorkommen steht für eine mögliche Bedeutung. Wir haben es also nicht allein mit dem Begriff 'run' zu tun, sondern mit  $run_1$ ,  $run_2$  und  $run_3$ , die jeweils für unterschiedliche Bedeutungen des Begriffes stehen. Die Beschreibung dieser unterschiedlichen Bedeutungen erfolgt nun durch die bereits bekannten Prinzipien semantischer Netze, nämlich über Relationen zu anderen Elementen im Netz. WordNet verwendet hierzu die folgenden linguistischen Relationen:

### **Synonymie**

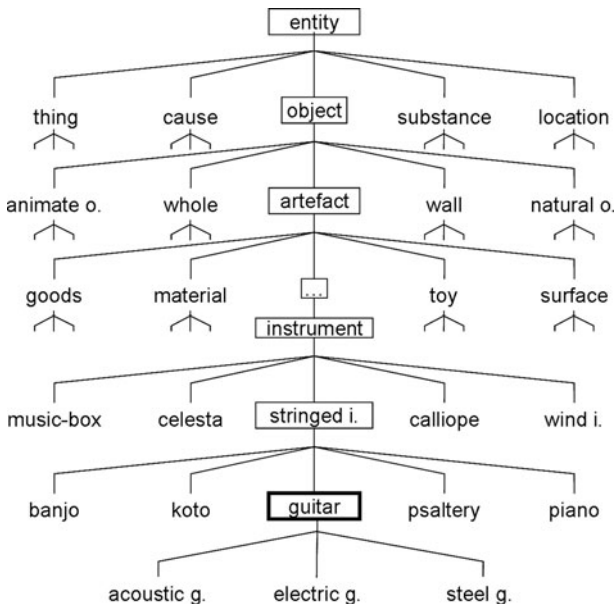
Synonymie bezeichnet eine Relation zwischen Begriffen, die sich direkt aus der lexikalischen Matrix ergibt. Zwei Begriffe sind Synonyme, wenn sie der gleichen Bedeutung in der lexikalischen Matrix zugeordnet sind. Alle Einträge derselben Zeile der lexikalischen Matrix stehen also in Beziehung zueinander. Wie bereits das oben genannte Beispiel zeigt, spielt die Synonymie von Begriffen eine wesentliche Rolle bei der Definition von Wortbedeutungen. Anhand der Beispielmatrix ist zu sehen, dass sich diese aus der Menge aller synonymen Begriffe ergibt, die diese Bedeutung tragen. Daher kann über diese Menge, die auch Synonymmenge genannt wird, eindeutig bestimmt werden. So ergibt sich folgende Definition der Wortbedeutung aus unserem Beispiel:

$$\begin{aligned}
s_1 &= \{move_1, impress_1, strike_1\} \\
s_2 &= \{move_2, motion_1, movement_1\} \\
s_3 &= \{move_3, go_1, run_1\} \\
s_4 &= \{work_1, go_2, run_2\} \\
s_5 &= \{run_2, test_1\}
\end{aligned}$$

Anhand dieser Definitionen ist es nun auch möglich, die Bedeutung eines bestimmten Begriffes genau anzugeben, indem auf die entsprechende Synonymmenge verwiesen wird. Diese Definitionen bilden nun die Knoten in dem semantischen Netz. Alle weiteren Beziehungen, die im Folgenden vorgestellt werden, beziehen sich jeweils auf das Verhältnis von Synonymmengen zueinander. Hierdurch erhält die Synonym-Beziehung einen besonderen Stellenwert als Grundlage für die Definition von Wortbedeutungen in WordNet.

## Hyponymie

Die Hyponymie-Beziehung ist das linguistische Gegenstück zu der im Kontext semantischer Netze angesprochenen AKO-Relation, welche eine Beziehung zwischen Unter- und Oberkategorien bezeichnet. Wie bereits erwähnt, ist diese Relation in WordNet aus Wortbedeutungen in Form von Synonymmengen definiert. Eine Synonymmenge  $U$  gilt hierbei als Hyponym einer anderen Menge  $O$ , wenn die durch Begriffe aus  $U$  bezeichneten Objekte eine Teilmenge der durch Begriffe aus  $O$  bezeichneten Objekte sind. Im Kontext von WordNet wird häufig auch eine alternative Definition verwendet, die besagt, dass eine Hyponym-Beziehung besteht, wenn ein Muttersprachler für alle Worte  $o \in O$  und  $u \in U$  den Satz „Ein  $u$  ist eine Art von  $o$ “ als wahr bezeichnen würden. Jede Wortbedeutung kann hierbei nur



**Abb. 3.2** Ausschnitt aus der Hyponym-Hierarchie von WordNet

mit einer anderen in einer direkten Hyponym-Beziehung stehen. Hierdurch werden die Wortbedeutungen in einer Baumstruktur angeordnet. Abbildung 3.2 zeigt einen kleinen Ausschnitt dieser Struktur.

Anhand des Begriffes „guitar“ kann man die Interpretation der Hyponymie-Beziehung nachvollziehen. Der Begriff ist als Hyponym von „stringed instrument“ definiert, da der Satz „Gitarre ist eine Art von Saiteninstrument“ im Allgemeinen als wahr angesehen wird. Abbildung 3.2 zeigt, dass das semantische Netz aus Hyponymie-Beziehungen in WordNet den gesamten Bereich

von sehr allgemeinen Begriffen, wie sie auch in klassischen Kategoriensystemen vorkommen (etwa „entity“, „object“ und „artefact“), bis hin zu relativ konkreten Begriffen der Umgangssprache wie etwa „guitar“ abdeckt.

Um auch die gesamte Breite der natürlichen Sprache berücksichtigen zu können, beschränkt sich WordNet nicht auf eine einzelne Hierarchie, in der, wie in der Abbildung dargestellt, Substantive zu unterschiedlichen Arten von Objekten beschrieben werden. Vielmehr besteht das semantische Netz aus 26 voneinander unabhängigen Teilbäumen mit den folgenden Wurzelbegriffen:

{act, action, activity}	{food}	{process}
{animal, fauna}	{group, collection}	{quantity, amount}
{artifact}	{location, place}	{relation}
{attribute, property}	{motive}	{shape}
{body, corpus}	{natural object}	{society}
{cognition, ideation}	{natural phenomenon}	{state, condition}
{communication}	{person, human being}	{substance}
{event, happening}	{plant, flora}	{time}
{feeling, emotion}	{possession, property}	

Alle in WordNet enthaltenen Wortbedeutungen sind indirekt Hyponyme einer dieser Bedeutungen, die in dieser Form eine Art Kategoriensystem bilden, welches jedoch nicht explizit spezifiziert, sondern von den WordNet-Entwicklern vorgegeben wurde. Das semantische Netz an sich bildet eine Verfeinerung dieses Kategoriensystems auf der Grundlage von Synonymmen.

## Meronymie

Meronymie bezeichnet die semantische Beziehung zwischen einem Objekt und seinen Teilen. Viele Begriffe, vor allem wenn

diese physikalische Objekte beschreiben, können durch diese Teil-Ganzes-Relation präziser beschrieben werden als durch die Vererbungsrelation allein. Aus diesem Grund besitzt die Teil-Ganzes-Relation in vielen Ontologien, wie auch in WordNet, einen besonderen Status und wurde ähnlich der Vererbungsrelation intensiv untersucht. Während über die Interpretation der letzteren als Teilmengen-Beziehung über die beschriebenen Begriffe weitgehend Einigkeit herrscht, scheint die Entwicklung eines einheitlichen Modells zur Interpretation der Teil-Ganzes-Relation nicht ohne Weiteres möglich zu sein. Dies liegt daran, dass die Teil-Ganzes-Relation im allgemeinen Sprachgebrauch für ganz unterschiedliche Phänomene eingesetzt wird:

- component-object (branch/tree)
- member-collection (tree/forest)
- portion-mass(slice/cake)
- stuff-object (aluminum/airplane)
- feature-activity (paying/shopping)
- place-area (Princeton/New Jersey)
- phase-process (adolescence/growing up).

Diese unterschiedlichen Arten von Teil-Ganzes-Relationen zeichnen sich zum Teil durch recht unterschiedliche formale Eigenschaften aus. Eine zentrale Frage ist hier unter anderem die nach der Transitivität der Teil-Ganzes-Relation. Während die meisten Menschen diese Frage intuitiv mit „ja“ beantworten würden, enthält die einschlägige Literatur eine Vielzahl von Gegenbeispielen wie das folgende:

*steel* part – of *ship* (3.1)

*ship* part – of *fleet* (3.2)



Trotz dieser beiden Aussagen würde wohl niemand behaupten, dass auch die folgende Aussage wahr ist:

$$\textit{steelpart} - \textit{offleet} \quad (3.3)$$

Es lassen sich beliebig viele solcher Beispiele finden, die scheinbar eindeutig gegen die Transitivität der Teil-Ganzes-Relation sprechen. Die meisten dieser Beispiele kann man jedoch dadurch erklären, dass sie unterschiedliche Arten von Teil-Ganzes-Relationen mischen. Im obigen Fall verwendet Aussage 3.1 die „stuff-object“ Beziehung, während Aussage 3.2 die „member-collection“ Relation verwendet. Es lassen sich andere Beispiele finden, in denen jeweils der gleiche Relationstyp verwendet wird und in denen man von einer Transitivität ausgehen kann. Ein Beispiel ist das folgende:

$$\textit{second part} - \textit{of minute} \quad (3.4)$$

$$\textit{minute part} - \textit{of hour} \quad (3.5)$$

$$\textit{second part} - \textit{of hour} \quad (3.6)$$

Auch innerhalb eines Relationstyps treten immer wieder Fälle auf, in denen keine Transitivität gegeben ist. Diese Fälle und generelle Aussagen über die formalen Eigenschaften würden den Rahmen der Betrachtung sprengen. Festzuhalten bleibt, dass die Teil-Ganzes-Relation eine wichtige Rolle bei der semantischen Beschreibung spielt und dass der Einsatz dieser Relation eine gewisse Sorgfalt voraussetzt. Darüber hinaus ist es oft notwendig, unterschiedliche Arten von Teil-Ganzes Relationen zu berücksichtigen. Im Kontext von WordNet werden drei Typen von Teil-Ganzes Relationen betrachtet:

**part-meronym:** beschreibt die Beziehung zwischen einem Objekt und den Komponenten, aus denen dieses Objekt zusammengesetzt ist (zum Beispiel: Schiff-Bug)

**member-meronym:** beschreibt die Beziehung zwischen einem Objekt und einer Gruppe, zu der dieses Objekt gehört (zum Beispiel: Schiff-Flotte)

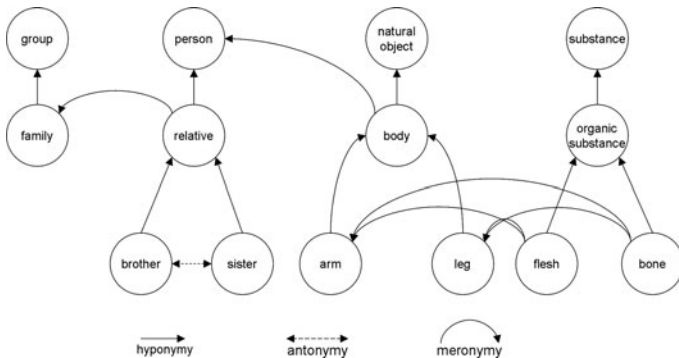
**substance-meronym:** beschreibt die Beziehung zwischen einer Substanz und Substanzen, aus denen diese besteht bzw. hergestellt wurde (zum Beispiel: Stahl-Eisen)

Wie das Beispiel des Schiffes zeigt, kann der gleiche Begriff durch mehrere dieser Teil-Ganzes-Beziehungen beschrieben werden.

### Antonymie

Eine weitere in WordNet verwendete Relation, die jedoch nicht die Wichtigkeit der bereits vorgestellten Relationen besitzt, ist die Antonymie. Diese verbindet Begriffe, die Gegensätze beschreiben. Beispiele sind: Helligkeit – Dunkelheit, Armut – Reichtum, aber auch Bruder – Schwester. Eine formale Definition der Antonymie ist hierbei nicht ganz einfach. Ihre Bedeutung ist vielschichtiger als die logische Negation, da Begriffe wie Armut und Reichtum häufig nur in einem bestimmten Kontext interpretiert werden können. Jemand, der in Indien reich ist, muss dies nicht auch in der Schweiz sein.

Die genannten Relationen bilden ein komplexes Netzwerk von Synonymmengen. Abbildung 3.3 zeigt einen kleinen Ausschnitt aus diesem Netzwerk, welches sowohl Vererbungs- als auch Teil-Ganzes-Beziehungen und Antonymie-Beziehungen enthält. Man sieht deutlich den komplementären Charakter von Vererbungs- und Teil-Ganzes-Beziehungen. Das Beispiel beschreibt unterschiedliche Aspekte von Verwandten. Zum einen wird deren Zugehörigkeit zu einer Familie über die „member-meronym“ Beziehung beschrieben. Zum anderen wird festgelegt, dass Verwandte, ebenso wie alle Personen, unter



**Abb. 3.3** Beispiel der Interaktion semantischer Relationen in WordNet

anderem aus einem Körper bestehen, der wiederum aus weiteren Teilen wie Armen und Beinen besteht. Diese Beziehung wird jeweils über die „part-meronym“ Beziehung beschrieben. Schließlich wird dargestellt, dass Arme und Beine aus Fleisch und Knochen bestehen, was der „substance-meronym“ Beziehung entspricht. Darüber hinaus werden all diese Teile in eine entsprechende Vererbungshierarchie eingeordnet (Fleisch und Knochen sind Substanzen, der menschliche Körper ist ein natürliches Objekt usw.). Zusätzlich werden die Begriffe Bruder und Schwester, die Antonyme darstellen, als spezielle Form von Verwandten beschrieben. Aus dem Beispiel ist ebenfalls ersichtlich, dass eine Interaktion zwischen den unterschiedlichen Relationen besteht. Insbesondere werden Teil-Ganzes-Relationen entlang der Vererbungsrelationen an die Unterkonzepte weitergegeben: Verwandte sind Personen und haben daher auch einen Körper als Teil, Brüder und Schwestern sind Verwandte und daher auch Teil einer Familie. Diese komplexen Interaktionen werden in WordNet zwar angenommen und auch in den entsprechenden Veröffentlichungen beschrieben, es

gibt jedoch keinen eingebauten Mechanismus, etwa im Sinne einer logischen Semantik, der eine Ableitung des entsprechenden Wissens erlauben würde. Diese Form des Schließens muss in WordNet daher als Teil der Anwendung, in der das semantische Netz verwendet wird, explizit implementiert werden.

Die bisherigen Beschreibungen betreffen lediglich Substantive, die allerdings den Großteil des Inhaltes darstellen. Darüber hinaus beinhaltet WordNet auch semantische Netze, in denen Adjektive und Verben beschrieben werden. Diese Teilnetze verwenden andere semantische Relationen zur Beschreibung der enthaltenen Wortbedeutungen, die jedoch an dieser Stelle nicht thematisiert werden sollen.

## **3.2 UMLS – Unified Medical Language System**

Kontrollierte Vokabulare spielen in der Medizin seit längerem eine wichtige Rolle. Grund ist vor allem der Bedarf einer lückenlosen Qualitätssicherung, die es erfordert, eine weitreichende Standardisierung von Behandlungen und deren Dokumentation zu erreichen. Hierfür ist es unerlässlich, Ungenauigkeiten durch mehrdeutige Bezeichnungen weitestgehend auszuschließen. Einige der wichtigsten Vokabulare aus dem Bereich der Medizin sind die folgenden:

**ICD10** Die „International Classification of Diseases“ ist ein von der Weltgesundheitsorganisation entwickeltes Vokabular zur eindeutigen Beschreibung von Krankheiten und Symptomen. Entsprechende Krankheitsbilder sind hierarchisch angeordnet und mit einer eindeutigen Schlüsselnummer versehen. In Deutschland wird ICD10 zur eindeutigen Beschrei-

bung von Diagnosen sowie zur Dokumentation von Todesursachen verwendet und bildet somit die Basis für die Kommunikation zwischen unterschiedlichen Ärzten sowie für die medizinische Statistik.

**SNOMED** Die „Systematized Nomenclature of Medicine“ ist ein umfassendes medizinisches Vokabular, welches sich nicht wie ICD10 auf Krankheitsbilder beschränkt, sondern auch andere Aspekte wie Behandlungen, Medikamente und Mikroorganismen umfasst. SNOMED wird vor allem im englischsprachigen Raum als Grundlage für den elektronischen Austausch medizinischer Informationen verwendet.

**MeSH** Die „Medical Subject Headings“ sind ein mehrsprachiger Thesaurus zur eindeutigen Annotation medizinischer Publikationen mit inhaltlichen Beschreibungen. Hauptanwendungsgebiet ist die Unterstützung der Suche in Literaturdatenbanken. Insbesondere ist MeSH die Basis von Medline, einer Literaturdatenbank mit mehr als 17 Millionen Veröffentlichungen aus dem Bereich der Medizin, die ein wesentliches Instrument der medizinischen Forschung darstellt.

Die genannten Vokabulare wurden in der Regel unabhängig voneinander entwickelt und unterscheiden sich im thematischen Fokus sowie Detaillierungsgrad.

Das Unified Medical Language System ist eine Entwicklung der National Library of Medicine der Vereinigten Staaten. Ziel von UMLS ist die Integration unterschiedlicher Vokabulare aus dem Bereich der Medizin, wie etwa der oben genannten, in ein Gesamtmodell, welches die Analyse medizinischer Fachtexte unterstützen soll. Hierzu besteht UMLS aus drei miteinander verbundenen Modellen, die jeweils unterschiedliche Aspekte der Bedeutung von Begriffen aus dem Bereich der Medizin beschreiben. Im Rahmen dieses Buches sind vor allem zwei dieser Modelle relevant.

- Der UMLS-Metathesaurus ordnet Begriffe aus unterschiedlichen medizinischen Terminologien Konzepten zu und ordnet diese in eine einheitliche Konzepthierarchie ein.
- Das UMLS-Semantic Network ist ein semantisches Netz, welches wichtige medizinische Konzepte und deren Relationen zueinander abbildet. Begriffe im Metathesaurus sind jeweils einem Konzept in diesem semantischen Netz zugeordnet.

Beide Modelle können als semantische Netze bezeichnet werden, sie unterscheiden sich jedoch in Aufbau und Zweck. Der Metathesaurus ist von seiner Struktur her ähnlich aufgebaut wie WordNet: Es wird lediglich eine begrenzte Anzahl fest vorgegebener Relationen verwendet, um die Beziehung zwischen Begriffen zu beschreiben. Die wesentlichen Relationen sind hierbei Synonymie sowie Hyponymie von Begriffen. Ziel ist, wie bei WordNet, die Disambiguierung der Bedeutung entsprechender Begriffe. Das UMLS-Semantic Network auf der anderen Seite ist eher ein semantisches Netz im klassischen Sinne, in dem eine Vielzahl unterschiedlicher Relationen verwendet wird, um Wissen über eine begrenzte Anzahl zentraler medizinischer Konzepte zu beschreiben. Ziel ist hierbei nicht allein die Disambiguierung von Begriffen, sondern auch die explizite Darstellung von Wissen über wichtige Relationen im Bereich der Medizin. Beide Modelle und ihre charakteristischen Eigenschaften werden im Folgenden genauer beschrieben.

### ***3.2.1 Der Metathesaurus***

Der Metathesaurus ist das zentrale Modell des Unified Medical Language System. Er vereint Begriffe aus mehr als einhundert unterschiedlichen Quellen in einem gemeinsamen Modell.

Dieses enthält mehr als 7,5 Millionen Begriffe, die mehr als 1,5 Millionen unterschiedliche Konzepte beschreiben. Um das Modell trotz dieser Größe handhabbar zu machen, können dynamisch Teilmodelle generiert werden, die lediglich die Begriffe bestimmter Quellen enthalten. Um dies zu ermöglichen, besitzt der Metathesaurus eine spezielle Struktur, die im Folgenden beschrieben wird:

## Integration von Quellen

Wesentliches Strukturierungselement des UMLS-Metathesaurus ist die Zusammenfassung von Begriffen aus unterschiedlichen Quellen in gemeinsame Konzepte, die, ähnlich wie in WordNet, über Synonyme (Terms) beschrieben werden. Außerdem beinhaltet der Metathesaurus Informationen über lexikalische Varianten (Strings), in denen diese Begriffe auftreten können. Dies können zum Beispiel Pluralformen oder unterschiedliche Schreibweisen sein. Diese lexikalischen Varianten sind wiederum mit sogenannten Atomen verbunden, die jeweils das Auftreten der entsprechenden Schreibweise in den unterschiedlichen Quellen wie MeSH oder SNOMED darstellen. Die Elemente besitzen jeweils eindeutige Kennnummern, um beispielsweise identische Begriffe aus unterschiedlichen Quellen unterscheiden zu können. Hierdurch werden Begriffe aus den unterschiedlichsten Quellen in ein gemeinsames Konzept zusammengefasst, ohne dass die individuellen Unterschiede bei der Bezeichnung von Begriffen und Konzepten verloren gehen. Abbildung 3.4 veranschaulicht dieses Prinzip anhand des Konzeptes „Atrial Fibrillation“.

Die Abbildung zeigt das als „Atrial Fibrillation“ bezeichnete Konzept mit der Kennnummer C0004238. Hierbei bezeichnet das führende C, dass es sich um ein Konzept handelt.

Concept (CUI)	Terms (LUIs)	Strings (SUIs)	Atoms (AUIs) * RRF Only
<b>C0004238</b> Atrial Fibrillation (preferred) Atrial Fibrillations Auricular Fibrillation Auricular Fibrillations	<b>L0004238</b> Atrial Fibrillation (preferred) Atrial Fibrillations	<b>S0016668</b> Atrial Fibrillation (preferred)	<b>A0027665</b> Atrial Fibrillation (from MSH)  <b>A0027667</b> Atrial Fibrillation (from PSY)
		<b>S0016669</b> Atrial Fibrillations	<b>A0027668</b> Atrial Fibrillations (from MSH)
	<b>L0004327</b> (synonym) Auricular Fibrillation Auricular Fibrillations	<b>S0016899</b> Auricular Fibrillation (preferred)	<b>A0027930</b> Auricular Fibrillation (from PSY)
		<b>S0016900</b> (plural variant) Auricular Fibrillations	<b>A0027932</b> Auricular Fibrillations (from MSH)

**Abb. 3.4** Konzepte, Begriffe, Worte und externe Quellen im UMLS Metathesaurus

Dieses Konzept kann durch die beiden synonymen Begriffe „Atrial Fibrillation“ (L0004238) und „Auricular Fibrillation“ (L0004327) beschrieben werden. Jeder dieser Begriffe kann in Singular- oder Pluralform vorkommen. Die entsprechenden Formen sind ebenfalls im Metathesaurus vorhanden und mit eigenen Kennnummern versehen. Die Verbindung mit den Quellen, aus denen diese Begriffe stammen, wird auf der Ebene dieser unterschiedlichen Ausprägungen hergestellt. So ist die Ausprägung „Atrial Fibrillation“ sowohl in MeSH – in der Abbildung bezeichnet durch das Kürzel MSH – vorhanden, als auch im „Thesaurus of Psychological Index Terms“, einer weiteren Quelle, die in den Metathesaurus integriert ist und hier durch das Kürzel PSY bezeichnet wird. Da die in UMLS integrierten Quellen meist



ebenfalls Informationen über Synonyme und lexikalische Variationen enthalten, ist es nicht erstaunlich, dass auch der Begriff „Aurical Fibrillation“ und bei MeSH zusätzlich die Pluralform „Artrial Fibrillations“ in diesen externen Quellen enthalten sind. Entsprechende Atome sind, wie die Abbildung zeigt, in der Konzeptbeschreibung enthalten.

## Disambiguierung von Begriffen

Die oben beschriebenen Mechanismen zur Zusammenführung von Begriffen aus unterschiedlichen Vokabularen kann ebenfalls benutzt werden, um homonyme Begriffe zu disambiguieren. Analog zu WordNet geschieht dies durch die Zuordnung dieser Begriffe zu unterschiedlichen Konzepten, die wiederum durch die Menge synonyme Begriffe beschrieben werden. Durch die Verbindung der Begriffe in den unterschiedlichen Quellen können hierbei auch unterschiedliche Verwendungen der Begriffe in diesen Quellen dokumentiert und die falsche Interpretation von Daten, die auf der Basis unterschiedlicher Vokabulare beschrieben wurde, verhindert werden. Abbildung 3.5 veranschaulicht dieses Prinzip am Beispiel des Begriffes „cold“.

Der Begriff „cold“ kann in UMLS für drei unterschiedliche Konzepte stehen. Diese sind auf der linken Seite der Tabelle in Abb. 3.5 dargestellt. Zum einen bezeichnet der Begriff das Konzept „cold temperature“, also eine niedrige Temperatur, zum anderen kann es zwei Krankheitsbilder beschreiben. Dies ist zum einen die „common cold“ also die klassische Erkältung. Zum anderen kann „cold“ als Abkürzung für „Chronic Obstructive Airways Disease“, also eine chronische Atemwegserkrankung stehen. Durch die explizite Verbindung

Concepts (CUIs)	Terms (LUIs)	String (SUIs)	Atoms (AUIs) ** RRF only
<b>C0009264</b> cold temperature	<b>L0215040</b> cold temperature	<b>S0288775</b> cold temperature	<b>A0318651</b> cold temperature (from CSP)
	<b>L0009264</b> Cold Cold	<b>S0007170</b> Cold	<b>A0016032</b> Cold (from MTH)
		<b>S0026353</b> Cold	<b>A0040712</b> Cold (from MSH)
<b>C0009443</b> Common Cold	<b>L0009443</b> Common Cold	<b>S0026747</b> Common Cold	<b>A0041261</b> Common Cold (from MSH)
	<b>L0009264</b> Cold Cold	<b>S0007171</b> Cold	<b>A0016033</b> Cold (from MTH)
		<b>S0026353</b> Cold	<b>A0040708</b> Cold (from COSTAR)
<b>C0024117</b> Chronic Obstructive Airway Disease	<b>L0498186</b> Chronic Obstructive Airway Disease	<b>S0837575</b> Chronic Obstructive Airway Disease	<b>A0896021</b> Chronic Obstructive Airway Disease (from MSH)
	<b>L0008703</b> Chronic Obstructive Lung Disease	<b>S0837576</b> Chronic Obstructive Lung Disease	<b>A0896023</b> Chronic Obstructive Lung Disease (from MSH)
	<b>L0009264</b> COLD COLD	<b>S0829315</b> COLD	<b>A0887858</b> COLD (from MTH)
		<b>S0474508</b> COLD	<b>A0539536</b> COLD (from SNMI)

**Abb. 3.5** Beispiel für die Disambiguierung des Begriffs „cold“ im Metathesaurus

der entsprechenden Konzepte mit in den verschiedenen Quellen enthaltenen Begriffen kann die Bedeutung dieser Begriffe im Kontext ihrer Quellen abgelesen werden. So unterscheidet MeSH explizit zwischen „cold“, „common cold“ und „chronic obstructive airways disease“ bzw. dem Synonym „chronic obstructive lung disease“, wie man an den entsprechenden Einträgen in der rechten äußeren Spalte der Tabelle sehen kann. Dies bedeutet, dass der Begriff „cold“ in MeSH eindeutig das Konzept „cold temperature“ bezeichnet, während die „Computer-Stored Ambulatory Records (COSTAR)“ mit dem Begriff „cold“, das Konzept „common cold“ und SNOMED (SMNI) das Konzept „chronic obstructive Airways disease“ bezeichnet.

Das Beispiel demonstriert sehr schön eines der großen ungelösten Forschungsprobleme im Bereich von Ontologien, das in diesem Buch jedoch nur oberflächlich behandelt werden kann, nämlich das Problem der semantischen Integration unterschiedlicher Ontologien. Wie gezeigt verwenden auch Ontologien unter Umständen die gleichen Begriffe zur Beschreibung unterschiedlicher Konzepte sowie unterschiedliche Begriffe zur Beschreibung gleicher Konzepte. Will man nun Daten zwischen Quellen, die unterschiedliche Ontologien verwenden, austauschen, so muss zunächst der Zusammenhang zwischen verwendeten Begriffen und Konzepten aufgeklärt werden. Im Fall von UMLS wurden die gezeigten Zusammenhänge zwischen unterschiedlichen Quellen in mühsamer Kleinarbeit manuell erstellt. Idealerweise will man diesen Prozess so weit wie möglich automatisieren. Obwohl eine ganze Reihe unterschiedlicher Ansätze zur Automatisierung dieses Prozesses entwickelt wurden, konnte dieses Problem bisher nicht zufriedenstellend gelöst werden. Weitere Aspekte der Integration von Ontologien und Daten werden im dritten Teil dieses Buches behandelt.

## Relationen im Modell

Die oben erläuterte Struktur zur Beschreibung von Konzepten basiert auf einer Reihe von Relationen, die Konzepte mit Begriffen, Ausprägungen sowie deren Auftreten in den verschiedenen Quellen verbinden. In den Abb. 3.4 und 3.5 wurden diese Relationen implizit durch die Nachbarschaft innerhalb der Zeilen der Tabelle dargestellt. Die wichtigste Relation ist hierbei sicherlich die Verbindung von Konzepten mit Begriffen, die im Wesentlichen der Synonym-Beziehung in WordNet entspricht. Zusätzlich enthält der Metathesaurus Relationen zwischen direkten über- und untergeordneten Konzepten, die diese in eine Konzepthierarchie einordnen. Diese Relationen entsprechen im Wesentlichen den aus WordNet bekannten Hyper- und Hyponym-Relationen. Ein spezielles Merkmal dieser Relationen ist, dass sie Begriffe aus unterschiedlichen Quellen verbinden und sie somit den eigentlichen Inhalt des Metathesaurus darstellen, der an sich keine neuen Begriffe definiert.

Abgesehen von dieser recht begrenzten Menge von Relationen enthält der Metathesaurus Relationen zwischen Begriffen, die sich in den integrierten Quellen finden. Diese Relationen verbinden jeweils nur die Vorkommen der Begriffe der entsprechenden Quelle und können je nach Quelle sehr unterschiedlich sein. Das ICD10-Vokabular beispielsweise besteht lediglich aus einer Konzepthierarchie und weist ansonsten keine weiteren Relationen zwischen Krankheitsbildern auf. MeSH enthält neben den typischen Relationen für Synonymie und Hyponymie auch Verweise auf ähnliche oder verwandte Begriffe. SNOMED schließlich enthält zusätzlich zu den genannten eine ganze Reihe unterschiedlicher Relationen, die beispielsweise ein Krankheitsbild (postoperative Esophagitis) mit der betroffenen Anatomie (Esophagus) und der Art des Problems (Entzündung) verbinden. Aufgrund dieser Unterschiede werden Relationen aus den un-

terschiedlichen Quellen separat behandelt und mit der jeweiligen Quelle gekennzeichnet. Einzige Form der Integration ist das Prinzip der Strukturhaltung im Gesamtmodell, welches besagt, dass Begriffe, die in einer der Quelle als Synonyme bzw. Hyponyme ausgewiesen sind, auf der Ebene der Begriffe im Metathesaurus ebenfalls in dieser Relation stehen. Im Fall von widersprüchlichen Definitionen in den verschiedenen Quellen werden alle dort enthaltenen Synonym- und Hyponym-Relationen übernommen. Die Hierarchien der einzelnen in den Metathesaurus integrierten Quellen sind somit in die Gesamthierarchie des Thesaurus eingebettet.

### ***3.2.2 Das semantische Netz***

Das semantische Netz ist grob gesagt eine Ergänzung des Metathesaurus um Informationen über Relationen zwischen Konzepten in den unterschiedlichen, in den Thesaurus integrierten Quellen. Da eine vollständige Spezifikation dieser Relationen zu aufwändig wäre, beschränkt sich das semantische Netz auf die Beschreibung von Relationen bezüglich einer kleinen Anzahl sehr allgemeiner Konzepte, die als „Semantic Types“ bezeichnet werden und die Knoten des semantischen Netzes bilden. Diese Konzepte sind so gewählt, dass sie das gesamte Themengebiet des Thesaurus abdecken. Jedes Konzept im Thesaurus ist einem dieser Semantic Types zugeordnet. Sind zwei Konzepte aus dem Thesaurus, die möglicherweise aus unterschiedlichen Quellen stammen, Semantic Types zugeordnet, die im semantischen Netz durch eine bestimmte Relation verbunden sind, so besagt dies, dass die entsprechende Relation auch zwischen diesen beiden Konzepten bestehen könnte. Auf diese Weise werden, wenn auch auf abstrakter Ebene, Informationen über mögliche Rela-

tionen zwischen Konzepten aus den unterschiedlichen Quellen dargestellt. Abbildung 3.6 zeigt einen Ausschnitt aus dem semantischen Netz mit Semantic Types zu den Themenbereichen Organismen, anatomische Strukturen und biologische Funktionen sowie den entsprechenden Relationen zwischen den Semantic Types. Andere Themenbereiche, die im semantischen Netz abgedeckt werden, sind chemische Substanzen, Ereignisse, physikalische Objekte und Konzepte oder Ideen. Insgesamt besteht das semantische Netz aus etwa 150 Semantic Types und 50 Relationen.

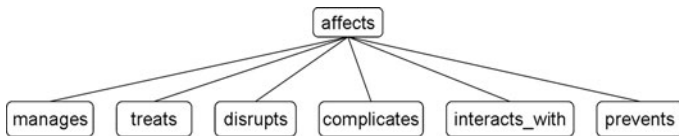
Aufgrund der oben genannten Zielsetzung spielen Relationen im semantischen Netz eine besondere Rolle. Daher sind hier weitergehende Möglichkeiten der Beschreibung von Relationen vorgesehen als in den bisher beschriebenen Modellen. Insbesondere ordnet das semantische Netz Relationen in Hierarchien an. Hierdurch ist es möglich, spezielle Varianten allgemeinerer Relationen zu beschreiben. Abbildung 3.7 zeigt ein Beispiel einer solchen Hierarchie von Relationen. In diesem Fall wird die allgemeine Relation „affected“ (beeinflusst) durch Informationen über spezielle Arten der Beeinflussung wie etwa Interaktion („interacts with“) oder Komplikation („complicates“) ergänzt. Hierdurch wird ausgedrückt, dass alle Objekte, die in der spezielleren Relation zueinander stehen, dies auch im Hinblick auf die allgemeinere Relation tun.

Insgesamt wird das Wissen im semantischen Netz durch die folgenden Elemente definiert, die sich gegenseitig ergänzen:

- Die Beschreibung von Semantic Types und Relationen durch Namen, ID, Beschreibung und, im Falle von Relationen, den Namen der entsprechenden inversen Relation:

```
RL|T148|prevents|R3.1.6|Stops, hinders or eliminates
an action or condition.|||PV|prevented_by|
```





**Abb. 3.7** Beispiel von Teilrelationen im semantischen Netz

- Informationen über die Struktur des Netzes in Form von Knoten und Kanten, wobei Kanten stets Relationen darstellen. Knoten können Konzepte, aber auch, wie das folgende Beispiel zeigt, Relationen sein:

```

Drug Delivery Device|isa|Medical Device|
prevents|isa|affects|
Medical Device|prevents|Pathologic Function

```

- Zusätzlich wird eine Version der Informationen angeboten, in der bereits alle Strukturen, die sich aus der Definition von inversen und Teilrelationen ergeben, bereits enthalten sind. Diese Version würde zum Beispiel auch die folgende Information enthalten, die sich aus den oben angegebenen Definitionen ergibt:

```

Medical Device|affects|Pathologic Function

```

### 3.3 Die Suggested Upper Merged Ontology

Die Suggested Upper Merged Ontology (SUMO) ist derzeit eine der größten Ontologien, deren Inhalt auf der Basis formaler Logik spezifiziert ist. Sie beinhaltet im Moment<sup>1</sup> etwa 20.000

---

<sup>1</sup> Stand Juni 2008

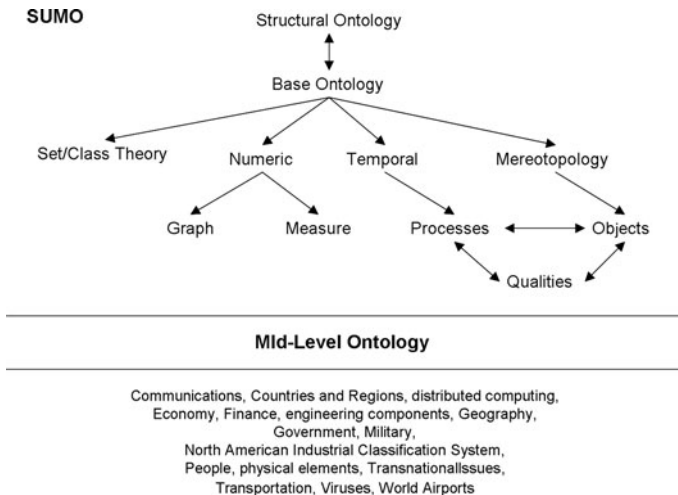


Begriffe, deren Bedeutung mithilfe von etwa 70.000 logischen Formeln beschrieben wird. Diese Definitionen unterteilen sich in eine Reihe von Teilmodellen, die jeweils bestimmte Themenbereiche abdecken. Die wichtigsten Teilmodelle sind zum einen die eigentliche Suggested Upper Merged Ontology, die abstrakte Begriffe definiert, welche zur exakteren Beschreibung konkreter Konzepte in den anderen Teilmodellen verwendet werden können, sowie die Mid-Level Ontology (MILO), welche die Verbindung zwischen SUMO und einer Reihe von Teilmodellen über konkrete Themen herstellt. Zurzeit gibt es Teilmodelle zu den folgenden Themenbereichen:

- Kommunikationstechnologie
- Länder und Regionen
- Verteiltes Rechnen
- Wirtschaft
- Finanzmärkte
- Maschinenbau
- Geographie
- Politik
- Militär
- Produktion
- Menschen
- Chemische Elemente
- Diplomatie
- Transportwesen
- Viren
- Flughäfen
- Terrorismus

Während SUMO und MILO einen gewissen Anspruch auf Vollständigkeit erheben, sind die übrigen Teilmodelle in der Regel das Ergebnis konkreter Projekte bzw. konkreter Anforderungen aus einer bestimmten Anwendung, so dass der Inhalt die-

ser Modelle das entsprechende Themengebiet oft aus einem bestimmten Blickwinkel betrachtet. Die verschiedenen Teilmodelle verwenden Definitionen aus SUMO und MILO und teilweise auch aus anderen Teilmodellen, so dass ein Gesamtmodell entsteht. Abbildung 3.8 illustriert das Verhältnis der verschiedenen Teilmodelle zueinander.



**Abb. 3.8** Architektur der SUMO-Ontologien

Neben den eigentlichen Ontologien bietet SUMO noch eine Reihe weiterer nützlicher Elemente, zum Beispiel die Übersetzung zentraler Begriffe in unterschiedliche Sprachen wie Hindi, Chinesisch, Deutsch und Tschechisch, eine Abbildung von WordNet Synsets (vgl. Kapitel 3.1) auf im SUMO-Teil spezifizierte Begriffe. Zusätzlich werden spezielle Software-Tools zur

Erstellung, Wartung und Verwendung der SUMO-Ontologien zur Verfügung gestellt.

### 3.3.1 *SUO-KIF und logische Definitionen*

Wie bereits erwähnt, werden Begriffe in SUMO durch logische Formeln definiert. Um die computerbasierte Verarbeitung der entsprechenden Definitionen zu erleichtern, verwendet SUMO eine spezielle formale Sprache, SUO-KIF, zur Darstellung logischer Formeln. SUO-KIF<sup>2</sup> ist in der Lage, prädikatenlogische Formeln mit Gleichheit darzustellen und deckt daher alle in [Kapitel 2.2](#) vorgestellten Konzepte ab. Zusätzlich verfügt SUO-KIF über ein spezielles Vokabular zur Darstellung typischer Ontologie-Elemente wie Vererbung und Klassenzugehörigkeit. Außerdem können in beschränktem Maße Aussagen über logische Formeln gemacht werden, um, ähnlich wie in [Kapitel 2.2](#) beschrieben, generelle Inferenzregeln darstellen zu können.

SUO-KIF verwendet eine Präfix-Notation zur Wiedergabe relationaler Ausdrücke, so wird zum Beispiel die Tatsache, dass Berlin die Hauptstadt von Deutschland ist, folgendermaßen dargestellt:

```
(capital-of Berlin Deutschland)
```

Als Grundlage für die Darstellung von Klassen und Objekten verwendet SUO-KIF die beiden vordefinierten Relationen `instance` und `subclass` sowie eine Reihe weiterer vordefinierter Relationen, wie etwa `subrelation`. So kann die Aussage, dass Berlin eine Instanz des Konzeptes Hauptstadt ist und dieses Konzept eine Unterklasse von Stadt im Allgemeinen und

---

<sup>2</sup> <http://suo.ieee.org/SUO/KIF/suo-kif.html>

capital-of eine Teilrelation von lies-in ist auf folgende Weise dargestellt werden:

```
(instance Berlin Capital)
(subclass Capital City)
(subrelation capital-of lies-in)
```

SUO-KIF erlaubt es nun, prädikatenlogische Formeln über relationale Ausdrücke zu definieren. Die logischen Operatoren sind hierbei die gleichen, die bereits in [Kapitel 2.2](#) vorgestellt wurden. Formeln (sentences) unterteilen sich hierbei in relationale Ausdrücke (relsentence), die den atomaren Formeln aus [Kapitel 2.2](#) entsprechen und wie oben beschrieben aufgebaut sind, sowie einfachen logischen (logsentence) und quantifizierten logischen Formeln (quantsentence). Sie sind hierbei in SUO-KIF wie folgt rekursiv definiert:

```
sentence ::= relsentence | logsentence | quantsentence

logsentence ::= (not sentence) |
                (and sentence+) |
                (or sentence+) |
                (=> sentence sentence) |
                (<=> sentence sentence)

quantsentence ::= (forall (variable+) sentence) |
                  (exists (variable+) sentence)
```

Hierbei werden Variablen durch Worte, die mit einem Fragezeichen beginnen, dargestellt. Außerdem können in SUO-KIF komplexe Terme als Argumente relationaler Ausdrücke verwendet werden. Diese werden ebenfalls in Präfix-Notation dargestellt und besitzen daher die gleiche Syntax wie einfache relationale Ausdrücke.

Unter Verwendung der beschriebenen Syntax enthält SUMO komplexe Definitionen für Begriffe. Dies lässt sich beispielhaft an dem Konzept HumanSlave zeigen. Dieses Konzept ist defi-

niert als die Menge aller Personen, die sich im Besitz einer anderen Person befinden:

```
(=>
  (instance ?SLAVE HumanSlave)
  (exists (?PERSON)
    (and
      (instance ?PERSON Human)
      (not
        (equal ?PERSON ?SLAVE))
      (possesses ?PERSON ?SLAVE))))
```

SUMO unterscheidet sich von vielen anderen formalen Ontologien dadurch, dass nicht nur Konzepte formal beschrieben werden, sondern die entsprechenden Modelle auch Formeln enthalten, welche die spezifischen Eigenschaften der verwendeten Relationen exakt definieren. So besitzt SUMO beispielsweise eine Reihe von Axiomen, welche die Natur der Besitzrelation weiter definieren. Zu diesen Axiomen gehört unter anderem das folgende, welches aussagt, dass jemand, der etwas besitzt, auch das Recht hat, dieses zu benutzen:

```
(=>
  (possesses ?PERSON ?OBJ)
  (modalAttribute
    (uses ?OBJ ?PERSON) Permission))
```

An diesem Beispiel lässt sich bereits sehen, dass die Ausdrucksmächtigkeit von SUO-KIF über die in Kapitel 2.2 vorgestellten Konzepte hinausgeht. Die Aussage, dass die Benutzung des Eigentums erlaubt ist, fällt nicht mehr in den Rahmen der Prädikatenlogik, da sie eine Aussage über eine atomare Formel darstellt. Zur automatischen Verarbeitung solcher Aussagen sind spezielle Beweisverfahren notwendig, deren Behandlung den Rahmen dieses Buches sprengen würde. Ein weiteres Beispiel für Definitionen, die nicht mehr im Rahmen der Prädikatenlogik liegen, sind Aussagen über den Wahrheitswert einer bestimmten Formel. In SUO-KIF können solche Aussagen mit

Hilfe der speziellen Relation `holds` gemacht werden. Die `holds` Relation bekommt einen Relationennamen sowie eine Reihe von Termen als Eingabe und legt fest, dass sich die Terme in der entsprechenden Relation befinden. Dies kann zum Beispiel für die Definition allgemeiner Ableitungsregeln wie der folgenden verwendet werden:

```
(=>
  (and
    (subrelation ?REL1 ?REL2)
    (holds ?REL1 ?ARG1 ?ARG2)
    (holds ?REL2 ?ARG1 ?ARG2))
```

Diese Regel besagt, dass jede zweistellige Relation dann automatisch zwischen zwei Objekten besteht, wenn diese sich in einer Teilrelation der entsprechenden Relation befinden.

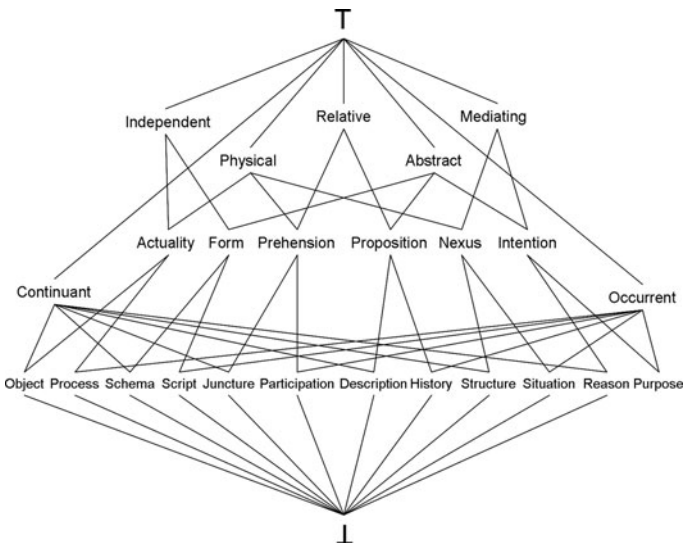
Durch die extrem hohe Ausdrucksmächtigkeit der Definitionen in SUMO sowie die Verwendung von Konzepten, die nicht mehr in der Prädikatenlogik liegen, ist die Ableitung von Wissen aus SUMO mit Hilfe logischen Schließens ein schwieriges Problem, welches noch nicht vollständig gelöst ist.

### 3.3.2 SUMO als Basis-Ontologie

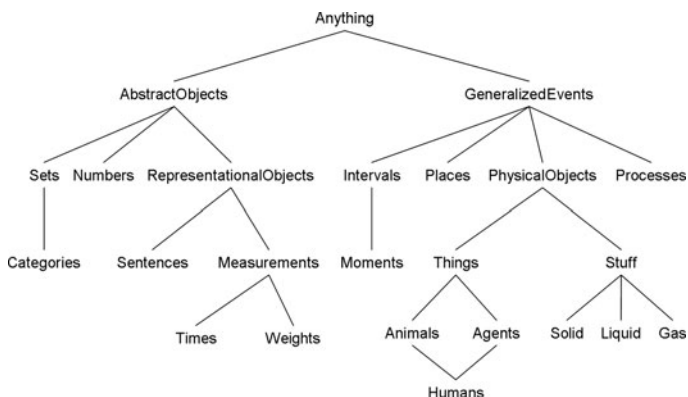
Obwohl SUMO, wie oben beschrieben, aus einer Reihe unterschiedlicher Ontologien besteht, bildet die eigentliche SUMO-Ontologie das Kernstück dieser Sammlung und ist deren wesentlicher Bestandteil. Dies liegt an der speziellen Natur der SUMO-Ontologie, die mit dem Ziel erstellt wurde, eine gemeinsame Basis für alle weiteren Entwicklungen spezifischer Ontologien zu bilden. Insbesondere werden drei grundlegende Funktionen der SUMO-Ontologie genannt:

- Grundlage für den Entwurf neuer Ontologien und Informationsquellen
- Wiederverwendung und Integration vorhandener Informationsquellen
- Integration bzw. Verknüpfung bestehender Ontologien

Speziell die Integration unterschiedlicher Ontologien und Informationsquellen setzt hierbei oft die Existenz eines gemeinsamen Vokabulars voraus. Um ein solches Vokabular zu schaffen, wurde die SUMO-Ontologie entwickelt. Die Grundlage hierfür bildeten eine Reihe bestehender Kategoriensysteme, die zunächst in SUO-KIF dargestellt und dann in ein gemeinsames Modell überführt wurden. Ausgangspunkt hierfür waren die in Abb. 3.9 und 3.10 dargestellten Kategoriensysteme.



**Abb. 3.9** Sowa's Top-Level Ontologie



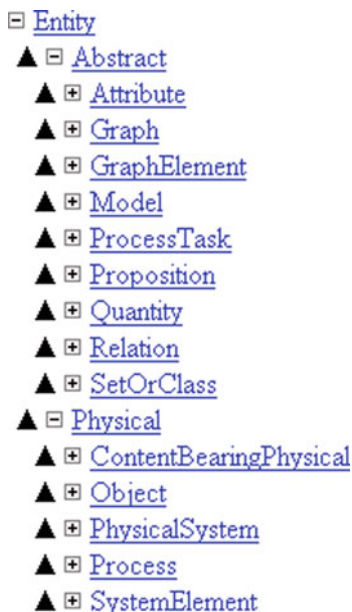
**Abb. 3.10** Die Upper-Level Ontologie von Russel und Norvig

Das Ergebnis der Integration dieser beiden Kategoriensysteme, die in der Tradition der in [Kapitel 1](#) diskutierten philosophischen Sicht auf Ontologie und Kategoriensysteme stehen, ist der sogenannte SUMO Top-Level, ein in SUO-KIF formalisiertes Kategoriensystem, in dem grundlegende Konzepte zur Beschreibung konkreter Ontologien festgehalten wurden. Es wurde in der Zwischenzeit wesentlich weiterentwickelt, um die Anforderungen konkreter Anwendungen erfüllen zu können. [Abbildung 3.11](#) zeigt den aktuellen Stand der SUMO Top-Level Konzepte.<sup>3</sup>

Wie [Abb. 3.8](#) andeutet, stehen hinter diesen recht abstrakten Konzepten jeweils ausgefeilte Theorien. Beispiele sind die Konzepte Relation, Graph sowie SetOrClass mit den jeweiligen mathematischen Theorien von Relationen, Graphen und Mengen, die im Rahmen von SUMO in SUO-KIF axiomatisiert wurden. Durch die Vererbung anwendungsspezifischer Konzepte von diesen bereits in SUMO enthaltenen Konzepten lassen sich die

<sup>3</sup> Stand: Juli 2008





**Abb. 3.11** Aktuelle Hierarchie der SUMO Top-Level Kategorien

entsprechenden Theorien direkt in Anwendungen benutzen, ohne dass entsprechende Eigenschaften mühsam definiert werden müssen. Ein gutes Beispiel hierfür sind die in SUMO enthaltenen Definitionen der mathematischen Eigenschaften binärer Relationen, wie etwa

- Reflexivität:

```

(<=>
(instance ?REL ReflexiveRelation)
(forall (?INST1 ?INST2)
(=>
  (?REL ?INST1 ?INST2)
  (?REL ?INST1 ?INST1))))

```

- Symmetrie:

```
(<=>
(instance ?REL SymmetricRelation)
(forall (?INST1 ?INST2)
  (=>
    (?REL ?INST1 ?INST2)
    (?REL ?INST2 ?INST1))))
```

- Transitivität:

```
(<=>
(instance ?REL TransitiveRelation)
(forall (?INST1 ?INST2 ?INST3)
  (=>
    (and
      (?REL ?INST1 ?INST2)
      (?REL ?INST2 ?INST3))
    (?REL ?INST1 ?INST3))))
```

Diese grundlegenden Definitionen werden nun in den konkreten Ontologien, aber auch in SUMO selbst verwendet, indem andere konkrete Relationen als Spezialfälle oder auch Instanzen dieser Relationen definiert werden. So ist das Konzept *EquivalenceRelation* als Unterklasse dieser drei Konzepte definiert, so dass die Äquivalenz sowohl die Reflexivitäts- als auch die Symmetrie- und die Transitivitätseigenschaft erbt. Es finden sich auch eine Reihe von Beispielen für konkrete Relationen, die direkt auf der Basis dieser abstrakten Klassen definiert werden. Beispiele sind

**reflexive Relationen:** *connected*, *overlaps* und *subGraph*;

**symmetrische Relationen:** *enemy*, *friend*, *coworker*, *consistent*, ...

**transitive Relationen:** `crosses,` `flows,`  
`dependentGeopoliticalArea,`  
`multiplicativeFactor, ...`

Die formalen Eigenschaften der Relationen können nun verwendet werden, um implizites Wissen in konkrete Ontologien abzuleiten. Hierdurch schafft SUMO einen Mehrwert für Ontologien in konkreten Anwendungen, den man ansonsten nicht mit vertretbarem Aufwand erreichen könnte.

### 3.3.3 *Verbindung zu WordNet*

Einen weiteren Mehrwert erzeugt SUMO durch eine bestehende Verbindung zum oben beschriebenen semantischen Netz WordNet, wodurch nicht nur formale Informationen aus den logischen Definitionen, sondern darüber hinaus auch linguistische Informationen zu bestimmten Begriffen zur Verfügung stehen. Hierbei sind vor allem die Informationen über Synonyme von Bedeutung, da die formalen Definitionen in SUMO zwar Ober- und Unterklassenrelationen sowie Informationen über Teil-Ganzes-Beziehungen und zu Gegensätzen enthalten, es jedoch an Informationen über unterschiedliche mögliche Bezeichnungen des gleichen Konzeptes fehlt. Die entsprechende Abbildung beschränkte sich dabei zunächst auf Abbildungen zwischen Substantiven in WordNet und entsprechenden Konzepten in SUMO. Inzwischen ist die Abbildung jedoch auch auf andere Wortformen ausgeweitet worden, so dass eine relativ vollständige Verbindung existiert. Hierbei werden drei Arten von Beziehungen zwischen WordNet Synsets und SUMO-Konzepten betrachtet:

**Synonymie:** Der in WordNet beschriebene Begriff entspricht exakt einem SUMO-Konzept. Zum Beispiel entspricht die

Synonymmenge  $\{plant, flora, plant\_life\}$  dem SUMO Konzept `Plant`.

**Hyponomie:** Der in WordNet beschriebene Begriff ist ein Unterbegriff eines SUMO-Konzeptes. Zum Beispiel ist der WordNet Begriff  $\{Christian\_Science\}$  ein Spezialfall des SUMO Konzeptes `ReligiousOrganization`.

**Instanz:** Der in WordNet beschriebene Begriff ist eine Instanz eines SUMO-Konzeptes. Zum Beispiel ist der WordNet Synset  $\{Underground\_Railroad, Underground\_Railway\}$ , der in diesem Fall eine geheime Hilfsorganisation für entflohene Sklaven vor dem amerikanischen Bürgerkrieg beschreibt, eine Instanz des SUMO Konzeptes `Organisation`.

Hyponomie ist als Beziehung zwischen WordNet und SUMO nicht vorgesehen, da WordNet wesentlich umfangreicher und detaillierter ist als SUMO und daher immer auf abstraktere SUMO-Konzepte verwiesen wird. Technisch umgesetzt ist die Abbildung durch eine Erweiterung der WordNet-Definitionen. Hierfür wurden WordNet-Definitionen um spezielle Kommentare erweitert. Dabei wird der Name des entsprechenden SUMO-Konzeptes mit dem Präfix `&%` versehen und an die Beschreibung des Begriffes angehängt. Zusätzlich wird die Art der Relation durch das Postfix `=` (Synonymie), `+` (Hyponomie) oder `@` (Instanz) erläutert. Die oben genannten Beispiele werden demnach folgendermaßen beschrieben:

- `plant, flora, plant life ((botany) a living organism lacking the power of locomotion) &%Plant=`
- `Christian Science (religious system based on teachings of Mary Baker Eddy emphasizing spiritual healing) &%ReligiousOrganization+`
- `Underground Railroad, Underground Railway (secret aid to escaping slaves that was provided by abolitionists in the years before the American Civil War) &%Organization@`

Diese Verbindung zu WordNet erleichtert es SUMO vor allem, den oben beschriebenen Zweck im Hinblick auf die Entwicklung und Integration von Ontologien und Informationen zu erfüllen. Insbesondere bildet WordNet einen idealen Einstiegspunkt in das formale SUMO-Modell, da sich Begriffe, die in Informationen oder anderen Ontologien auftreten, sich in der Regel in WordNet finden lassen. Durch die beschriebene Annotation von WordNet mit SUMO-Konzepten ergibt sich in diesem Fall die semantische Beziehung zu SUMO-Konzepten, so dass SUMO als gemeinsames Begriffssystem für die Beschreibung und Integration von Daten fungieren kann.

## **Teil II**

# **Technologien**

Während die explizite Beschreibung von Ontologien im Bereich der formalen Ontologie hauptsächlich dazu dient, ein Verständnis für Prinzipien der Kategorisierung von Objekten zu gewinnen und somit eher Mittel zum Zweck als das eigentliche Ziel der Arbeit darstellt, sind in der Informatik die Produkte einer solchen Formalisierung, die erstellten Ontologien, selbst das Ziel. Dementsprechend kommt der Erstellung und Verwaltung von Ontologien als komplexe Informationsobjekte in der Informatik eine Bedeutung zu, die im Bereich der klassischen Ontologie unbekannt ist. Zwar werden die entsprechenden Technologien, die inzwischen entwickelt wurden, um die Erstellung von Ontologien zu unterstützen, auch zunehmend im Bereich der klassischen Ontologie als nützliche Werkzeuge eingesetzt, ein echter Bedarf an Technologien, die sich mit Ontologien beschäftigen, wurde erst durch ihre Verwendung in der Informatik geschaffen. Dieser Teil des Buches widmet sich Technologien, welche die Erstellung und Verwendung von Ontologien unterstützen. Kapitel 4 widmet sich zunächst Formalismen für die Darstellung und Speicherung von Ontologien. Der Fokus liegt hierbei nicht so sehr auf der physikalischen Speicherung. Es werden vielmehr verschiedene Klassen von Darstellungsformen für Ontologien diskutiert, die jeweils unterschiedliche Arten von Operationen auf der Ontologie unterstützen. Die Betrachtung orientiert sich hierbei an einer Logik-basierten Repräsentation und diskutiert neben der Ausdrucksmächtigkeit unterschiedlicher Formalismen vor allem auch Möglichkeiten des logischen Schließens über Ontologie-Modellen. Ontologien, die den lexikalischen Ansatz zur Darstellung von Semantik verfolgen, werden hierbei ebenfalls erfasst, da sich semantische Relationen, wie sie in der Linguistik verwendet werden, durch nicht-logische Symbole darstellen und sich entsprechende Inferenzmechanismen durch spezielle Ableitungsregeln implementieren lassen. Kapitel 5 widmet sich Methoden und

Werkzeugen der Ontologie-Erstellung. Im Gegensatz zum vorangegangenen Kapitel steht hier nicht so sehr das Produkt als solches im Vordergrund, sondern Vorgehensweisen und Methoden, die Anwender bei der Erstellung oder Wartung einer Ontologie unterstützen. Wir betrachten hierbei zunächst allgemeine Vorgehensweisen, die sich bei der Erstellung von Ontologien bewährt haben. Anschließend werden Werkzeuge zur Erstellung und Wartung von Ontologien beschrieben. Dabei werden am Beispiel des Ontologie-Editors Protege typische Funktionalitäten solcher Werkzeuge, die zum Teil auf den verwendeten Formalismen basieren, vorgestellt. Am Ende des Kapitels werden Methoden zur automatischen Erzeugung von Ontologien aus Texten beschrieben. Hierdurch schließen wir den Kreis und kehren zur semantischen Analyse in der Linguistik zurück, die einer der Ausgangspunkte unserer Betrachtung im ersten Teil des Buches war und die im Rahmen der automatischen Ontologie-Erstellung eine wesentliche Rolle spielt.



## Kapitel 4

# Ontologiesprachen

Die wahrscheinlich bedeutendste Entwicklung in Bezug auf Ontologien und deren Popularität ist weniger ein grundlegender technologischer oder theoretischer Fortschritt im Bereich der Formalisierung von Wissen, sondern eher eine organisatorische Veränderung. Ontologien, wie die im ersten Teil dieses Buches gezeigt in der Tradition der philosophischen Ontologie, wurden in der Regel als normative Modelle einer bestimmten Domäne angesehen. Die Tatsache, dass es unterschiedliche Ontologien gibt – im Bereich allgemeiner Ontologien sind dies neben SUMO-Ontologien DOLCE oder Cyc – wurde eher als Unfall betrachtet, und es wurden hitzige Diskussionen darüber geführt, welche Ontologie die Realität am korrektesten widerspiegelt.

Inzwischen hat sich eine pragmatischere Sicht durchgesetzt, nach der die Ontologien eher in der Tradition konzeptueller Datenmodelle stehen. Hiernach ist die Funktion einer Ontologie weniger die korrekte Beschreibung der Realität als vielmehr die Unterstützung einer korrekten Interpretation eines gegebenen Datenbestandes. Diese Sicht, die ja bereits am Anfang dieses

Buches motiviert wurde, bringt eine Reihe von Konsequenzen mit sich, die sich nicht nur auf die Verwendung von Ontologien, sondern vor allem auch auf die Verfügbarkeit und Funktion von Technologien zur Erstellung und Verarbeitung von Ontologien auswirken.

Eine erste Konsequenz der Verwendung von Ontologien als Datenmodell ist die Notwendigkeit standardisierter Modellierungssprachen. Solange Ontologien als normativ für eine bestimmte Domäne angesehen werden, besteht keine echte Notwendigkeit einer Standardisierung der Darstellung, da die bestehende Darstellung der Ontologie den De-facto-Standard für diese Domäne setzt. Erst durch die Ko-Existenz unterschiedlicher Modelle entsteht ein Bedarf nach Standardisierung der Repräsentationssprache, die den Austausch und die einheitliche Verarbeitung unterschiedlicher Modelle ermöglicht.

Ein weiterer wesentlicher Unterschied der Sicht auf Ontologien als Modelle zur Beschreibung von Daten ist die Veränderung in der Forschung. In der traditionellen Sicht von Ontologien als normatives Modell kann davon ausgegangen werden, dass nur Experten mit der Erstellung von Ontologien befasst sind. In der Tat werden große und wichtige Ontologien, wie die im ersten Teil genannten, auch heute noch von Teams gebaut, die sich aus Experten des Anwendungsgebietes sowie Experten im Bereich der Modellierung zusammensetzen. Von ihnen kann angenommen werden, dass sie ein ausreichendes Verständnis sowohl im Hinblick auf die Allgemeingültigkeit als auch auf die korrekte Form der Repräsentation der entsprechenden Fakten besitzen. Die zunehmende Verwendung von Ontologien als lokale Datenmodelle führt dazu, dass mehr und mehr auch Personen, die sich zwar beruflich mit einem bestimmten Anwendungsgebiet und den entsprechenden Daten beschäftigen, jedoch eher über Praxiswissen als über ein umfassendes Verständnis des gesamten Anwendungsgebietes verfügen und zudem wenig oder keine

Erfahrung mit formaler Wissensrepräsentation haben, mit der Erstellung und Verwendung von Ontologien beschäftigt sind.

Aus technologischer Sicht bedeutet dies, dass zum einen die Modellierung und die Interaktion mit Ontologien so intuitiv wie möglich gestaltet werden müssen, um den genannten Personengruppen den Zugang zu dieser Technologie zu erleichtern. Zum anderen besteht ein besonderer Bedarf an Technologien zur Unterstützung der Anwender durch spezielle Werkzeuge und fortgeschrittene Inferenzmethoden, die zum Beispiel in der Lage sind, Modellierungsfehler aufzudecken.

Dieses Kapitel widmet sich grundlegenden Technologien, welche den Einsatz von Ontologien als intelligente Datenmodelle unterstützen. Der Schwerpunkt der Betrachtung liegt hierbei zum einen auf formalen Ontologiesprachen, zum anderen auf Werkzeugen zur Verarbeitung formaler Ontologiemodelle im Kontext der Datenmodellierung. Im Hinblick auf Ontologiesprachen soll hierbei auf der Grundlage der im ersten Teil vorgestellten Verwendung von Prädikatenlogik erster Stufe als Mittel der Wissensrepräsentation die Verbindung zu aktuellen Ontologiesprachen hergestellt werden, die im Kontext des Semantic Web verwendet werden. Der Abschnitt zu Ontologie-Werkzeugen befasst sich vor allem mit Systemen zum logischen Schließen über Ontologien sowie mit der Beantwortung von Anfragen an Daten mit Hilfe von Ontologien.

Die oben genannten Anforderungen an die Unterstützung der Benutzer von Ontologiesprachen hat dazu geführt, dass sich im Bereich der Verwendung von Ontologien als Datenmodelle Logik-basierte Ansätze zur Spezifikation durchgesetzt haben. Grund hierfür sind vor allem die Eindeutigkeit der entsprechenden Spezifikationen, die den Austausch von Modellen erleichtert, sowie die Möglichkeit, den Benutzer durch logisches Schließen zu unterstützen. In der Praxis ist jedoch die direkte Verwendung von Prädikatenlogik kaum sinnvoll. Dies

liegt an der Komplexität der Prädikatenlogik sowohl im technischen Sinn als auch im Hinblick auf eine Verwendung als Modellierungssprache. Als Reaktion auf diese Komplexität beschränken sich bestehende Ontologiesprachen in der Regel (die im ersten Teil im Zusammenhang mit der SUMO-Ontologie vorgestellte Sprache SUO-KIF bildet hier eine Ausnahme) auf eine Teilmenge der Prädikatenlogik. Die Wahl dieser Teilmenge ist hierbei teils durch formale Eigenschaften der entsprechenden Teilsprachen, teils durch Modellierungsaspekte motiviert. Dies zeigt sich besonders deutlich bei der Web Ontology Language OWL und ihren Teilsprachen. OWL-DL bezeichnet zum Beispiel eine Teilsprache von OWL, die einer Teilmenge der Prädikatenlogik mit vorteilhaften theoretischen Eigenschaften entspricht. Die Sprache OWL-lite hingegen ist eine Teilmenge von OWL-DL, deren Einschränkungen weitestgehend in der einfacheren Modellierbarkeit begründet ist. Auf theoretischer Ebene entspricht OWL-lite einer nur unwesentlich schwächeren Teilmenge der Prädikatenlogik als OWL-DL.

Im Folgenden orientieren wir uns an speziellen Unterformen der Prädikatenlogik, die sich als nützlich zur Darstellung von Ontologien herausgestellt haben und in aktuellen Sprachen Verwendung finden. Diese Unterformen sind zum einen der Bereich der Logik-Programmierung, welcher auf der Verwendung spezieller Formeln in Regelform, der eingeschränkten Verwendung von Quantoren sowie der bereits erwähnten Closed-World Annahme basiert. Zum anderen werden wir uns mit Beschreibungslogiken beschäftigen, welche die Verwendung von Variablen in Formeln einschränken und spezielle Konstrukte für die Beschreibung von Klassen und Relationen verwenden. Wir zeigen, dass bekannte Ontologiesprachen wie RDF Schema und F-Logic auf der Verwendung von Logik-Programmierung basiert und stellen den Zusammenhang zwischen Beschreibungslogiken und der Web Ontology Language dar. Am Ende des Kapitels

wird kurz das Verhältnis zwischen den beiden behandelten Paradigmen diskutiert und bestehende Ansätze zu deren Integration vorgestellt.

## 4.1 Logik-Programme

Logik-Programmierung ist ein Programmierparadigma, in dem Programme durch einfache logische Formeln, welche die Form von Regeln haben (siehe unten), angegeben werden. Die Auswertung dieser Programme erfolgt mit Hilfe spezieller Interpreter, die Regeln nach einem fest vorgegebenen Schema auswerten, um das Ergebnis des Programms zu bestimmen. Im Allgemeinen haben Regeln, aus denen Logik-Programme bestehen, die folgende Form:

$$B \rightarrow H$$

Hierbei wird  $B$  als der Regelkörper (Body) und  $H$  als der Regelkopf (Head) bezeichnet. Eine Menge solcher Regeln ist das Programm. Zusätzlich werden Regeln mit leerem Körper als Spezialfall betrachtet. Solche Regeln ( $\rightarrow H$ ) bezeichnen Fakten, da keine Voraussetzungen für die Wahrheit von  $H$  bestehen.

Unterschiedliche Ansätze der Logik-Programmierung unterscheiden sich in der Form, die  $B$  und  $H$  annehmen können. Die üblichste Variante erlaubt für  $B$  und  $H$  eine Konjunktion atomarer Formeln:

$$B_1 \wedge \cdots \wedge B_k \rightarrow H_1 \wedge \cdots \wedge H_n \quad (4.1)$$

Hierbei werden Regeln mit einer Konjunktion im Regelkopf meist in eine Menge von Regeln mit gleichem Regelkörper und nur einer atomaren Formel im Kopf unterteilt. Entsprechende Regeln haben daher die folgende Form:

$$\begin{aligned}
& B_1 \wedge \cdots \wedge B_k \rightarrow H_1 \\
& \quad \dots \\
& B_1 \wedge \cdots \wedge B_k \rightarrow H_n
\end{aligned} \tag{4.2}$$

Disjunktionen können hierbei implizit durch die Angabe unterschiedlicher Regeln mit gleichem Regelkopf dargestellt werden. So sind die beiden folgenden Programme logisch äquivalent:

$$B_1 \vee \cdots \vee B_k \rightarrow H \tag{4.3}$$

$$B_1 \rightarrow H, \dots, B_k \rightarrow H \tag{4.4}$$

Eine weitere Unterscheidung zwischen den verschiedenen Ansätzen der Logikprogrammierung ergibt sich durch Einschränkungen auf atomare Formeln und insbesondere durch die Verwendung von Funktionssymbolen in Termen atomarer Formeln. Hierin unterscheiden sich die beiden geläufigsten Varianten der Logik-Programmierung. PROLOG (Programming in Logic), die bekannteste auf dem Prinzip der Logik-Programmierung basierende Programmiersprache zum Beispiel, erlaubt explizit Funktionssymbole zur Definition komplexer Terme in atomaren Formeln. Datalog, eine Variante der Logik-Programmierung, die vor allem als Datenmodellierungs- und Anfragesprache im Datenbankbereich zum Einsatz kommt, erlaubt nur Variablen (in Anfragen) und Konstanten in atomaren Formeln und schränkt damit die Ausdrucksmächtigkeit der entsprechenden Programme stark ein.

Mit Hilfe dieser eingeschränkten Form der Prädikatenlogik lassen sich jedoch bereits eine Reihe relevanter Modelle ausdrücken, wie wir schon in [Kapitel 2.2](#) gesehen haben. So fallen, wie sich leicht nachprüfen lässt, viele der Beispiele aus [Kapitel 2.2](#) in diese Teilmenge der Prädikatenlogik. Wir wollen uns

dies nun anhand der Beispiele aus [Kapitel 2.2](#) ansehen. Die Definitionen in [Ausdruck 2.3](#) entsprechen direkt der regelbasierten Darstellung, die in der Logik-Programmierung verwendet wird. Komplizierter wird die Darstellung der Einschränkungen auf den charakteristischen Eigenschaften in [Ausdruck 2.4](#) und [2.5](#). Hier wird klar, dass sich Definitionen, wie die in [Ausdruck 2.5](#), ebenfalls im Rahmen eines Logikprogramms wiedergeben lassen. Die folgende Regel entspricht dem Beispiel aus [Ausdruck 2.5](#).

$$lives - in(x, y) \wedge fish(x) \rightarrow water(y) \quad (4.5)$$

Die in [Ausdruck 2.4](#) gezeigte Definition hingegen lässt sich nicht darstellen, da keine Möglichkeit besteht, den Existenzquantor im Ausdruck korrekt in eine Regel zu übersetzen. Die Unfähigkeit, diese Definition darzustellen ist ein Effekt der Beschränkung der Sprache auf eine Teilmenge, die deutlich einfacher zu verstehen und zu verarbeiten ist als vollständige Prädikatenlogik. Die in diesem Kapitel vorgestellten Ontologiesprachen schränken die Prädikatenlogik auf unterschiedliche Arten ein. So kann die Definition [2.4](#) zwar leicht in Beschreibungslogiken und somit in der Web Ontology Language dargestellt werden. Dafür ist es in dieser Sprache aber nicht ohne Weiteres möglich, komplexe Zusammenhänge zwischen verschiedenen Relationen darzustellen, was wiederum auf der Basis von Logik-Programmen problemlos möglich ist. Die Wahl einer geeigneten Sprache stellt daher eine wichtige Entscheidung bei der Erstellung und Verwendung von Ontologien dar.

Eine besondere Rolle kommt in der Logik-Programmierung der Negation zu, da diese oft anders interpretiert wird als in der klassischen Prädikatenlogik. Der Unterschied entspricht hierbei der bereits in [Kapitel 2.2](#) diskutierten Unterscheidung zwischen der Closed World und der Open World Annahme. In der klassischen Logik gilt die Formel  $\neg B$  nur dann als wahr, wenn bewiesen werden kann, dass  $B$  falsch ist (vgl. [Kapitel 2.2](#)), was

der Open World Annahme entspricht. Im Bereich der Logik-Programmierung wird dagegen in der Regel das Prinzip des „negation as failure“ verwendet. Dieses besagt, dass die Formel  $\neg B$  bereits dann als wahr gilt, wenn  $B$  nicht aus dem Programm abgeleitet werden kann. Diese Variante entspricht der Closed World Annahme, bei der davon ausgegangen wird, dass alle relevanten Fakten bekannt sind und daher ein Fakt, der nicht bekannt ist, als falsch angenommen werden kann. So ließe sich aus einem Logik-Programm, welches aus den Definitionen 2.1 und 2.2 besteht, unter anderem die Aussage  $\neg has(fish, vertebra)$  ableiten.

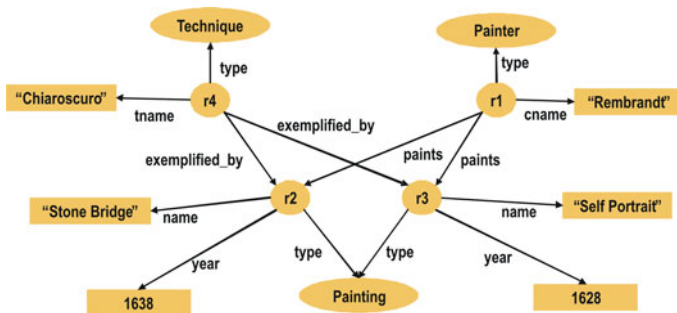
Das Paradigma der Logik-Programmierung lässt sich demnach zur Repräsentation von ontologischem Wissen vor allem dann verwenden, wenn bei der Beschreibung besonders komplexe Relationen zwischen Konzepten von Bedeutung sind. Eher weniger geeignet ist die Logik-Programmierung für solche Fälle, in denen der Schwerpunkt der Beschreibung auf notwendigen und hinreichenden Bedingungen für die Zugehörigkeit von Objekten zu bestimmten Klassen liegt, da sich solche Bedingungen nicht immer adäquat abbilden lassen. Sprachen, die aktuell im Kontext von Semantic Web Technologien häufig genutzt werden, vor allem diejenigen, die eine mehr oder weniger direkte Beziehung zur Logik-Programmierung besitzen. Dies sind zum einen RDF-Schema und zum anderen F-Logic. Im Folgenden werden diese beiden Sprachen und ihre Verbindung zu den oben beschriebenen Konzepten der Logik-Programmierung vorgestellt.

### ***4.1.1 RDF-Schema als Regelsystem***

Das Resource Description Framework (RDF) ist ein Datenmodell zur Repräsentation von Wissen im Internet. Zentrale Idee ist hierbei, binäre Relationen zwischen eindeutig bezeichneten



Ressourcen zu beschreiben. Diese können hierbei Objekte, Konzepte, Relationen oder konkrete Werte bezeichnen. Hiermit steht RDF in der Tradition semantischer Netze, wie sie in [Kapitel 2.1](#) beschrieben wurden. Grundlegende Datenstruktur von RDF sind binäre Relationen, die als Tripel der Form (Subjekt, Prädikat, Objekt) dargestellt werden, denen Kanten in einem gerichteten Graphen entsprechen. Hierbei beschreibt „Prädikat“ die Relation zwischen den durch Subjekt und Objekt bezeichneten Ressourcen. Abbildung 4.1 zeigt das Beispiel eines einfachen RDF-Modells als Graph.



**Abb. 4.1** Einfaches RDF-Modell über Rembrandt und seine Werke

Der Graph beschreibt den Maler Rembrandt, dargestellt durch die Ressource r1, zwei seiner Werke, dargestellt durch die Ressourcen r2 und r3, sowie eine bestimmte Maltechnik, die bei der Erstellung dieser Werke angewendet wurde und im Graph als Ressource r4 auftritt. Diese Ressourcen werden über ihren Typ, durch konkrete Werte für bestimmte Eigenschaften, insbesondere die Namen von Künstlern, Werken und Technik, sowie über ihre verschiedenen Relationen beschrieben. Wie bereits angedeutet, entspricht jede Kante in Abb. 4.1 einem Triple im

Modell. Wir können diese Tripel mit Hilfe eines speziellen Prädikates „triple“ als atomare Formeln in Prädikatenlogik wiedergeben. Der dargestellte Graph entspricht somit der folgenden Formel:

$$\begin{aligned}
 & triple(r1, rdf : type, Painter) \wedge \\
 & triple(r2, rdf : type, Painting) \wedge \\
 & triple(r3, rdf : type, Painting) \wedge \\
 & triple(r4, rdf : type, Technique) \wedge \\
 & triple(r1, paints, r2) \wedge \\
 & triple(r1, paints, r3) \wedge \\
 & triple(r4, exemplified - by, r2) \wedge \\
 & triple(r4, exemplified - by, r3) \wedge \\
 & triple(r1, cname, 'Rembrandt') \wedge \\
 & triple(r2, name, 'Self - Portrait') \wedge \\
 & triple(r3, name, 'StoneBridge') \wedge \\
 & triple(r2, year, 1628) \wedge \\
 & triple(r3, year, 1638) \wedge \\
 & triple(r4, tname, 'Chiaroscuro') \quad (4.6)
 \end{aligned}$$

Die Konstanten in dieser Formel stehen hierbei für Objekte, Werte und Konzepte. Zusätzlich definiert RDF Konstanten, die eine besondere Bedeutung für das Datenmodell besitzen. Diese speziellen Konstanten werden durch das Präfix „rdf:“ identifiziert. Im gezeigten Modell ist beispielsweise „rdf:type“ eine Konstante, die eine bestimmte Bedeutung besitzt. Sie beschreibt eine spezielle Relation zwischen Konstanten, nämlich die Mitgliedschaft eines Objektes in einem Konzept, und entspricht somit der im Kontext semantischer Netze als „isa“ bezeichneten Relation.

Des Weiteren unterscheidet RDF zwischen unterschiedlichen Arten von Werten, die in einem Tripel auftreten können. Insbesondere werden zwei Spezialfälle von Ressourcen unterschieden: Literale sind konkrete Werte von Eigenschaften. In unserem Modell sind Literale in Anführungszeichen dargestellt. Die übrigen Konstanten in obiger Formel sind Ressourcen. Literale dürfen in Tripeln lediglich als Objekt auftreten. Properties bilden eine spezielle Form von Ressourcen, die stets an der Position des Prädikats in einem Tripel auftreten. Genauer gesagt macht das Auftreten einer Konstante an zweiter Stelle eines „triple“-Prädikats automatisch zu einer Property. Diese Tatsache lässt sich in Form eines einfachen Logik-Programms darstellen.

$$\begin{aligned}
 \text{triple}(S, P, O) &\rightarrow \text{triple}(S, \text{rdf:type}, \text{rdf:Ressource}) \\
 \text{triple}(S, P, O) &\rightarrow \text{triple}(P, \text{rdf:type}, \text{rdf:Property}) \\
 \text{triple}(S, P, O) &\rightarrow \text{triple}(O, \text{rdf:type}, \text{rdf:Ressource})
 \end{aligned}
 \tag{4.7}$$

Wie das Programm zeigt, wird das Konzept aller Ressourcen sowie die Unterkonzepte Literal und Property ebenfalls durch spezielle Konstanten („rdf:Ressource“, „rdf:Literal“ und „rdf:Property“) beschrieben.

Dieses einfache Beispiel zeigt bereits sehr schön das grundlegende Prinzip, welches in RDF verwendet wird, um ontologisches Wissen zu beschreiben: die Einführung eines speziellen Vokabulars, das mit allgemeinen Ableitungsregeln in Form einfacher Logik-Programme verbunden ist. Sie definieren die intendierte Bedeutung dieses Vokabulars und erlauben es, sich aus dieser Bedeutung ergebendes implizites Wissen abzuleiten. Dieses Vokabular besteht vor allem aus speziellen Konstanten, die Relationen zwischen Ressourcen beschreiben. Von besonderer Bedeutung sind hierbei die folgenden Elemente:

**rdf:type** Wie bereits beschrieben, bezeichnet `rdf:type` die Beziehung zwischen Objekten und Konzepten, zu denen diese Objekte gehören. `rdf:type` spielt somit in RDF die gleiche Rolle, wie die *is-a* Beziehung in semantischen Netzen. So legt die Formel `triple(r1, rdf:type, Painter)` fest, dass das als `r1` bezeichnete Objekt zu der Kategorie „Painter“ gehört.

**rdfs:SubClassOf** Die Beziehung `rdfs:SubClassOf` bezeichnet die Vererbungsbeziehung zwischen Konzepten und ihren Oberkonzepten und entspricht somit der bereits beschriebenen *kind-of* Beziehung in semantischen Netzen. So beschreibt die Formel `triple(Painter, rdf:SubPropertyOf, Artist)` die Tatsache, dass das Konzept „Painter“ ein Unterkonzept des allgemeineren Konzeptes „Artist“ ist.

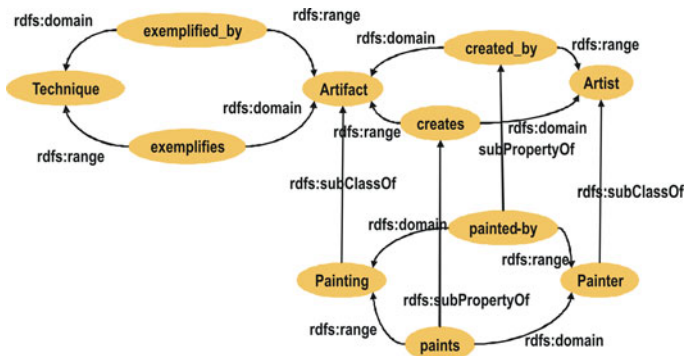
**rdfs:SubPropertyOf** Die Beziehung `rdfs:subPropertyOf` beschreibt die Teilrelation-Beziehung zwischen zwei Relationen und ermöglicht es so, Relationen weiter zu spezialisieren. Die Formel `triple(paints, rdfs:SubPropertyOf, creates)` definiert die Relation „paints“ als Spezialfall der allgemeineren Relation „creates“.

**rdfs:domain** Die Relation `rdfs:domain` beschreibt den Typ von Ressourcen, die in einer bestimmten Relation auftreten. `rdfs:domain` beschreibt hierbei den Typ der Ressource, die als Subjekt in dem entsprechenden Tripel auftritt. So lässt sich mit Hilfe der Formel `triple(paints, rdfs:domain, Painter)` festlegen, dass jemand, der etwas malt, automatisch zum Konzept der Maler gehört.

**rdfs:range** Die Relation stellt das Gegenstück zu „`rdfs:domain`“ dar und beschreibt den Typ der Ressource, die als Objekt in einem Tripel mit der entsprechenden Relation auftritt. So lässt sich mit Hilfe der Formel `triple(paints,`

`rdfs:range, Painting`) zum Beispiel festlegen, dass Dinge, die gemalt werden, automatisch vom Typ Gemälde sind.

Mit Hilfe dieses Vokabulars lassen sich einfache Ontologien darstellen, die Zusammenhänge in einer Anwendungsdomäne beschreiben. Im Rahmen unseres Beispiels könnte eine entsprechende Ontologie aussehen, wie in Abb. 4.2 gezeigt.



**Abb. 4.2** Einfache Ontologie über Künstler und ihre Werke

Die gezeigte Ontologie beschreibt im Wesentlichen die Beziehungen zwischen den in unserem Beispiel verwendeten Relationen und Klassen. So wird festgelegt, dass die Relation „`exemplifies`“ immer Ressourcen vom Typ `Technique` und `Artifact` verbindet, die Relation „`exemplified-by`“ Ressourcen vom Typ `Artifact` und `Technique`. Die Tatsache, dass „`exemplified-by`“ die inverse Relation zu „`exemplifies`“ darstellt, kann mit Hilfe des speziellen RDF-Vokabulars jedoch nicht ausgedrückt werden. Analog werden die Typen von Ressourcen durch die Relationen „`creates`“ und „`created-by`“ sowie „`paints`“ und

„painted-by“ festgelegt. Zusätzlich enthält die Ontologie die Information, dass Painter ein Unterkonzept von Artist und Painting ein Unterkonzept von Artifact ist. Bei der Beschreibung handelt es sich ebenfalls um ein RDF-Modell, also einen gerichteten Graphen, der lediglich das spezielle RDF-Vokabular verwendet. Dementsprechend lässt sich diese Beschreibung ebenfalls wie folgt als logische Formel darstellen:

$$\begin{aligned}
& \text{triple}(\text{exemplified} - \text{by}, \text{rdf} : \text{domain}, \text{Technique}) \wedge \\
& \text{triple}(\text{exemplified} - \text{by}, \text{rdf} : \text{range}, \text{Artifact}) \wedge \\
& \text{triple}(\text{exemplifies}, \text{rdf} : \text{domain}, \text{Artifact}) \wedge \\
& \text{triple}(\text{exemplifies}, \text{rdf} : \text{range}, \text{Technique}) \wedge \\
& \text{triple}(\text{creates}, \text{rdf} : \text{domain}, \text{Artist}) \wedge \\
& \text{triple}(\text{creates}, \text{rdf} : \text{range}, \text{Artifact}) \wedge \\
& \text{triple}(\text{created} - \text{by}, \text{rdf} : \text{domain}, \text{Artifact}) \wedge \\
& \text{triple}(\text{created} - \text{by}, \text{rdf} : \text{range}, \text{Artist}) \wedge \\
& \text{triple}(\text{paints}, \text{rdf} : \text{domain}, \text{Painter}) \wedge \\
& \text{triple}(\text{paints}, \text{rdf} : \text{range}, \text{Painting}) \wedge \\
& \text{triple}(\text{painted} - \text{by}, \text{rdf} : \text{domain}, \text{Painting}) \wedge \\
& \text{triple}(\text{painted} - \text{by}, \text{rdf} : \text{range}, \text{Painter}) \wedge \\
& \text{triple}(\text{Painting}, \text{rdfs} : \text{subClassOf}, \text{Artifact}) \wedge \\
& \text{triple}(\text{Painter}, \text{rdfs} : \text{subClassOf}, \text{Artist}) \wedge \\
& \text{triple}(\text{paints}, \text{rdfs} : \text{subPropertyOf}, \text{creates}) \wedge \\
& \text{triple}(\text{painted} - \text{by}, \text{rdfs} : \text{subPropertyOf}, \text{created} - \text{by})
\end{aligned}
\tag{4.8}$$

Für sich genommen ist dies noch keine angemessene Formalisierung, da die oben beschriebene intendierte Bedeutung des speziellen RDF-Vokabulars von dem Modell nicht abgebildet

wird. Um diese Bedeutung nachzubilden, verwendet RDF eine feste Menge an Ableitungsregeln, die sich auf die Elemente des speziellen Vokabulars beziehen und dessen Bedeutung modellieren. Dieser Ansatz entspricht im Wesentlichen dem in [Kapitel 2.2](#) vorgestellten Ansatz der Formalisierung semantischer Netze. Die entsprechenden Ableitungsregeln lassen sich als Logik-Programme darstellen und relativ effizient ausführen, was RDF zu einer sehr praxisrelevanten Ontologiesprache macht. Standardmäßig werden dabei folgende Ableitungsregeln zur Abbildung der Bedeutung des speziellen RDF-Vokabulars verwendet:<sup>1</sup>

**rdf1** Das Prädikat eines Tripels ist immer von Typ `rdf:Property`

$$\text{triple}(S, P, O) \rightarrow \text{triple}(P, \text{rdf} : \text{type}, \text{rdf} : \text{Property})$$

**rdfs2** Werte, die als Subjekt in einem Tripel auftreten, sind von dem unter Umständen durch `rdfs:domain` festgelegten Typ:

$$\begin{aligned} \text{triple}(S, P, O) \wedge \text{triple}(P, \text{rdfs} : \text{domain}, C) \\ \rightarrow \text{triple}(S, \text{rdf} : \text{type}, C) \end{aligned}$$

**rdfs3** Werte, die als Objekt in einem Tripel auftreten, sind von dem unter Umständen durch `rdfs:range` festgelegten Typ:

$$\begin{aligned} \text{triple}(S, P, O) \wedge \text{triple}(P, \text{rdfs} : \text{range}, C) \\ \rightarrow \text{triple}(O, \text{rdf} : \text{type}, C) \end{aligned}$$

---

<sup>1</sup> Der Einfachheit halber verzichten wir auf einige Regeln, die sich auf Datentypen und sogenannte Blank-Notes beziehen. Eine vollständige Liste der Ableitungsregeln findet sich in [\[Hayes, 2004\]](#).

**rdfs4a** Die Subjekte eines Tripels sind vom Typ `rdfs:Ressource`:

$$\text{triple}(S, P, O) \rightarrow \text{triple}(S, \text{rdf} : \text{type}, \text{rdfs} : \text{Ressource})$$

**rdfs4b** Die Objekte eines Tripels sind vom Typ `rdfs:Ressource`:

$$\text{triple}(S, P, O) \rightarrow \text{triple}(O, \text{rdf} : \text{type}, \text{rdfs} : \text{Ressource})$$

**rdfs5** `rdfs:subPropertyOf` ist eine transitive Relation:

$$\begin{aligned} &\text{triple}(P, \text{rdfs} : \text{subPropertyOf}, Q) \\ &\wedge \text{triple}(Q, \text{rdfs} : \text{subPropertyOf}, R) \end{aligned} \quad (4.9)$$

$$\rightarrow \text{triple}(P, \text{rdfs} : \text{subPropertyOf}, R) \quad (4.10)$$

**rdfs6** Jede Relation ist eine Teilrelation von sich selbst:

$$\begin{aligned} &\text{triple}(P, \text{rdf} : \text{type}, \text{rdf} : \text{Property}) \\ &\rightarrow \text{triple}(P, \text{rdfs} : \text{subPropertyOf}, P) \end{aligned}$$

**rdfs7** Ressourcen, die in einer bestimmter Relation zueinander stehen, stehen auch in jeder durch `rdfs:subPropertyOf` bezeichneten allgemeineren Relation zueinander:

$$\begin{aligned} &\text{triple}(S, P, O) \wedge \text{triple}(P, \text{rdfs} : \text{subPropertyOf}, R) \\ &\rightarrow \text{triple}(S, R, O) \end{aligned}$$

**rdfs8** Klassen sind spezielle Ressourcen:

$$\begin{aligned} &\text{triple}(C, \text{rdf} : \text{type}, \text{rdfs} : \text{Class}) \\ &\rightarrow \text{triple}(C, \text{rdfs} : \text{subClassOf}, \text{rdf} : \text{Ressource}) \end{aligned}$$

**rdfs9** Die Objekte, die zu einer bestimmten Klasse gehören, gehören auch zu deren Oberklassen:



$$\begin{aligned} & triple(S, rdf : type, C) \wedge triple(C, rdfs : subClassOf, D) \\ & \rightarrow triple(S, rdf : type, D) \end{aligned}$$

**rdfs10** Jede Klasse ist eine Unterklasse von sich selbst:

$$\begin{aligned} & triple(C, rdf : type, rdfs : Class) \\ & \rightarrow triple(C, rdfs : subClassOf, C) \end{aligned}$$

**rdfs11** Die rdfs:subClassOf Relation ist transitiv:

$$\begin{aligned} & triple(C, rdfs : subClassOf, D) \\ & \wedge triple(D, rdfs : subClassOf, E) \quad (4.11) \\ & \rightarrow triple(C, rdfs : subClassOf, E) \end{aligned}$$

Vervollständigt wird ein RDF-Modell durch eine Reihe von Axiomen, die das unterliegende Datenmodell beschreiben und somit Teil jedes Modells darstellen. In Bezug auf den hier gezeigten Teil von RDF sind dies die folgenden Axiome:

$$\begin{aligned} & triple(rdf : type, rdf : type, rdf : Property) \\ & triple(rdf : type, rdfs : domain, rdfs : Resource) \\ & triple(rdfs : domain, rdfs : domain, rdf : Property) \\ & triple(rdfs : range, rdfs : domain, rdf : Property) \\ & triple(rdfs : subPropertyOf, rdfs : domain, rdf : Property) \\ & triple(rdfs : subClassOf, rdfs : domain, rdfs : Class) \\ & triple(rdf : type, rdfs : range, rdfs : Class) \\ & triple(rdfs : domain, rdfs : range, rdfs : Class) \\ & triple(rdfs : range, rdfs : range, rdfs : Class) \\ & triple(rdfs : subPropertyOf, rdfs : range, rdf : Property) \\ & triple(rdfs : subClassOf, rdfs : range, rdfs : Class) \quad (4.12) \end{aligned}$$

Insgesamt bildet ein RDF-Modell, auch wenn es in der Regel als Graph bzw. als geeignete Serialisierung eines Graphen vorliegt, ein Logik-Programm. Hierbei bilden die beschriebenen Axiome zusammen mit den Beschreibungen der Fakten und der Ontologie die Fakten des Logik-Programms. Die Ableitungsregeln `rdf1` bis `rdfs11` stellen die Regeln des Programms dar. Auf der Grundlage dieser Interpretation als Logik-Programm lassen sich implizite Aussagen über die Ressourcen im Modell ableiten.

Diese ableitbaren Aussagen betreffen sowohl konkrete Objekte der Domäne, als auch Elemente auf der Ebene der Ontologie-Definition. Aussagen, die sich typischerweise über konkrete Objekte ableiten lassen, sind:

**Aussagen über den Typ eines Objektes** Aussagen über den Typ eines Objektes lassen sich auf mehrere Arten ableiten. Wichtigster Mechanismus ist hierbei die Verwendung von `rdfs:domain` und `rdfs:range`, mit deren Hilfe sich der Typ von Objekten aus den Relationen, in denen sie auftreten, ableiten lässt. So ließe sich die Aussage  *$triple(r2, rdf:type, Painting)$*  aus der Tatsache ableiten, dass *r1* in der Relation „paints“ zu *r2* steht. Die andere wesentliche Quelle von Typinformationen ist die Regel `rdfs9`, die besagt, dass Objekte einer Klasse auch in allen Oberklassen dieser Klasse enthalten sind. Diese Regel ermöglicht es, auf der Grundlage vorhandener Aussagen über den Typ eines Objektes weitere abzuleiten. In unserem Beispiel können wir so aus der Tatsache, dass *r2* vom Typ „Painting“ ist und „Painting“ eine Unterklasse von „Artifact“ ist, ableiten, dass  *$triple(r2, rdf:type, Artifact)$*  gilt. Neue Aussagen entstehen hierbei auch vor allem durch die Kombination mit Regel `rdfs5`. Komplexe Ableitungen ergeben sich aus der Kombination all dieser Regeln.

**Aussagen über Relationen zwischen Objekten** Mit Hilfe der Regel *rdfs7* lassen sich auch neue Aussagen über Relationen zwischen Objekten ableiten. So lässt sich durch die Anwendung dieser Regel und der Aussage *triple*(*r1*, *paints*, *r2*) etwa die Aussage *triple*(*r1*, *creates*, *r2*) ableiten. Diese Regel erscheint sehr einfach, sie hat jedoch einen wesentlichen Einfluss auf die Ableitung weiterer Aussagen, da die durch sie erzeugte neue Aussage die Anwendung weiterer Regeln ermöglicht. In unserem Fall können nun auf der Grundlage der *rdfs:range* und *rdfs:domain* Konstrukte über den Typ von *r1* und *r2* abgeleitet werden. Insbesondere lassen sich die Aussagen *triple*(*r1*, *rdf : type*, *Artist*) und über diesen Weg auch die bereits oben erwähnte Aussage *triple*(*r2*, *rdf : type*, *Artifact*) herleiten.

Weitere Möglichkeiten der Herleitung neuer Aussagen ergeben sich auf der Ontologie-Ebene. Hier lassen sich im Wesentlichen die folgenden Typen von Aussagen ableiten:

**Identifikation von Klassen** Da ein RDF-Modell, wie oben beschrieben, sowohl Aussagen auf Objekt- als auch auf Klassenebene enthalten kann, die alle in Form von Tripeln dargestellt werden, ist nicht immer von vornherein klar, welche der Konstanten im Modell für Klassen stehen. Oft muss diese Information erst aus dem Modell abgeleitet werden. Hierbei spielen vor allem die Axiome eine Rolle, welche das RDF-Datenmodell beschreiben und unter anderem festlegen, dass alle Ressourcen, die als Objekt eines *rdf:type* Statements oder als Subjekt oder Objekt eines *rdfs:subClassOf* Statements auftreten, automatisch vom Typ *rdfs:Class* sind. In unserem Beispiel ist keine der enthaltenen Klassen (*Artist*, *Creator* usw.) explizit als solche definiert. Welche der Ressourcen eine Klasse darstellt, ergibt sich vielmehr aus dem Kontext, in dem sie verwendet werden. Da

RDF eine freie Kombination von Konstanten in Aussagen erlaubt, können hierfür jedoch recht komplexe Ableitungen notwendig sein.

**Identifikation von Relationen** Analog hierzu ist in einem RDF-Modell nicht immer von vornherein klar, welche Konstanten Relationen darstellen. In den meisten Fällen ergibt sich dies direkt aus der Position der entsprechenden Ressourcen in den Aussagen. Relationen können jedoch auch beschrieben sein, ohne dass sie als Prädikat eines Tripels verwendet werden. In diesen Fällen lässt sich jedoch ebenfalls aus den Axiomen ableiten, welche der Ressourcen vom Typ `rdf:Property` sind.

### **Aussagen über Subklassen- und Teilrelations-Beziehungen**

Neue Subklassen-Beziehungen ergeben sich im Wesentlichen aus der Transitivität und der Reflexivität der `rdfs:subClassOf` und `rdfs:subPropertyOf` Relationen. Aufgrund der Flexibilität von RDF lassen sich Subklassen-Relationen jedoch zum Beispiel auch durch die Definition einer beliebigen Domänen-Relation als Teil-Relation von `rdfs:subClassOf` erzeugen. Wollen wir etwa WordNet in RDF darstellen, können wir beispielsweise die Hyponym-Relation als Spezialfall von `rdfs:subClassOf` definieren. Hierdurch werden alle Hyponym-Beziehungen automatisch zu Subklassen-Beziehungen und die durch diese Relation verbundenen Ressourcen, welche die Synonymmengen darstellen, werden zu Instanzen von `rdfs:Class`.

Zusammengefasst bietet RDF die Möglichkeit, einfache Ontologie sowie die dazugehörigen Daten auf der Basis einfacher Logik-Programme zu beschreiben. Die Ableitungsregeln zur Berechnung impliziter Fakten sind hierbei ebenso wie eine Reihe von Axiomen fest vorgegeben. Der Benutzer hat lediglich die

Möglichkeit, die Daten sowie die dazugehörige einfache Ontologie zu modellieren. Dies hat auf der einen Seite den Nachteil, dass komplexe, regelhafte Zusammenhänge in einer Domäne in der Regel nicht dargestellt werden können, auf der anderen Seite hat diese Einschränkung den Vorteil eines relativ einfachen und leicht zu lernenden Modells. Aufgrund dieser Einfachheit ist RDF auch die zurzeit wohl am meisten benutzte Ontologiesprache. Die Möglichkeit der freien Kombination von Ressourcen, unabhängig davon, ob diese Objekte, Konzepte oder Relationen darstellen, eröffnet hierbei oft Modellierungsmöglichkeiten, die sich im praktischen Einsatz als sehr nützlich erweisen. Bestes Beispiel ist die oben erwähnte Definition der Hyponym-Relation als Spezialfall der Subklassen-Relationen, welche die Darstellung von semantischen Netzen wie WordNet stark vereinfacht. Darüber hinaus bieten bestehende Inferenzsysteme für RDF oft die Möglichkeit, die Menge der fest vorgegebenen Ableitungsregeln um anwendungsspezifische Regeln zu ergänzen. Um die Verarbeitung dieser Regeln zu vereinfachen, wird häufig gefordert, dass diese die gleiche Struktur besitzen wie die vorgegebenen Regeln. Konkret bedeutet dies, dass der Regelkopf aus einer atomaren Formel der Form  $triple(S, P, O)$  und der Regelkörper aus einer Konjunktion solcher atomaren Formeln besteht.

### 4.1.2 F-Logic

Eine weitere populäre Sprache zur Darstellung von Ontologien, die auf dem Konzept der Logik-Programmierung basiert, ist F-Logic. Streng genommen basiert F-Logic von sich aus nicht auf dem Konzept der Logik-Programmierung, sondern wurde als Erweiterung der Prädikatenlogik um Mechanismen zur natürlicheren Darstellung objektorientierter Konzepte wie Klassen und

Vererbung entwickelt. Die meisten praktischen Implementierungen der F-Logic verwenden jedoch Konzepte der Logik-Programmierung, um eine effiziente Verarbeitung zu erlauben, auch wenn die ursprüngliche Semantik hierdurch abgewandelt wird. Wir orientieren uns hier an der Interpretation von F-Logic als Logik-Programm, das in praktischen Systemen verwendet wird, da diese in der Regel zur Darstellung und Verarbeitung von Ontologien eingesetzt werden.

Im Gegensatz zu RDF mit seiner festen Menge von Ableitungsregeln bietet F-Logic zusätzlich zu einem Datenmodell zur Beschreibung von Klassen und Relationen mit entsprechenden Ableitungsregeln die Möglichkeit, beliebige Ableitungsregeln in Form von Logik-Programmen zu spezifizieren. Dies ist vor allem in komplexen Domänen ein großer Vorteil, da komplizierte Zusammenhänge, die sich in RDF-Schema nicht beschreiben lassen, hierdurch erfasst werden können. Im Folgenden führen wir zunächst die Syntax von F-Logic anhand einiger Beispiele ein und diskutieren dann die Übersetzung in Logik-Programme.

F-Logic bietet eine Reihe spezieller Konstrukte zur Beschreibung von Ontologien, die im Folgenden aufgeführt werden. Wir verwenden hierbei die Syntax von Ontobroker, einer kommerziellen Inferenzmaschine für F-Logic.

**Subklassen:** Klassen werden in F-Logic ebenso wie in RDF und anders als in der Prädikatenlogik durch Konstanten dargestellt. Um auszudrücken, dass zwei Klassen in einer Subklassenbeziehung zueinander stehen, wird eine spezielle zweistellige Relation verwendet. Diese wird in der hier vorgestellten Syntax durch zwei Doppelpunkte in Infix-Notation dargestellt. Die Vererbungsrelationen im oben gezeigten Beispiel lassen sich mit Hilfe dieses Konstruktes folgendermaßen wiedergeben:

```
Painter::Creator  
Painting::Artifact
```

**Instanzen:** Konkrete Objekte werden ebenso wie Klassen als Konstanten dargestellt. Ähnlich wie in RDF ergibt sich die Interpretation einer Konstante als Klasse oder Objekt aus der Position in entsprechenden Definitionen. Die Tatsache, dass ein Objekt Instanz einer Klasse ist, wird in F-Logic hierbei ebenfalls über eine spezielle Relation dargestellt, die der `rdf:type` Relation entspricht. In F-Logic wird diese Relation durch einen Doppelpunkt in Infix-Notation wiedergegeben. Die entsprechenden Aussagen unseres Beispiels besitzen daher folgende Darstellung:

```
r1:Painter  
r2:Painting  
r3:Painting  
r4:Technique
```

**Klassenattribute:** Ein weiteres wichtiges Konstrukt in F-Logic ist die Beschreibung charakteristischer Eigenschaften von Klassen. Sie erfolgt über sogenannte Klassenattribute. Eine entsprechende Definition legt fest, dass eine bestimmte Relation typischerweise in einer bestimmten Relation mit Objekten eines anderen Typs steht. F-Logic unterscheidet hierbei zwischen mehrwertigen und einwertigen Attributen. Mehrwertige Attribute beschreiben Relationen, die zwischen einem Objekt der beschriebenen Klasse und mehreren Objekten einer anderen Klasse bestehen können. Bei einwertigen Attributen kann die entsprechende Relation mit höchstens einem anderen Objekt bestehen. Ein typisches Beispiel für ein mehrwertiges Attribut ist die Relation „paints“ aus unserem Beispiel, da ein Künstler in der Regel mehr als ein Werk erschafft. Der Name des Künstlers kann hingegen als einwertiges Attribut modelliert werden, um eine eindeutige Namensgebung zu erzwingen. Die entsprechenden Definitionen hätten in F-Logic

das folgende Format, bei dem einfache Pfeile für einwertige und Doppelpfeile für mehrwertige Attribute stehen:

```

Creator[created=>>Artifact]
Creator[cname=>String]
Artifact[year=>Integer]
Artifact[name=>String]
Artifact[created-by=>Artist]
Artifact[exemplifies=>>Technique]
Painter[paints=>>Painting]
Painting[painted-by=>Painter]
Technique[exemplified-by=>>Artifact]
Technique[tname=>String]

```

**Instanzattribute:** Analog zur Beschreibung charakteristischer Eigenschaften von Klassen verfügt F-Logic über Mechanismen zur Beschreibung von Ausprägungen der Eigenschaften konkreter Objekte. Im Gegensatz zur Beschreibung von Klassenattributen werden diese konkreten Ausprägungen durch einen Pfeil mit einfacher Linie anstelle der Doppellinie dargestellt. Hierbei wird ebenfalls zwischen einwertigen und mehrwertigen Attributen unterschieden. Die Unterscheidung wird, wie im Fall der Klassenattribute, durch einen einfachen bzw. einen Doppelpfeil dargestellt.

```

r1[paints->>{r2,r3}]
r1[cname->"Rembrandt"]
r2[name->"Self Portrait"]
r2[year->1628]
r3[name->"Stone Bridge"]
r3[year->1638]
r4[tname->"Chiaroscuro"]
r4[exemplified-by->>{r2,r3}]

```

Die beschriebenen Modellierungskonzepte können als spezielle Prädikate mit vorgegebener Bedeutung angesehen werden. Zur Auswertung eines F-Logic Programms werden die Definitionen wieder in die entsprechenden Prädikate übersetzt. Folgende Tabelle 4.1 zeigt diese Übersetzung:



**Tabelle 4.1** Übersetzung von F-Logic Definitionen in prädikatenlogische Ausdrücke

F-Logic Definition	Prädikatenlogische Formel
$C : D$	$\text{sub\_}(C,D)$
$C [A=>R]$	$\text{atttype\_}(C,A,R)$
$C [A=>>R]$	$\text{setatttype\_}C,A,R$
$o : C$	$\text{isa\_}(o,C)$
$o [A->v]$	$\text{att\_}(o,A,v)$
$o [A->\{v1, \dots, vn\}]$	$\text{att\_}(o,A,v1), \dots, \text{att\_}(o,A,vn)$

Ein F-Logic Programm besteht nun aus atomaren Formeln, die sowohl aus den oben gezeigten Definitionen als auch aus atomaren Formeln gemäß der Definition 2.2 bestehen können. Im Gegensatz zu RDF können also auch domänenspezifische Relationen definiert und verwendet werden. Diese Relationen müssen nicht, wie bisher, zweistellig sein. So ist es beispielsweise denkbar, eine dreistellige Relation `exhibited-at` zu definieren, die beschreibt, welche Artefakte in welchem Zeitraum in welchem Museum ausgestellt werden. So könnten wir das obige Modell zum Beispiel um folgende Aussagen erweitern:

```
exhibited-at(r2,Louvre, May)
exhibited-at(r3,Louvre, May)
exhibited-at(r2,Rijksmuseum, June)
exhibited-at(r3,Rijksmuseum, June)
```

F-Logic Programme bestehen nun aus solchen Aussagen und Ableitungsregeln der Form  $B \rightarrow H$ . Hierbei ist  $H$  eine atomare Formel;  $B$  kann eine beliebige prädikatenlogische Formel sein. Quantoren werden hierbei durch die Schlüsselwörter `FORALL` und `EXISTS`, gefolgt von einer Liste durch Kommata voneinander getrennter Variablen dargestellt. Boolesche Operatoren werden durch die Schlüsselwörter `AND`, `OR` und `NOT` wiedergegeben.

Hier einige Beispiele von Ableitungsregeln, die sich in F-Logic beschreiben lassen:

```
FORALL X,Y X[exhibits->>Y] <-
  X:Museum AND
  Y:Artist AND
  (EXISTS T
    exhibited-at(A,X,T) AND
    Y[creates->>A].
```

Die Regel besagt, dass der Ausdruck  $X[\text{exhibited} \rightarrow Y]$  für alle  $X$  und  $Y$  wahr ist, für die gilt, dass  $X$  ein Museum und  $Y$  ein Künstler ist und es einen Zeitpunkt  $T$  gibt, zu dem das Museum ein von dem Künstler geschaffenes Kunstwerk ausgestellt hat. Das Beispiel zeigt hierbei anschaulich die Möglichkeit, vorgegebene und selbstdefinierte Prädikate in Regeln zu kombinieren. Zudem zeigt das Beispiel die Verwendung von Quantoren im Regelkörper. Durch die Verwendung der Disjunktion ließe sich die Formel noch um weitere mögliche Definitionen erweitern. So könnte ein Museum beispielsweise eine Abteilung haben, die einem bestimmten Künstler gewidmet ist. Ein weiteres Beispiel wäre die folgende Regel:

```
FORALL X X:importantPainter <-
  X:Painter AND
  (dead(X) OR
    FORALL Y
      X[paints->>Y] ->
        (has-price(Y,Z) AND Z > 100.000))
```

Diese Regel definiert bedeutende Maler als solche, die entweder bereits tot sind oder deren Bilder alle einen Preis von mehr als hunderttausend (Euro) haben. Diese Regel zeigt die Verwendung weiterer Sprachelemente in Regeln. Neben der Disjunktion wird hier eine Implikation im Regelkörper verwendet. Zudem gebraucht die Regel eine spezielle Relation zum Vergleich numerischer Werte. Praktische Implementierungen von F-Logic

verfügen über eine Reihe solcher sogenannter Build-In Prädikate, die eine angemessene Behandlung konkreter Daten wie Zahlen oder Wörter erlauben.

Da im Regelkörper allgemeine prädikatenlogische Formeln auftreten können, ist es zur effizienten Ausführung notwendig, diese in ein Logik-Programm zu übersetzen. Insbesondere müssen explizite Quantoren und Implikationen entfernt werden. Hierzu wird die sogenannte Lloyd-Topor Transformation verwendet. Diese ist in der Lage, mit Hilfe folgender Transformationsregeln eine Formel der oben beschriebenen Form in ein PROLOG Program zu übersetzen:

$$lt(W) =_{def} W, W \text{ ist ein atomare Formel} \quad (4.13)$$

$$lt(W_1 \wedge W_2) :=_{def} lt(W_1) \wedge lt(W_2) \quad (4.14)$$

$$lt(W_1 \vee W_2) :=_{def} lt(W_1) \vee lt(W_2) \quad (4.15)$$

$$lt(W_1 \rightarrow W_2) :=_{def} lt(\neg W_1 \vee W_2) \quad (4.16)$$

$$lt(\forall x W) :=_{def} lt(\neg \exists x \neg W) \quad (4.17)$$

$$lt(\exists x W) :=_{def} lt(W) \quad (4.18)$$

$$lt(\neg \neg W) :=_{def} lt(W) \quad (4.19)$$

$$lt(\neg(W_1 \wedge W_2)) :=_{def} lt(\neg W_1) \vee lt(\neg W_2) \quad (4.20)$$

$$lt(\neg(W_1 \vee W_2)) :=_{def} lt(\neg W_1) \wedge lt(\neg W_2) \quad (4.21)$$

$$lt(\neg(W_1 \rightarrow W_2)) :=_{def} lt(\neg(\neg W_1 \vee W_2)) \quad (4.22)$$

$$lt(\neg \forall x W) :=_{def} lt(\exists x \neg W) \quad (4.23)$$

$$lt(\neg \exists x W) :=_{def} \neg lt(W) \quad (4.24)$$

Die Übersetzung eines F-Logic Programms in ein Logik-Programm erfolgt nun in zwei Schritten. Zunächst werden die entsprechenden Regeln in Prädikatenlogik transformiert. Eine entsprechende Übersetzung der ersten der beiden oben gezeigten Beispielregeln wäre die folgende:

$$isa\_ (X, museum) \wedge isa\_ (Y, artist) \wedge \quad (4.25)$$

$$\exists T (exhibited - at(A, X, T) \wedge setatt\_ (Y, creates, A)) \quad (4.26)$$

$$\rightarrow setatt\_ (X, exhibits, Y) \quad (4.27)$$

Die Anwendung der Transformationsregeln entfernt den Existenzquantor aus der Formel und erzeugt das folgende Logik-Programm, welches mit entsprechenden Sprachen wie zum Beispiel PROLOG ausgeführt werden kann:

$$isa\_ (X, museum) \wedge isa\_ (Y, artist) \wedge \quad (4.28)$$

$$exhibited - at(A, X, T) \wedge setatt\_ (Y, creates, A) \quad (4.29)$$

$$\rightarrow setatt\_ (X, exhibits, Y) \quad (4.30)$$

Die zweite der gezeigten Regeln kann auf diese Weise ebenfalls in ein Logik-Programm übersetzt werden. Die zunächst erzeugte prädikatenlogische Formel hat hierbei die folgende Form:

$$isa\_ (X, Painter) \wedge (dead(X) \vee (\forall Y setatt\_ (X, paints, Y) \rightarrow \quad (4.31)$$

$$(has - price(Y, Z) \wedge Z > 100.000))) \rightarrow isa\_ (X, importantPainter) \quad (4.32)$$

Die Anwendung der Lloyd-Topor Transformation auf den Körper dieser Regel führt zu einer Reihe von Veränderungen. Zunächst wird der Allquantor durch einen negierten Existenzquantor ersetzt und die quantifizierte Formel negiert. Anschließend wird der entstandene Existenzquantor eliminiert und die Implikation mit Hilfe der Regel für negierte Implikationen in eine Disjunktion umgewandelt. Durch eine anschließende Anwendung der De Morganschen Regel entsteht das folgende Logik-Programm, welches mit Hilfe bestehender Systeme ausgewertet werden kann:

$$isa\_ (X, Painter) \wedge dead(X) \vee \quad (4.33)$$

$$(\neg setatt\_ (X, paints, Y) \vee (has - price(Y, Z) \wedge (Z > 100.000))) \quad (4.34)$$

$$\rightarrow isa\_ (X, importantPainter) \quad (4.35)$$

Das beschriebene Vorgehen ermöglicht zwar die Auswertung benutzerdefinierter Regeln, die inhärente Bedeutung der speziellen Sprachkonstrukte wie Vererbung und Instantiierung werden hierdurch jedoch nicht abgedeckt. Hierfür ist zusätzlich wie bei RDF eine Menge von fixen Ableitungsregeln, die über die speziellen Sprachkonstrukte von F-Logic definiert sind und deren Bedeutung festlegen, notwendig. Die entsprechenden Regeln haben hierbei durchaus Ähnlichkeit mit den in RDF verwendeten Ableitungsregeln. So sind beispielsweise Regeln notwendig, welche die Transitivität der Subklassenrelation sowie die Vererbung von Eigenschaften festlegen. Entsprechende Regeln sind unter anderem die folgenden:

$$sub\_ (A, B) \wedge sub\_ (B, C) \rightarrow sub\_ (A, C)$$

$$isa\_ (o, A) \wedge sub\_ (A, B) \rightarrow isa\_ (o, B)$$

Leider ist es wesentlich schwieriger, eindeutige Ableitungsregeln zur Vererbung von Eigenschaften und deren konkreter Werte festzulegen. Dies liegt daran, dass durch die kombinierte Benutzung fixer Regeln, welche die Semantik der Vererbungsrelation beschreiben, und benutzerdefinierter Regeln Konflikte auftreten können. Arten von Vererbung, die relativ unproblematisch sind, betreffen die Klassenattribute. Die entsprechenden Einschränkungen können vererbt werden, da die Ableitung weiterer Werte für eine entsprechende Relation keinen Konflikt ergibt. Dementsprechend könnte die Vererbung von Einschränkungen auf Attributen durch folgende Regeln modelliert werden:

$$sub\_ (A,B) \wedge atttype(A,M,C) \rightarrow atttype(B,M,C)$$

$$sub\_ (A,B) \wedge setatttype(A,M,C) \rightarrow setatttype(B,M,C)$$

$$att\_ (o,M,v) \wedge isa\_ (o,A) \wedge atttype\_ (A,M,C) \rightarrow isa\_ (v,C)$$

$$att\_ (o,M,v) \wedge isa\_ (o,A) \wedge setatttype\_ (A,M,C) \rightarrow isa\_ (v,C)$$

Probleme treten bei einwertigen Attributen auf, wenn Klassen konkrete Werte von Attributen an ihre Instanzen vererben. Eine solche Vererbung konkreter Werte könnte durch eine einfache Ableitungsregel implementiert werden:

$$isa\_ (o,A) \wedge att\_ (A,M,v) \rightarrow att\_ (o,M,v) \quad (4.36)$$

Die folgenden Modelle zeigen typische Beispiele, in denen die Verwendung dieser Regel Probleme bereitet:

```
o:c
c[m -> a]
o[m -> b] <- o[m -> a]
```

Durch die Anwendung von Regel 4.36 lässt sich hier aus den Fakten  $o:c$  und  $c[m \rightarrow a]$  die Aussage  $o[m \rightarrow a]$  ableiten. Diese wiederum löst die im Modell enthaltene benutzerdefinierte Regel aus, welche die Aussage  $o[m \rightarrow b]$  erzeugt. Sie steht jedoch mit der zuvor abgeleiteten Aussage im Widerspruch, da es sich bei  $m$  um eine einwertige Relation handelt, die nicht beide Werte  $a$  und  $b$  annehmen kann.

Es gibt eine Reihe von Möglichkeiten, dieses Problem zu lösen. Diese sollen hier kurz erwähnt werden, da es sich um recht grundlegende Ansätze zum Umgang mit Konflikten in formalen Sprachen zur Darstellung von Ontologien handelt.

**Syntaktische Einschränkungen** Die einfachste Art, die eben genannten Konflikte zu vermeiden, ist die Einschränkung von

F-Logic Modellen auf syntaktischer Ebene. Im hier gezeigten Fall kann das Problem vermieden werden, indem eine klare Unterscheidung zwischen Klassen- und Objektkonstanten eingeführt und zusätzlich gefordert wird, dass Klassenattribute nur für Klassenkonstanten und Objektattribute nur für Objektkonstanten definiert werden dürfen. In diesem Fall kann Regel 4.36 niemals angewendet werden, da im Beispiel oben  $A$  nicht gleichzeitig in beiden Bedingungen des Regelkörpers vorkommen darf.

**Nicht-eindeutige Namen** Eine andere Möglichkeit, den beschriebenen Konflikt zu verhindern, ist die Aufgabe der Annahme, dass Objekte eindeutige Namen besitzen. In unserem Beispiel gäbe es keinen Konflikt, wenn  $a$  und  $b$  das gleiche eindeutige Objekt beschreiben würden, welches den Wert des einwertigen Attributs  $m$  darstellt. Zur Umsetzung dieser Lösung ist eine zusätzliche Regel notwendig, die ableitet, welche Namen das gleiche Objekt beschreiben. Eine solche Regel könnte wie folgt aussehen:

$$\begin{aligned} att\_ (o, M, v1) \wedge att\_ (o, M, v2) \wedge isa\_ (o, A) \wedge atttype\_ (A, M, V) \\ \rightarrow v1 = v2 \end{aligned}$$

**Logische Inkonsistenz** Eine dritte Möglichkeit mit dem oben genannten Konflikt umzugehen, ist diesen im logischen Modell explizit zu modellieren. Dies bedeutet, eine Regel einzuführen, die F-Logic Programme, welche einwertigen Attributen unterschiedliche Werte zuordnen, sei es explizit oder durch entsprechende Ableitungsregeln, auf logischer Ebene inkonsistent macht. Eine solche Regel würde den Wahrheitswert „falsch“ ableiten, wann immer der beschriebene Konflikt auftritt. Hierzu ist es sinnvoll, die Verwendung

von Wahrheitswerten in Formeln zu erlauben.<sup>2</sup> In der Regel wird der Wert „wahr“ durch das Symbol  $\top$  und der Wert „falsch“ durch  $\perp$  dargestellt. Eine entsprechende Ableitungsregel wäre die folgende:

$$\begin{aligned} att\_ (o, M, v1) \wedge att\_ (o, M, v2) \wedge isa\_ (o, A) \wedge atttype\_ (A, M, V) \\ \rightarrow \perp \end{aligned}$$

### 4.1.3 Fazit

Zusammenfassend lässt sich feststellen, dass Logik-Programme einen praxisnahen und vergleichsweise effizient verarbeitbaren Ansatz zur Formalisierung von Ontologien bieten. Mit ihrer Hilfe lassen sich hierbei sowohl einfache Modellierungsansätze wie RDF als auch komplexere, benutzerdefinierte Regeln abbilden, wie dies durch F-Logic möglich ist. Wie wir gesehen haben, werden Klassen ebenso wie Objekte in bekannten Sprachen wie RDF und F-Logic häufig als logische Konstanten dargestellt und nicht, wie in [Kapitel 2.2](#) beschrieben, als einstellige Prädikate. Der Hauptgrund hierfür ist die Möglichkeit der Metamodellierung, die sich hierdurch ergibt. Konstanten können gleichzeitig für Konzepte und Objekte stehen, so dass die Unterscheidung verschwimmt. Dies widerspricht zwar den in [Kapitel 1](#) vorgestellten Prinzipien, bietet in der Praxis jedoch häufig Vorteile. Ein vielzitiertes Beispiel ist die Modellierung des Tierreiches und seiner Arten, Rassen und Gattungen. Hierbei ist relativ unumstritten, dass ein konkretes Tier, etwa Clyde der Elefant aus [Kapitel 2.2](#), ein Objekt ist, welches zur Klasse der Elefanten gehört. Andererseits ist Elefant eine Instanz der Klasse aller Ar-

---

<sup>2</sup> In Logiken, die Gleichheit unterstützen, können Wahrheitswerte durch die Formeln  $c = c$  bzw.  $\neg c = c$  dargestellt werden.



ten, so dass in dieser Beziehung die Klasse Elefant ein Objekt ist, welches zu einer (Meta-)Klasse gehört. Mit Hilfe der in diesem Kapitel beschriebenen Formalismen lassen sich diese Zusammenhänge ohne Probleme beschreiben. Wie wir am Ende des letzten Abschnittes gesehen haben, bringt das Fehlen einer klaren Trennung zwischen Objekten und Klassen jedoch auch Probleme mit sich, die sich oft auch auf die Entscheidbarkeit der entsprechenden Logiken auswirken. Aus diesem Grund wurde unter der Bezeichnung Beschreibungslogik eine Gruppe von Logiken entwickelt, welche speziell auf die Beschreibung von Ontologien ausgerichtet sind und eine klare Trennung zwischen Klassen und Objekten vornehmen. Diese Logiken, die in gewisser Weise in Konkurrenz zu den bisher beschriebenen Ansätzen stehen und sich mehr an den in [Kapitel 2.2](#) beschriebenen Prinzipien orientieren, werden im nachfolgenden Kapitel behandelt.

## 4.2 Beschreibungslogik

Beschreibungslogiken sind spezielle Logiken, die seit Ende der achtziger Jahre entwickelt wurden, um terminologisches Wissen angemessen formalisieren zu können. Diese Logiken besitzen eine Reihe von Eigenschaften, die sie sehr nützlich für die Darstellung von Ontologien machen. Zum einen realisieren sie, wie oben angedeutet, eine klare Trennung zwischen Klassen und Objekten und vermeiden somit die beschriebenen Probleme. Darüber hinaus verfügen diese Logiken über spezielle Konstrukte zur Beschreibung typischer Eigenschaften von Klassen und deren Objekten, wie sie bereits in [Kapitel 2.2](#) erläutert wurden. Hierbei stellen Beschreibungslogiken eine eindeutig definierte Teilmenge der Prädikatenlogik erster Stufe dar. Sie haben mit der Prädikatenlogik zwar den Vorteil einer eindeutigen Semantik

gemeinsam, ermöglichen durch die Einschränkung der Sprache jedoch relativ effiziente Inferenzmethoden. Im Folgenden führen wir zunächst die wichtigsten Elemente von Beschreibungslogiken anhand eines Beispiels ein. Anschließend wird die Relation zwischen Beschreibungslogiken und Prädikatenlogik erster Stufe genauer betrachtet. Hierbei werden wir sehen, dass einige der in Beschreibungslogiken verwendeten Elemente im Prinzip Kurzschreibweisen für typische Definitionen sind, wie sie in [Kapitel 2.2](#) diskutiert wurden. Am Ende dieses Kapitels werden wir zusätzlich das Verhältnis der Web Ontology Language OWL zu Beschreibungslogiken diskutieren.

### ***4.2.1 Grundlagen von Beschreibungslogiken***

Ein grundlegendes Prinzip von Beschreibungslogiken ist, wie bereits erwähnt, die Trennung der Beschreibung von konkreten Objekten und ihren Beziehungen auf der einen und Konzepten und deren Definitionen auf der anderen Seite. Die Beschreibungen konkreter Objekte erfolgt hierbei in einer sogenannten A-Box. Diese besteht aus atomaren Formeln, die weder Variablen noch Funktionssymbole enthalten. Außerdem beschränken Beschreibungslogiken in der Regel die Verwendung von ein- und zweistelligen Prädikaten. Einstellige Prädikate repräsentieren hierbei Konzepte, zweistellige Prädikate stehen für Relationen zwischen Objekten, die jeweils durch Konstanten dargestellt werden. Da Beschreibungslogiken nicht immer davon ausgehen, dass Objekte eindeutig durch Konstanten bezeichnet werden, erlauben manche Beschreibungslogiken die Verwendung von Gleichheit und Ungleichheit von Konstanten. Außerdem wird oft zwischen solchen Relationen unterschieden, die zwei Objekte miteinander und solchen, die ein Objekt mit

einem konkreten Wert für eine bestimmte Eigenschaft verbinden. Eine entsprechende A-Box zur Beschreibung der Objekte in Abb. 2.3 in [Kapitel 2](#) wäre die folgende:

$$\begin{aligned}
 & Elephant(e1) \\
 & has - friend(e1, clement) \\
 & has - friend(e1, cleopatra) \\
 & has - legs(e1, '3') \\
 & e1 = clyde, e1 \neq clement, e1 \neq cleopatra
 \end{aligned}$$

So weit unterscheiden sich Beschreibungslogiken nicht wesentlich von den im letzten Abschnitt diskutierten Logik-Programmen, in denen faktisches Wissen auf ganz ähnliche Weise dargestellt wurde, auch wenn im Allgemeinen weniger Einschränkungen in Bezug auf die Stelligkeit von Relationen und die Verwendung von Funktionssymbolen bestanden. Vergleichen wir A-Boxen mit der Darstellung von Fakten in RDF, so fällt auf, dass zwar die logische Darstellung eine andere ist (in RDF wird, wie beschrieben, die allgemeine Relation „triple“ eingesetzt, um Fakten zu beschreiben, Beschreibungslogiken verwenden dagegen die klassische Art der Darstellung durch spezifische Prädikate). Trotz dieser Unterschiede lässt sich der Inhalt einer A-Box, mit Ausnahme von Gleichheit und Ungleichheit, direkt nach RDF übersetzen.<sup>3</sup> Das entsprechende RDF Modell wäre das folgende:

$$\begin{aligned}
 & triple(e1, rdf : type, Elephant) \\
 & triple(e1, has - friend, clement)
 \end{aligned}$$

---

<sup>3</sup> Gleichheit und Ungleichheit sind in RDF nicht vorgesehen, da die Eindeutigkeit von Namen eine grundlegende Annahme des RDF-Datenmodells ist.

$$\text{triple}(e1, \text{has} - \text{friend}, \text{cleopatra})$$

$$\text{triple}(e1, \text{has} - \text{legs}, '3')$$

Diese Darstellbarkeit von A-Boxen in RDF bildet die Basis für die Verwendung von Beschreibungslogiken bei der formalen Wiedergabe von Ontologien im Web.

Während RDF, wie oben beschrieben, nur relativ einfache Möglichkeiten zur Definition von Ontologien bietet, stellen Beschreibungslogiken hier eine ausdrucksmächtige logische Sprache zur Verfügung, die es erlaubt, komplexe Konzepte durch notwendige und hinreichende Bedingungen (vgl. [Kapitel 1](#)) zu beschreiben. Diese Beschreibung von Konzepten wird auch als T-Box bezeichnet und bildet das Kernstück einer Ontologie-Definition in Beschreibungslogik. Obwohl Beschreibungslogiken letztendlich eine Teilmenge der Prädikatenlogik erster Stufe sind, verwenden diese zur Beschreibung der T-Box sowohl eine abweichende Syntax als auch eine abweichende Semantik, die sich näher an der Bedeutung von Ontologien orientiert, als die bereits eingeführte Prädikatenlogik. Im Folgenden führen wir zunächst die üblicherweise verwendete Syntax und Semantik von Beschreibungslogiken ein und beschreiben anschließend ihr Verhältnis zur Prädikatenlogik.

## Konzepte und Relationen

Grundlegende Bausteine einer T-Box sind Konzepte und Relationen. Diese werden nicht durch ein Prädikat, sondern durch ihre Namen dargestellt. In unserem Beispiel wären also *Elephant* und *Mammal* Konzepte und *has – friend*, *has – legs*, *has – hair* usw. Relationen. Diese Namen stehen im Falle von Konzepten für eine Menge von Objekten, die zu dem Konzept gehören. Relationennamen repräsentieren eine Menge von Objektpaaren,

also Paare von Objekten, die in der entsprechenden Relation zueinander stehen. Diese mengentheoretische Interpretation von Konzepten und Relationen bildet auch die Grundlage für die Definition der formalen Semantik: Die Semantik eines Beschreibungslogik-Modells (A-Box plus T-Box) wird durch die Menge aller denkbaren Objekte sowie durch eine Funktion, die Objekte, Konzepte und Relationen des Modells in diese Menge abbildet, bestimmt. Die Menge wird hierbei in der Regel durch  $\Delta$  und die Abbildungsfunktion durch  $\cdot^I$  bezeichnet. In Fällen, in denen konkrete Datenwerte verwendet werden, wird zusätzlich noch ein Wertebereich  $D$  mit  $D \cap \Delta = \emptyset$  definiert, aus dem die Datenwerte stammen. Die Abbildung  $\cdot^I$  ist eine Interpretation einer Ontologie, wenn folgende Bedingungen erfüllt sind:

- Ist  $o$  ein Objektname, so gilt  $o^I \in \Delta$ .
- Ist  $C$  ein Konzeptname, so gilt  $C^I \subseteq \Delta$ .
- Ist  $R$  eine Relation, so gilt  $R^I \subseteq \Delta \times \Delta$  oder  $R^I \subseteq \Delta \times D$  falls die Relation Datenwerte enthält.

Diese formale Definition deckt sich weitestgehend mit der grundlegenden Idee von Kategoriensystemen, die bereits in [Kapitel 1](#) beschrieben wurden. Die Interpretation  $\cdot^I$  ordnet jeder Kategorie ihre Extension, also die Menge aller Objekte, welche die charakteristischen Eigenschaften der Kategorie besitzen, zu. Ein weiterer Vorteil dieses Ansatzes, neben der Kompatibilität mit grundlegenden Konzepten von Ontologien, ist die Möglichkeit, auf der Basis mengentheoretischer Konzepte komplexe Konzepte und ihr Verhältnis zueinander zu beschreiben. Einige Beschreibungslogiken besitzen darüber hinaus eine R-Box, in der komplexe Relationen definiert werden. Dieser Aspekt von Beschreibungslogiken wird im nachfolgenden Kapitel zu Ontologien und Regeln genauer betrachtet.

## Konzeptausdrücke

Konzeptausdrücke beschreiben Mengen von Objekten mit bestimmten charakteristischen Eigenschaften. Die Beschreibung dieser Eigenschaften erfolgt durch die Anwendung spezieller logischer Operatoren auf Konzept-, Relations- und Objektnamen. Jeder dieser Operatoren besitzt hierbei eine mengentheoretische Interpretation, die es erlaubt, die Menge der beschriebenen Objekte relativ zu den durch Konzept- und Relationsnamen angegebenen Mengen zu bestimmen. Man kann sich dieses Prinzip gut an den üblichen Booleschen Operatoren Konjunktion, Disjunktion und Negation verdeutlichen.

Angenommen, wir haben drei Konzeptnamen: *Elephant*, *Friend* und *Lonely*. Intuitiv bezeichnet hierbei *Elephant* die Menge aller Elefanten, *Friend* die Menge aller Objekte, die Freunde haben, und *Lonely* die Menge aller Objekte, die keine Freunde besitzen. Aufgrund dieser Interpretation der Konzeptnamen würde das Objekt Clyde ein Element der Konzepte *Elephant* und *Friend* sein. Die meisten Beschreibungslogiken erlauben es nun, Boolesche Ausdrücke über Konzeptnamen zu bilden:

**Konjunktion**  $Elephant \sqcap Friend$  bezeichnet die Menge aller Objekte, die sowohl Instanzen des Konzeptes *Elephant* als auch des Konzeptes *Friend* sind.

**Disjunktion**  $Elephant \sqcup Friend$  bezeichnet die Menge aller Objekte, die Instanzen des Konzeptes *Elephant* oder des Konzeptes *Friend* sind.

**Negation**  $\neg Lonely$  bezeichnet die Menge aller denkbaren Objekte, die nicht Instanzen des Konzeptes *Lonely* sind.

Anhand dieser Definitionen können wir nun leicht entscheiden, ob ein bestimmtes Objekt ein Element der durch solch einen Ausdruck beschriebenen Menge ist. Wenn wir das Objekt

Clyde als Beispiel nehmen, so folgt direkt, dass Clyde ein Element der durch  $\textit{Elephant} \sqcap \textit{Friend}$  und  $\textit{Elephant} \sqcup \textit{Friend}$  bezeichneten Mengen ist, da wir wissen, dass er zu jedem der beteiligten Konzepte gehört. Intuitiv können wir auch ableiten, dass Clyde auch ein Element der durch  $\neg \textit{Lonely}$  bezeichneten Menge sein müsste, da er ja Freunde hat und daher nicht einsam ist. Um diese intuitive Folgerung auch formal durchführen zu können, ist es notwendig, die intuitive Beschreibung der Konzepte *Friend* und *Lonely* zu formalisieren. Wie für den Fall der Logik-Programmierung bereits oben beschrieben, gibt es auch im Kontext von Beschreibungslogiken spezielle Symbole für die Wahrheitswerte wahr und falsch, nämlich  $\top$  und  $\perp$ . Im Fall von Beschreibungslogiken besitzen diese eine mengentheoretische Interpretation und repräsentieren die Menge aller Objekte (also  $\Delta$ ) und die leere Menge. Zusätzlich bieten einige Logiken die Möglichkeit, Konzepte durch das explizite Aufzählen von Objekten zu definieren. So steht  $\{\textit{Clyde}, \textit{Clement}, \textit{Cleopatra}\}$  für das Konzept, welches genau die drei Objekte Clyde, Clement, Cleopatra als Instanzen besitzt.

Es ist offensichtlich, dass sich Konzepte wie *Friend* und *Lonely* nicht allein auf der Grundlage von Konzeptnamen formulieren lassen, da die Beziehungen zwischen Objekten eine wesentliche Rolle im Hinblick auf deren intendierte Bedeutung spielen. Um auch solche Konzepte beschreiben zu können, verfügen Beschreibungslogiken über spezielle Operatoren zur Darstellung bestimmter Einschränkungen hinsichtlich der Existenz und Natur von Relationen zu anderen Objekten. Hier einige Beispiel zur Verwendung dieser Operatoren:

**Existenzrestriktion** Die Existenzrestriktion fordert die Existenz einer Relation zu einem anderen Objekt. Mit Hilfe dieser Restriktion lässt sich zum Beispiel das Konzept *Friend* als die Menge aller Objekte beschreiben, die mit mindestens

einem beliebigen Objekt in der friend-of Beziehung steht. Der entsprechende Ausdruck ist:

$$\exists \text{friends}$$

Da Clyde in unserem Beispiel Freunde hat, ist er automatisch ein Element dieser Menge. Zusätzlich kann noch gefordert werden, dass das entsprechende Objekt von einem bestimmten Typ ist. So beschreibt etwa der folgende Ausdruck die Menge aller Objekte, die mit einem Elefanten befreundet sind:

$$\exists \text{friends.Elephant}$$

Nur auf der Grundlage der oben angegebenen Informationen über die Objekte in unserem Beispiel können wir nicht entscheiden, ob Clyde ein Element der durch diesen Ausdruck beschriebenen Menge ist, da nicht bekannt ist, ob Clement und Cleopatra vom Typ Elefant sind, was eine Voraussetzung hierfür wäre.

**Typrestriktion** Die Typrestriktion fordert, dass alle Objekte, die sich in einer bestimmten Relation mit dem beschriebenen Objekt befinden, sofern es solche Objekte gibt, von einem bestimmten Typ sind. Diese Restriktion kann zum Beispiel verwendet werden, um alle Objekte zu beschreiben, die ausschließlich mit Elefanten befreundet sind. Der entsprechende Konzeptausdruck wäre folgender:

$$\forall \text{friends.Elephant}$$

Wiederum können wir nicht entscheiden, ob Clyde zu der durch diesen Ausdruck beschriebenen Menge von Objekten gehört, da wir den Typ von Clement und Cleopatra nicht kennen. Aufgrund der Open World Annahme gehört ein Objekt erst zu dieser Menge, wenn aus seiner Definition eindeutig abgeleitet werden kann, dass alle Objekte, die sich in der friends Relation zu diesem Objekt befinden, vom



Typ „Elephant“ sein müssen oder aber wenn sich ableiten lässt, dass es kein Objekt in der friends Relation geben kann. Dieser letzte Fall sorgt häufig für Verwirrung, da er oft zu nicht intuitiven Ableitungen führt.

**Kardinalitätsrestriktionen** Ausdrucksmächtige Beschreibungslogiken verfügen neben den oben genannten Arten von Restriktionen zusätzlich über die Möglichkeit, Restriktionen auf der Kardinalität von Relationen, also der Anzahl von Objekten, mit denen sich das beschriebene Objekt in einer bestimmten Relation befindet, zu formulieren. Grundsätzlich können hierbei obere und untere Grenzen für die Anzahl angegeben werden. Durch die Kombination von entsprechenden Ausdrücken können auch exakte Anzahlen gefordert werden. So können wir beispielsweise das bereits erwähnte Konzept „Lonely“ präziser durch eine Restriktion beschreiben, die angibt, dass die Anzahl der Objekte in der friends Relation kleiner oder gleich 0 ist:

$$\leq 0 \text{ friends}$$

Anhand dieses Ausdrucks wird klar, dass eine Beziehung zwischen den bereits beschriebenen Restriktionen und den Kardinalitätsrestriktionen besteht. Insbesondere ist der gezeigte Ausdruck äquivalent zu folgendem:  $\neg \exists \text{ friends}$ . Zusätzlich erlauben Kardinalitätsrestriktionen jedoch die Angabe beliebiger unterer und oberer Grenzen. Die entsprechenden Ausdrücke lassen sich nicht mehr durch Existenz- und Typrestriktionen ausdrücken, zum Beispiel:

$$\geq 5 \text{ friends} \sqcap \leq 10 \text{ friends}$$

Analog zu den Existenzrelationen beinhalten ausdrucks mächtige Beschreibungslogiken oft auch sogenannte qualifizierte Kardinalitätsrestriktionen. Diese erlauben es, Kardinalitätsrestriktionen in Bezug auf Objekte eines

bestimmten Typs zu formulieren. So lassen sich etwa unterschiedliche Schranken für menschliche und tierische Freunde angeben. Folgender Ausdruck etwa beschreibt Objekte, die mit keinem Menschen, aber mit mindestens zwei Elefanten befreundet sind:

$$\leq 0 \text{ friends.Human} \sqcap \geq 2 \text{ friends.Elephant}$$

Es kann leicht gezeigt werden, dass sich mit Hilfe dieser qualifizierten Kardinalitätsrestriktionen alle bisher beschriebenen Restriktionen darstellen lassen. Insbesondere gilt:

$$\exists \text{ friends ist äquivalent zu } \geq 1 \text{ friends.}\top \quad (4.37)$$

$$\exists \text{ friends.Elephant ist äquivalent zu } \geq 1 \text{ friends.Elephant} \quad (4.38)$$

$$\forall \text{ friends.Elephant ist äquivalent zu } \leq 0 \text{ friends.}(\neg \text{Elephant}) \quad (4.39)$$

Wie bereits in den Beispielen angedeutet, lassen sich diese Konzeptausdrücke beliebig miteinander kombinieren, um komplexe Definitionen zu bilden. Vor allem kann an jeder Stelle, an der ein Konzeptname (etwa „Elephant“) auftritt, ebenso ein komplexer Konzeptausdruck erscheinen. Hieraus ergibt sich folgende formale Definition komplexer Konzeptausdrücke:

- Konzeptnamen sind Konzeptausdrücke.
- Seien  $o_1, \dots, o_n$  Objektnamen, dann ist  $\{o_1, \dots, o_n\}$  ein Konzeptausdruck.
- Seien  $C$  und  $D$  Konzeptausdrücke, dann sind auch  $C \sqcap D$ ,  $C \sqcup D$  und  $\neg C$  Konzeptausdrücke.
- Seien  $C$  ein Konzeptausdruck,  $r$  ein Relationenname und  $n$  eine positive natürliche Zahl, dann sind auch  $\exists r$ ,  $\exists r.C$ ,  $\forall r.C$ ,  $\leq nr$ ,  $\geq nr$  und  $= nr$  sowie  $\leq nr.C$ ,  $\geq nr.C$  und  $= nr.C$  Konzeptausdrücke.

Auf der Basis dieser rekursiven Definition lässt sich nun auch leicht die formale Bedeutung von Konzeptausdrücken festlegen. Hierzu gibt es zwei unterschiedliche Ansätze. Zum einen kann die mengentheoretische Interpretation der einzelnen Konstrukte auf der Menge aller Objekte angegeben werden, zum anderen können wir eine Übersetzungsfunktion in die Prädikatenlogik eingeben. Wir verwenden hier letztere Option, um die Verbindung zu der in [Kapitel 2.2](#) beschriebenen Formalisierung von Ontologien mit Hilfe von Prädikatenlogik herzustellen. Zu diesem Zweck definieren wir eine Übersetzungsfunktion  $T$  von der Menge aller Konzeptausdrücke in die Menge der definierten prädikatenlogischen Formeln von [Kapitel 2.2](#). Die Übersetzungsfunktion ist mit Variablennamen parametrisiert, um sicherzustellen, dass die entsprechenden Namen korrekt vergeben werden. Diese Übersetzungsfunktion ist wie folgt definiert:

Konzeptausdruck $A$	Übersetzung $T^x(A)$
$\top$	$x = x$
$\perp$	$x \neq x$
$\{o_1, \dots, o_n\}$	$x = o_1 \vee \dots \vee x = o_n$
$C$	$C(x)$
$C \sqcap D$	$T^x(C) \wedge T^x(D)$
$C \sqcup D$	$T^x(C) \vee T^x(D)$
$\neg C$	$\neg T^x(C)$
$\exists r$	$\exists y T^{x,y}(r)$
$\exists r.C$	$\exists y (T^{x,y}(r) \wedge T^y(C))$
$\forall r.C$	$\forall y (T^{x,y}(r) \rightarrow T^y(C))$
$\leq nr$	$\exists^{\leq n} y (T^{x,y}(r))$
$\geq nr$	$\exists^{\geq n} y (T^{x,y}(r))$
$= nr$	$\exists^{=n} y (T^{x,y}(r))$
$\leq nr.C$	$\exists^{\leq n} y (T^{x,y}(r) \wedge T^y(C))$
$\geq nr.C$	$\exists^{\geq n} y (T^{x,y}(r) \wedge T^y(C))$
$= nr.C$	$\exists^{=n} y (T^{x,y}(r) \wedge T^y(C))$

Die Übersetzung von Relationennamen ist hierbei gegeben durch  $T^{x,y}(r) = r(x,y)$ . Bei Logiken, die komplexe Definitionen von Relationen erlauben, kann diese Übersetzung jedoch auch komplizierter ausfallen.

## Axiome

Die bisher betrachteten Konstrukte erlauben es uns, auf der Basis von Konzept-, Relations- und Objektnamen komplexe Konzeptausdrücke zu definieren, die eine bestimmte Menge von Objekten beschreiben. Wie im Kapitel über Konzepte und Relationen erläutert, werden Konzeptausdrücke verwendet, um Konzepte wie das erwähnte *Lonely* genauer zu beschreiben. Hierzu ist es jedoch notwendig, eine Verbindung zwischen Konzeptnamen und der entsprechenden Definition in Form eines Konzeptausdrucks herzustellen. Dies erfolgt in Form von Axiomen. In der allgemeinsten Form heißt ein Axiom

$$C \sqsubseteq D,$$

wobei  $C$  und  $D$  Konzeptausdrücke und  $\sqsubseteq$  die sogenannte Subsumptionsrelation sind. Dieses Axiom besagt, dass die Menge der durch den Ausdruck  $C$  beschriebenen Objekte eine Teilmenge der durch den Ausdruck  $D$  beschriebenen Objekte ist. Ein Beispiel für ein Axiom, welches aufgrund seiner Struktur immer wahr ist, ist folgendes:

$$\exists \text{friend.Elephant} \sqsubseteq \exists \text{friend} \quad (4.40)$$

Die Subsumptionsrelation entspricht demnach der Vererbungsrelation, welche, wie bereits mehrfach betont, eine zentrale Rolle bei der Modellierung von Ontologien spielt. Auf formaler Ebene entspricht die Subsumptionsrelation der logischen

Implikation. Dementsprechend können Axiome auf folgende Weise direkt in Prädikatenlogik übersetzt werden:

$$T^x(C \sqsubseteq D) = T^x(C) \rightarrow T^x(D)$$

Im Zuge der praktischen Anwendung von Beschreibungslogiken zur Modellierung von Ontologien haben sich eine Reihe von Axiomen als nützlich erwiesen, die alle Spezialfälle der oben beschriebenen allgemeinen Form sind. Der erste dieser Spezialfälle ist die Definition von Äquivalenzen zwischen Konzepten. Äquivalenzaxiome haben die Form:

$$C \equiv D$$

Sie entsprechen einer Konjunktion von  $C \sqsubseteq D$  und  $D \sqsubseteq C$  und drücken aus, dass die durch die Konzeptausdrücke beschriebenen Objektmengen identisch sind. Die Übersetzung in Prädikatenlogik ergibt sich direkt aus der Reduktion auf die angegebenen Subsumptionsaxiome. Ein Beispiel für ein Äquivalenzaxiom, welches aufgrund der beteiligten Definitionen immer wahr ist, ist folgendes:

$$\leq 0 \text{ friend} \equiv \neg \exists \text{ friend}$$

Subsumption und Äquivalenzaxiome können nun verwendet werden, um die Bedeutung von Konzeptnamen zu definieren. Hierzu werden Axiome der Form  $A \sqsubseteq C$  bzw.  $A \equiv C$  verwendet. Dabei ist  $A$  ein Konzeptname und  $C$  ein Konzeptausdruck. Hierdurch wird dem Konzeptnamen  $A$  eine präzisere Bedeutung zugeordnet. Im Fall eines Axioms der Form  $A \sqsubseteq C$  wird festgelegt, dass alle Objekte des Konzeptes  $A$  die in  $C$  definierten Bedingungen erfüllen müssen (da sie ja eine Teilmenge der durch  $C$  beschriebenen Menge darstellen). Diese Axiomform entspricht also der Definition von *notwendigen Eigenschaften* des Konzeptes (vgl. Abschnitte 1.2 und 2.2). In unserem Beispiel würden

wir das Konzept eines Elefanten und seiner charakteristischen Eigenschaften mit Hilfe entsprechender Axiome wie folgt beschreiben:

$$Elephant \sqsubseteq (= 4legs) \sqcap (= 1trunk) \sqcap \quad (4.41)$$

$$(= 1tail) \sqcap \exists skin - color: \{grey\} \quad (4.42)$$

Dieses Axiom legt also fest, dass alle Elefanten vier Beine, einen Rüssel, einen Schwanz und die Hautfarbe grau haben müssen. Wie bereits im Kontext Logik-basierter Semantik erwähnt, bietet diese Definition keinen Platz für Ausnahmen wie Clyde.

Äquivalenzaxiome der oben erwähnten Form werden verwendet, um notwendige und hinreichende Bedingungen für die Klassenzugehörigkeit zu definieren. Die Semantik der Äquivalenzrelation fordert, dass die Menge der durch einen Konzeptnamen beschriebenen Objekte exakt den durch den entsprechenden Konzeptausdruck spezifizierten entsprechen. Dies bedeutet auch, dass Objekte, welche die angegebenen Bedingungen erfüllen, automatisch zu dem definierten Konzept gerechnet werden können. Als Beispiel können wir das bereits häufiger erwähnte Konzept einsamer Objekte anführen:

$$Lonely \equiv \neg \exists friend$$

Diese Definition besagt, dass alle Objekte, von denen wir zeigen können, dass sie keine Freunde besitzen – es sei nochmal erwähnt, dass es nicht ausreicht, dass keine Freunde explizit angegeben sind – zum Konzept „Lonely“ gehören. Zusätzlich gilt, dass kein Objekt vom Typ „Lonely“ einen Freund haben kann.

Hinreichende Bedingungen spielen bei der Ableitung von Vererbungsrelationen eine wichtige Rolle, wie folgendes Beispiel zeigt. Gegeben seien die beiden Axiome:

$$Lonely \equiv \neg \exists friend$$

$$Lonely - Elephant \equiv \neg \exists friend.Elephant$$

Aus diesen Definitionen können wir ableiten, dass *Lonely – Elephant*  $\sqsubseteq$  *Lonely* gilt. Die Begründung hierfür ist die folgende:

$$Lonely - Elephant \sqsubseteq \neg \exists friend.Elephant \sqsubseteq \neg \exists friend \sqsubseteq Lonely$$

Die Situation ändert sich, wenn wir es stattdessen lediglich mit notwendigen Bedingungen zu tun haben:

$$Lonely \sqsubseteq \neg \exists friend$$

$$Lonely - Elephant \sqsubseteq \neg \exists friend.Elephant$$

Aus diesen Definitionen folgt nicht, dass „Lonely-Elephant“ ein Unterkonzept von „Lonely“ ist. Dies erscheint zunächst nicht intuitiv, wird jedoch klar, wenn man versucht wie oben eine Begründung zu konstruieren:

$$Lonely - Elephant \sqsubseteq \neg \exists friend.Elephant \sqsubseteq \neg \exists friend \sqsupseteq Lonely$$

Es lässt sich zwar ableiten, dass für alle Objekte vom Typ *Lonely-Elephant* die Bedingung  $\neg \exists friend$  gilt. Da diese jedoch nur eine notwendige, nicht aber eine hinreichende Bedingung für das Konzept *Lonely* ist, ist nicht sichergestellt, dass alle Objekte vom Typ *Lonely-Elephant* auch vom Typ *Lonely* sind. Die Bedeutung von hinreichenden Bedingungen bei der Konstruktion von Ontologien werden wir in [Kapitel 5.1](#) noch vertiefen.

Eine weitere Form von Axiomen, die im Kontext der Modellierung von Ontologien eine große Bedeutung besitzt, sind sogenannte domain und range-Restriktionen. Diese legen den Typ von Objekten fest, die in einer Relation auftreten, und

entsprechen demnach im Wesentlichen den RDF-Konstrukten `rdfs:domain` und `rdfs:range`. Diese Konstrukte lassen sich nicht direkt in Beschreibungslogiken ausdrücken, den gewünschten Effekt erhält man jedoch durch die Verwendung der folgenden Axiome:

$$\exists r \sqsubseteq C$$

Dieses Axiom entspricht der Restriktion der domain von Relation  $r$  auf Objekte vom Typ  $C$ , da das Axiom fordert, dass alle Objekte, die mit einem anderen Objekt in der Relation  $r$  stehen (und demnach durch den Ausdruck  $\exists r$  beschrieben werden), auch vom Typ  $C$  sind. Die Einschränkung der range einer Relation lässt sich durch folgendes Axiom darstellen:

$$\top \sqsubseteq \forall r.C$$

Dieses Axiom besagt, dass der Ausdruck  $\forall r.C$  für alle Objekte, die ja durch das Symbol  $\top$  dargestellt werden, gilt. Somit können Objekte, mit denen ein Objekt über die Relation  $r$  verbunden ist, nur vom Typ  $C$  sein.

## Wissensbasen und Inferenzmethoden

Der Vollständigkeit halber wollen wir noch den Begriff der Wissensbasis einführen, der im Kontext von Beschreibungslogiken im Wesentlichen dem einer Ontologie entspricht. Eine Wissensbasis besteht aus einer A-Box  $\mathcal{A}$  und einer T-Box  $\mathcal{T}$ . Wie bereits angedeutet, enthält die A-Box eine Menge von Aussagen der Form  $A(c)$  bzw.  $r(c, d)$ . Hierbei ist  $r$  ein Relationenname,  $C$  ein Konzeptname und  $c$  und  $d$  Objektnamen.<sup>4</sup> Die T-Box besteht

---

<sup>4</sup> Im Allgemeinen sind auch Ausdrücke der Form  $C(c)$  für Konzeptausdrücke erlaubt. Diese lassen sich jedoch in die oben aufgeführte Form



aus einer Menge von Axiomen der Form  $C \sqsubseteq D$ , wobei C und D Konzeptausdrücke sind.<sup>5</sup> Eine Wissensbasis entspricht somit einer prädikatenlogischen Theorie  $\mathcal{P}$ , die wie folgt definiert ist:

$$\mathcal{P} = \left( \bigwedge_{C \sqsubseteq D \in \mathcal{T}} T^x(C) \rightarrow T^x(D) \right) \wedge \left( \bigwedge_{A \in \mathcal{A}} A \right)$$

Eine Aussage folgt aus der Wissensbasis, wenn sie aus  $\mathcal{P}$  folgt. Im Kontext von Beschreibungslogiken beschäftigt man sich in der Regel nur mit der Ableitung von Aussagen, welche Subsumptionbeziehungen und Objekttypen betreffen. Typische Operationen auf einer Wissensbasis sind die folgenden:

**Erfüllbarkeit** Eine Wissensbasis ist erfüllbar, wenn die entsprechende logische Theorie  $\mathcal{P}$  ein Modell besitzt. Unerfüllbarkeit ist oft ein Indiz für Fehler in den Definitionen einzelner Konzepte und wird daher genutzt, um die Qualität einer Ontologie zu überprüfen.

**Konzepterfüllbarkeit** Eine schwächere Form der Erfüllbarkeit ist die sogenannte Konzepterfüllbarkeit. Diese gibt für einen bestimmten Konzeptausdruck an, ob es Objekte geben kann, welche die Definition des Konzeptes erfüllen. Formal ist ein Konzept C erfüllbar, wenn  $\mathcal{P} \wedge C(o)$  ein Modell besitzt; o ist hierbei eine beliebige Konstante. Diese Form der Erfüllbarkeit ist relevant, da auch Theorien, die unerfüllbare Konzepte enthalten, oft erfüllbar sind. Insbesondere ist eine Wissensbasis, die keine A-Box enthält, immer erfüllbar, da stets ein Modell gefunden werden kann, welches den unerfüllbaren Konzepten keine Objekte zuordnet. Die Konzept-

---

bringen, indem ein neuer Konzeptname A eingeführt und das Axiom  $A \equiv C$  in die T-Box eingefügt wird.

<sup>5</sup> Äquivalenzaxiome lassen sich, wie oben beschrieben, auf Paare von Subsumptionsaxiomen zurückführen.

unerfüllbarkeit spielt daher eine noch wesentlichere Rolle bei der Qualitätsüberprüfung einer Ontologie als die allgemeine Unerfüllbarkeit.

**Subsumptionstest** Wie oben bereits angedeutet, ist die Ableitung von Subsumptionsrelationen zwischen Konzeptausdrücken eine Kernfunktionalität von Beschreibungslogiken. Formal lässt sich die Subsumptionsbeziehung  $C \sqsubseteq D$  aus einer Wissensbasis ableiten, wenn  $T^x(C) \rightarrow T^x(D)$  logisch aus  $\mathcal{P}$  folgt. Eine Alternative hierzu ist zu zeigen, dass  $\mathcal{P} \wedge \neg(T^x(C) \rightarrow T^x(D))$  unerfüllbar ist, also kein Modell besitzt. Diese Reduktion auf ein Erfüllbarkeitsproblem lässt sich mit bestehenden Inferenzmaschinen leichter überprüfen und wird daher in der Regel verwendet.

**Klassifikation** Eine direkte Anwendung des Subsumptionstests ist die sogenannte Klassifikation. Diese bezeichnet die Berechnung der Konzepthierarchie aus den Definitionen der Konzepte in der Wissensbasis, wobei in der Regel nur Konzeptnamen berücksichtigt werden. Hierzu wird im Prinzip für jedes Paar von Konzeptnamen überprüft, ob diese in der Subsumptionsrelation zueinander stehen. Da man jedoch nicht an allen Subsumptionsbeziehungen, sondern lediglich an den Beziehungen zu direkten Ober- und Unterkonzepten interessiert ist, kann die Zahl der Tests durch einen sukzessiven Aufbau der Hierarchie reduziert werden. Hierbei werden die Konzepte nach und nach in die bereits ermittelte Hierarchie eingefügt, wobei Tests mit Konzepten in irrelevanten Teilen der Hierarchie ausgelassen werden. Die Klassifikation ist, wie wir noch sehen werden, eine wichtige Funktion bei der Modellierung von Ontologien, da sie dem Benutzer direktes Feedback über die Beziehungen der beschriebenen Konzepte zueinander bietet und oft die Basis für die Darstellung einer Ontologie als Konzepthierarchie bildet.

Neben den oben genannten Funktionen, die sich mit Ausnahme der allgemeinen Erfüllbarkeit auf die Definitionen in der T-Box beziehen, rückt in letzter Zeit verstärkt auch das Problem des Schließens über die Aussagen in der T-Box in den Mittelpunkt des Interesses. Sie bildet die Grundlage für die Beantwortung von Anfragen an Datenbestände, welche auf der Basis von Ontologien beschrieben wurden. In der klassischen Beschreibungslogik wurden in diesem Kontext vor allem die beiden folgenden Probleme untersucht:

**Instanzttest** Wie der Name bereits andeutet, besteht dieses Problem darin zu überprüfen, ob ein Objekt  $o$  Instanz einer bestimmten Konzeptbeschreibung  $C$  ist. Dies ist offensichtlich dann der Fall, wenn  $C(o)$  aus  $\mathcal{P}$  folgt. Analog zum Subsumptionstest kann auch die Unerfüllbarkeit von  $\mathcal{P} \wedge \neg C(o)$  überprüft werden.

**Retrieval** Retrieval beschreibt das Problem der Bestimmung aller Objekte, die Instanzen eines gegebenen Konzeptes  $C$  sind, also die Berechnung der Menge  $\{o | C(o) \text{ folgt aus } \mathcal{P}\}$ . Dies sind neben den explizit als Instanzen von  $C$  beschriebenen Konzepten auch solche Objekte, die aufgrund ihrer Beziehungen zu anderen Objekten die hinreichenden Bedingungen von  $C$  erfüllen. Insbesondere sind dies auch die Instanzen von Konzepten, die in der Subsumptionsrelation zu  $C$  stehen. Ein naives Vorgehen wäre es, den Instanztest für alle Objekte  $o$  durchzuführen, was bei großen A-Boxen zu sehr langen Laufzeiten führt.

**Realisierung** Die Realisierung ist das Gegenstück zum Retrieval. Hier werden ausgehend von einem Objekt  $o$  alle Konzepte bestimmt, von denen  $o$  eine Instanz ist. Die Realisierung beschränkt sich auf Konzeptnamen, da sich stets unendlich viele Konzeptausdrücke finden lassen, die diese Eigenschaft besitzen (insbesondere alle Konzepte der Form  $\top \sqcup C$ , wobei  $C$  ein beliebiger Konzeptausdruck ist).

Sowohl die oben beschriebenen Konstrukte als auch die erwähnten Inferenzmechanismen tauchen in den meisten Systemen zur Verarbeitung von Ontologien auf. Aufgrund abweichender Bezeichnungen und Syntax sowie lückenhafter Dokumentation ist es jedoch nicht immer ganz einfach, diese zu identifizieren. Einige Beispiele hierfür werden wir in [Kapitel 5](#) diskutieren.

### **4.2.2 OWL als Beschreibungslogik**

Die Bedeutung von Beschreibungslogiken als Grundlage für die Darstellung von Ontologien ist in den letzten Jahren enorm gestiegen. Dies ist vor allem auf die Standardisierung der Web Ontology Language (OWL) als Ontologiesprache für das Web zurückzuführen. OWL ist neben dem in [Kapitel 4.1.1](#) vorgestellten RDF-Datenmodell die Grundlage für Entwicklungen im Bereich Semantic Web und wurde hierdurch inzwischen zur meistbenutzten Ontologiesprache aller Zeiten. Aus Sicht des Semantic Web ist OWL eine Erweiterung der RDF-Schema Spezifikation um zusätzliche vordefinierte Relationen, die eine festgelegte Bedeutung besitzen. Aufgrund der Komplexität der in OWL darstellbaren Konstrukte ist es jedoch nicht mehr möglich, die Semantik dieser Konstrukte, wie im Fall von RDF-Schema, in Form einfacher Ableitungsregeln zu definieren. Die Semantik eines OWL Modells lässt sich jedoch durch eine Übersetzung in eine ausdrucksmächtige Beschreibungslogik festlegen.<sup>6</sup> Im Folgenden wird dieser Zusammenhang zwischen OWL und Beschreibungslogiken und dadurch implizit auch mit der Prädikatenlogik erster Stufe kurz eingeführt, ohne genauer auf die

---

<sup>6</sup> Streng genommen gilt dies nur für eine Teilmenge von OWL, nämlich die als OWL DL bezeichnete Untermenge, die wiederum die Teilsprache OWL Lite als Teilmenge enthält.

Besonderheiten von OWL einzugehen, die sich durch die Einbettung in den RDF Sprachstandard ergeben. In diesem Zusammenhang muss zunächst erwähnt werden, dass es für OWL, ähnlich wie bei RDF, unterschiedliche Möglichkeiten der syntaktischen Darstellung des gleichen Modells gibt. Neben der direkten Wiedergabe als RDF-Modell wurde für den hier vorgestellten Teil von OWL auch eine kompaktere Syntax definiert, die sich an der Struktur von Ontologien orientiert. Im Folgenden zeigen wir zunächst diese Syntax und beschreiben dann die Übersetzung von OWL Modellen in Beschreibungslogiken auf der Basis der vorgestellten Sprachkonstrukte, von denen wir die meisten bereits im vorangegangenen Kapitel erläutert haben.

## Ontologien

Ein OWL Modell besteht aus einer Menge von Konzeptdefinitionen und Axiomen und unterscheidet somit im Gegensatz zu Beschreibungslogiken explizit zwischen unterschiedlichen Typen von Axiomen. Kernstück einer Ontologie sind die Konzeptdefinitionen, welche in partielle und vollständige sowie Definitionen durch Aufzählung unterschieden werden. Partielle Klassendefinitionen entsprechen hierbei der Definition eines Konzeptnamens durch ein Subsumptionsaxiom (vgl. Abschnitt 4.2.1). Die Definition der Klasse „Elephant“ aus unserem Beispiel wäre eine partielle Klassendefinition und hätte die folgende Form:

```
Class(Elephant partial(  
  intersectionOf(  
    restriction(legs cardinality(4))  
    restriction(trunk cardinality(1))  
    restriction(tail cardinality(1))  
    restriction(skin-color value(grey))  
  )  
)
```

Das Schlüsselwort `Class` signalisiert hierbei den Beginn einer Klassendefinition, erster Parameter ist der Name der Klasse, der einem Konzeptnamen in Beschreibungslogik entspricht. Das nachfolgende Schlüsselwort `partial` gibt an, dass es sich um eine partielle Klassendefinition handelt und die darin geschachtelten Definitionen, mit denen wir uns später genauer beschäftigen, notwendige Bedingungen der Klasse darstellen. Vollständige Klassendefinitionen werden durch das Schlüsselwort `complete` identifiziert und entsprechen der Definition eines Konzeptnamens durch ein Äquivalenzaxiom. Unsere Beispielklasse `Lonely` würde in OWL wie folgt dargestellt werden:

```
Class(Lonely complete(
    complementOf(
        restriction(friends minCardinality(1))
    )
)
```

Außerdem können, wie bei Beschreibungslogiken, Konzepte durch die Aufzählung der enthaltenen Objekte definiert werden. Das Schlüsselwort hierfür lautet `EnumeratedClass` und die explizite Definition der Klasse aller Freunde von Clyde durch deren Aufzählung würde in OWL wie folgt aussehen:

```
EnumeratedClass(FriendsOfClyde Clement Cleopatra)
```

Neben diesen Konstrukten zur Definition von Konzepten bietet OWL, wie bereits angedeutet, die Möglichkeit, generellere Axiome darzustellen. Allgemeine Axiome, die die Grundlage von Beschreibungslogiken bilden, werden in OWL analog zu RDF durch das Schlüsselwort `SubClassOf` identifiziert, welches auf zwei beliebige Konzeptausdrücke in OWL-Notation angewendet werden kann. Der folgende Ausdruck entspricht [Ausdruck 4.40](#) aus dem vorherigen Kapitel

```
SubClassOf(
  restriction(friends someValuesFrom(Elephant))
  restriction(friends minCardinality(1))
)
```

In ähnlicher Weise können durch die Verwendung des Schlüsselwortes `EquivalentClasses` Äquivalenzrelationen zwischen beliebigen Konzeptausdrücken dargestellt werden, dabei können allerdings direkt mehr als zwei äquivalente Ausdrücke aufgezählt werden, wenn solche existieren. Das folgende Beispiel definiert drei unterschiedliche Formulierungen der Existenz von Freunden als äquivalent. Das Konzept `owl:Thing` steht hierbei für alle denkbaren Objekte und entspricht dem Symbol  $\top$  in Beschreibungslogiken:

```
EquivalentClasses(
  restriction(friends minCardinality(1))
  complementOf(restriction(friends maxCardinality(0)))
  restriction(friends someValuesFrom(owl:Thing))
)
```

Zusätzlich zu diesen aus dem Bereich der Beschreibungslogiken bekannten Axiomen ermöglicht OWL die explizite Beschreibung von Konzepten als disjunkte Mengen von Objekten. Dies bedeutet, dass es kein Objekt geben kann, welches gleichzeitig zu zwei als disjunkt definierten Konzepten gehört. Wie im Fall der äquivalenten Konzepte können beliebig viele Konzepte in einem einzigen Axiom als disjunkt definiert werden. Diese Möglichkeit dient vor allem dazu, nicht näher definierte Konzeptnamen zu differenzieren. Hier ein Beispiel:

```
DisjointClasses(Elephant Trunk Tail Skin)
```

Dieses Axiom beschreibt die offensichtliche Tatsache, dass ein Elefant nicht gleichzeitig ein Rüssel, Schwanz oder eine Haut sein kann und dass ein Rüssel kein Schwanz und keine Haut sein kann usw. Solche offensichtlichen Tatsachen werden bei der Modellierung von Ontologien häufig vergessen und führen aufgrund

der Open World Annahme dazu, dass erwünschte Ableitungen nicht möglich sind.

## Konzeptausdrücke

Wie bereits in den Beispielen im letzten Kapitel zu sehen war, basiert OWL auf der Verwendung komplexer Konzeptausdrücke, die analog zu Beschreibungslogiken durch die Anwendung spezieller Operatoren auf Konzept- und Relationennamen erzeugt werden. Da diese Operatoren bereits diskutierten Beschreibungslogik-Operatoren entsprechen, werden diese hier nicht im Detail diskutiert. Tabelle 4.2 zeigt die Abbildung von OWL-Ausdrücken auf die entsprechenden Elemente einer Beschreibungslogik. Hierzu definieren wir eine Übersetzungsrelation  $V$ , die einen OWL-Ausdruck in einen Konzeptausdruck einer Beschreibungslogik übersetzt.

Eine Besonderheit von Konzeptausdrücken in OWL ist die Unterscheidung zwischen zwei verschiedenen Typen von Relationen, die als Object Properties bzw. Datatype Properties bezeichnet werden. Erstere entsprechen hierbei direkt üblicherweise den in Beschreibungslogik verwendeten Relationen, die Objekte miteinander verbinden. Letztere sind spezielle Relationen, die ein Objekt mit einem konkreten Wert für eine bestimmte Eigenschaft verbinden. Solche konkreten Werte entsprechen den im Kontext von RDF beschriebenen Literalen und werden eindeutig von Objekten unterschieden. Die in der Tabelle beschriebenen Einschränkungen auf Relationen können sowohl auf Object- als auch auf Datatype Properties angewendet werden. Im Fall der `allValuesFrom` und `someValuesFrom` Einschränkung wird dann anstelle einer Konzeptdefinition ein Datentyp und im Fall der `value` Einschränkung anstatt eines Objektes ein konkreter Wert eines geeigneten Datentyps angegeben.



**Tabelle 4.2** Abbildung von OWL-Ausdrücken auf Beschreibungslogiken

OWL Ausdruck D	Übersetzung $V(D)$
owl:Thing	$\top$
owl:Nothing	$\perp$
C (Konzeptname)	C
R (Relationenname)	R
o (Objektname)	o
intersectionOf( $C_1 \dots C_n$ )	$V(C_1) \sqcap \dots \sqcap V(C_n)$
unionOf( $C_1 \dots C_n$ )	$V(C_1) \sqcup \dots \sqcup V(C_n)$
complementOf(C)	$\neg V(C)$
oneOf( $o_1 \dots o_n$ )	$\{o_1, \dots, o_n\}$
restriction(R $r_1 \dots r_n$ )	$V(\text{restriction}(R r_1)) \sqcap \dots \sqcap V(\text{restriction}(R r_n))$
restriction(allValuesFrom(C))	$\forall V(R).V(C)$
restriction(someValuesFrom(C))	$\exists V(R).V(C)$
restriction(R value(o))	$\exists V(R).\{V(o)\}$
restriction(R minCardinality(n))	$\geq n V(R)$
restriction(R maxCardinality(n))	$\leq n V(R)$
restriction(R cardinality(n))	$= n V(R)$

Im OWL Standard ist neben der Verwendung einfacher Datentypen wie String und Integer auch die Möglichkeit der Definition komplexer Datentypen vorgesehen. Dies wird jedoch von den meisten Systemen nur unzureichend oder gar nicht unterstützt und hat daher bisher keine praktische Bedeutung.

## Relationen

Ein Unterschied der Web Ontologie Language zu traditionellen Beschreibungslogiken ist das höhere Maß an Bedeutung, welches den Relationen als wichtiges Element einer Ontologie zukommt. Zwar gab es auch vor der Entwicklung von OWL bereits Arbeiten zur Modellierung zusätzlicher Informationen über

Relationen im Rahmen von Beschreibungslogiken; ins Zentrum des Interesses rückte die Möglichkeit der Definition komplexer Zusammenhänge zwischen Relationen jedoch erst mit der Verwendung dieser Konstrukte in OWL. Ein Grund für die relativ geringe Beachtung der Relationen liegt in der Tatsache, dass bereits in den achtziger Jahren gezeigt wurde, dass bestimmte, die Relationen betreffende Definitionen selbst Logiken mit relativ eingeschränkten Möglichkeiten zur Modellierung von Konzeptausdrücken unentscheidbar machen. Im Rahmen der Definition von OWL und vor allem der kürzlich vorgeschlagenen Erweiterungen OWL 1.1 und OWL 2.0 rückt die Möglichkeit der Modellierung von Relationen wieder stärker in den Vordergrund und soll daher an dieser Stelle beschrieben werden.

OWL erlaubt es insbesondere, Subsumption und Äquivalenz zwischen Relationen darzustellen sowie mathematische Eigenschaften von Relationen, vor allem Symmetrie, Transitivität und Funktionalität. Des Weiteren lässt sich festlegen, dass eine Relation invers zu einer anderen Relation ist. Die Verwendung dieser Möglichkeiten lässt sich an unserem Beispiel zeigen. So können wir beispielsweise eine Relation `Animal-Friends` einführen, die zeigt, mit welchen anderen Tieren ein Tier befreundet ist. Diese Relation könnten wir in OWL wie folgt beschreiben:

```
ObjectProperty(Animal-Friends sub(friends)
               domain(Animal)
               range(Animal)
               Symmetric)
```

Diese Definition legt fest, dass die Relation ein Spezialfall der oben bereits verwendeten Relation `friends` ist, dass diese nur zwischen Objekten vom Typ `Animal` bestehen kann und dass sie symmetrisch ist. Man kann also nicht mit jemandem befreundet sein, der nicht auch mit einem selbst befreundet ist. Letztere Eigenschaft könnte auch als generelle Eigenschaft der allgemeineren Relation `friend` spezifiziert werden und würde

dann an die Relation `Animal-Friends` vererbt werden. Weitere Eigenschaften lassen sich anhand der folgenden Verwandtschaftsrelationen erklären:

```
ObjectProperty(Ancestor inverseOf(Descendant)
               Transitive)

ObjectProperty(Mother super(Ancestor)
               inverseOf(MotherOf)
               Functional)
```

Die `Ancestor` Relation beschreibt das Verhältnis eines Objektes zu seinen Vorfahren im Sinne eines Familienstammbaums. Die inverse Relation hierzu ist die Relation `descendant`, welche Nachkommen beschreibt. Die Relation ist transitiv, da sowohl direkte als auch indirekte Verwandte als Vorfahren bezeichnet werden. Ein besonderes Problem der Transitivität zeigt sich, wenn wir nun `Mother` als Spezialfall der `Ancestor` Relation definieren. Diese Relation, die zwischen einer Person und deren Mutter besteht, ist nämlich nicht transitiv. Dies bedeutet, dass Transitivität nicht notwendigerweise vererbt wird. Bei der Verwendung transitiver Relationen ist daher Vorsicht geboten. Zudem bestehen Einschränkungen hinsichtlich der Verwendung von Kardinalitätseinschränkungen auf transitiven Relationen. Insbesondere dürfen solche Einschränkungen sich nicht auf transitive Relationen oder deren Teilrelationen, die ja wie erwähnt ebenfalls transitiv sind, beziehen. Die `Mother` Relation ist zudem ein Beispiel einer funktionalen Relation, da man nur eine einzige Mutter haben kann. Die inverse Relation `Mother-Of` wird hierdurch automatisch zu einer invers funktionalen Relation, welche ebenfalls eine Eigenschaft ist, die in OWL verwendet werden kann.

Alle diese Eigenschaften haben eine entsprechende Abbildung in ausdrucksmächtigen Beschreibungslogiken. Der Einfachheit halber definieren wir an dieser Stelle die Bedeutung der einzelnen Konstrukte jedoch durch eine direkte Übersetzung in Prädikatenlogik. Diese Übersetzung ist in Tabelle 4.3 dargestellt.

**Tabelle 4.3** Übersetzung der Definition von Relationen in der Web Ontology Language OWL in Prädikatenlogik erster Stufe

OWL Axiom A	Übersetzung in Prädikatenlogik $T^{x,y}(A)$
R (Relationenname)	$R(x, y)$
ObjectProperty(R r1 ... rn)	$T^{x,y}(DatatypeProperty(Rr1) \wedge \dots \wedge T^{x,y}(DatatypeProperty(Rrn)))$
ObjectProperty(R super(S))	$T^{x,y}(R) \rightarrow T^{x,y}(S)$
ObjectProperty(R inverseOf(S))	$T^{x,y}(R) \leftrightarrow T^{y,x}(S)$
ObjectProperty(R Functional)	$T^{x,y}(R) \wedge T^{x,z}(R) \rightarrow y = z$
ObjectProperty(R InverseFunctional)	$T^{x,z}(R) \wedge T^{y,z}(R) \rightarrow x = y$
ObjectProperty(R Symmetric)	$T^{x,y}(R) \leftrightarrow T^{y,x}(R)$
ObjectProperty(R Transitive)	$T^{x,y}(R) \wedge T^{y,z}(R) \rightarrow T^{x,z}(R)$
ObjectProperty(R domain(C))	$T^{x,y}(R) \rightarrow T^y(V(C))$
ObjectProperty(R range(C))	$T^{x,y}(R) \rightarrow T^y(V(C))$
EquivalentProperties(R1 ... Rn)	$T^{x,y}(Ri) \leftrightarrow T^{x,y}(Rj), 1 \leq i, j \leq n$
SubPropertyOf(R1 R2)	$T^{x,y}(R1) \rightarrow T^{x,y}(R2)$

Wie bereits kurz erwähnt, werden diese Definitionen im Kontext von Beschreibungslogiken manchmal auch als R-Box bezeichnet. Die Bedeutung eines OWL-Modells bzw. des entsprechenden Beschreibungslogik-Modells ergibt sich durch eine Übersetzung von T-Box, A-Box und der hier definierten Übersetzung der R-Box in Prädikatenlogik.

## Kapitel 5

# Erstellen von Ontologien

In diesem Buch haben wir uns bisher hauptsächlich mit der Frage beschäftigt, welche Art von Definitionen Teil einer Ontologie sind und wie wir diese so darstellen können, dass eine automatische Verarbeitung möglich ist. Die Existenz entsprechender Sprachstandards wie OWL bietet zwar ein formales Gerüst, an dem sich Entwickler von Ontologien orientieren können, es gibt jedoch keine Antwort auf die Frage, wie man eine gute Ontologie erstellt. In der Tat stellt die Formulierung exakter und korrekter Ontologien noch immer eine der größten Herausforderungen dar, und der hiermit verbundene Aufwand ist eines der Hauptprobleme, die der Verwendung von Ontologien in vielen praktischen Bereichen entgegenstehen. Dies hat mehrere Gründe. Zunächst müssen wir feststellen, dass bei der Erstellung zwei Ebenen eine Rolle spielen. Die konzeptuelle Ebene betrifft die Struktur und Terminologie der beschriebenen Domäne, die formale Ebene deren Darstellung im Rahmen des formalen Modells, also zum Beispiel der Sprache OWL. Ontologiesprachen bieten zwar, wie wir gesehen haben, spezielle Sprachkonstrukte zur Darstellung von Terminologien und

konzeptuellen Strukturen an, die korrekte Verwendung dieser Sprachkonstrukte im Hinblick auf eine adäquate Abbildung der beschriebenen Domäne obliegt jedoch dem Anwender und ist keine einfache Aufgabe. Erschwert wird die richtige Anwendung der Sprachkonstrukte durch die Tatsache, dass es häufig nicht eine korrekte Art der Strukturierung gibt, sondern dass diese von der jeweiligen Perspektive oder geplanten Verwendung der Ontologie abhängt. Um diesem Problem zu begegnen, wurden eine Reihe von Vorgehensmodellen vorgeschlagen, die eine systematische Erfassung der Anforderungen sowie deren Umsetzung in entsprechende Definitionen unterstützen.

In [Kapitel 5.1](#) wird die Modellierung von Ontologien behandelt. Ziel ist nicht die systematische Darstellung der erwähnten Vorgehensmodelle, sondern eher eine pragmatische Anleitung zur Erstellung von Ontologien. Ein spezielles Problem, auf das hier nur kurz eingegangen werden kann, ist die Tatsache, dass es zu Konflikten zwischen den Anforderungen der Domäne und den von der verwendeten Sprache angebotenen Konstrukten kommen kann. Bestimmte wesentliche Elemente der Domäne lassen sich daher nicht direkt in der Sprache ausdrücken. Ein typisches Beispiel hierfür ist die Darstellung der Teil-Ganzes-Relation in Beschreibungslogiken, deren intuitive Bedeutung sich nur unzureichend durch eine transitive Relation erfassen lässt. Das zweite grundlegende Problem bei der Erstellung von Ontologien liegt in der Komplexität der modellierten Zusammenhänge sowohl auf der konzeptuellen als auch der formalen Ebene. Reale Ontologien, wie die in [Kapitel 3](#) dargestellte, bestehen aus Tausenden von Konzepten und Definitionen und sind vom Anwender unmöglich vollständig zu überblicken. Dies ist jedoch streng genommen notwendig, um formale und konzeptuelle Inkonsistenzen bei der Erstellung von Ontologien zu verhindern. Um diesem Problem zu begegnen, wurden eine ganze Reihe von Editoren zur Erstellung von Ontologien entwickelt. Durch graphische

Oberflächen, Benutzerführung, aber auch spezielle Methoden zur Überprüfung von Konsistenz und Detektion von Fehlern unterstützen sie den Benutzer bei der Erstellung von Ontologien und versuchen, die Komplexität der Aufgabe zu reduzieren.

Entsprechende Editoren und deren Funktionalitäten werden in [Kapitel 5.2](#) vorgestellt. Neuere Ansätze, die den modularen Aufbau großer Ontologien aus Teilmodellen unterstützen, können aus Platzgründen nicht im Detail diskutiert werden. Das dritte grundlegende Problem ist die einfache Tatsache, dass die Erstellung einer qualitativ hochwertigen Ontologie trotz fortgeschrittener Editoren und anderer Hilfsmittel ein sehr zeit- und arbeitsintensiver Prozess und somit mit hohen Kosten verbunden ist. Um diesen Aufwand zu reduzieren, wurden in den letzten Jahren verstärkt Methoden untersucht, welche die automatische Generierung von ontologischem Wissen unterstützen. Entsprechende Ansätze können im Rahmen dieses Buches jedoch ebenfalls nicht behandelt werden.

## 5.1 Modellierung von Ontologien

Die Modellierung von Ontologien ist eine komplexe Aufgabe, die sowohl einen guten Einblick in die zu modellierende Domäne als auch ein Verständnis der verwendeten Modellierungssprache voraussetzt. Im Folgenden gehen wir von der Web Ontology Language OWL als Modellierungssprache aus und erläutern das empfohlene Vorgehen bei der Erstellung von OWL-Modellen sowie eine Reihe von Modellierungsmustern, die sich bei der Beschreibung von Ontologien in OWL als nützlich erwiesen haben. Während die beschriebenen Muster spezifisch für OWL bzw. Beschreibungslogiken als Grundlage von Ontologien sind, kann das dargestellte Vorgehen weitgehend auf andere Sprachen übertragen werden.

### **5.1.1 Vorgehen**

Im Gegensatz zum Bereich des Software Engineering, in dem Vorgehensmodelle bereits seit Jahrzehnten eine wichtige Rolle spielen, haben sich im Bereich der Ontologien bisher keine Standard-Vorgehensmodelle herausbilden können. Es ist jedoch allgemein anerkannt, dass die Erstellung von Ontologien ein iterativer Prozess ist, in dem unterschiedliche Teile der Ontologie schrittweise formalisiert werden. Die im Folgenden beschriebenen Schritte sind daher nicht als strenges Ablaufmodell zu verstehen, sondern als Komponenten in einem zyklischen Vorgehensmodell, die mehr oder weniger strikt in der angegebenen Reihenfolge durchlaufen werden sollten. Wie in vergleichbaren Modellen des Software Engineering wird von Feedbackzyklen ausgegangen, in denen zuvor getroffene Modellierungsentscheidungen an neue Anforderungen angepasst oder abgeändert werden, um auftretende Konflikte aufzulösen. Die entsprechenden Schritte, die in einem solchen Prozess durchlaufen werden, sind im Wesentlichen folgende:

#### **Fokussierung des Anwendungsgebiets**

Erster und entscheidender Schritt bei der Erstellung einer Ontologie ist die Definition des abzudeckenden Ausschnittes der Anwendungsdomäne. Hierbei wird nicht nur festgelegt, welche Aspekte eines Anwendungsgebietes in der Ontologie beschrieben werden sollen – und viel wichtiger: welche nicht beschrieben werden sollen –, sondern auch wie detailliert und für welche Anwendung diese Beschreibung ausfallen soll. Diese Fokussierung ist notwendig, um den Aufwand für die Erstellung der Ontologie zu begrenzen, aber auch um Konflikte aufgrund von Unterschieden im Detaillierungsgrad oder in der Sichtweise zu



verhindern. Diese Fokussierung führt zwar häufig später zu Problemen, wenn Ontologien zusammengeführt oder wiederverwendet werden sollen, ist aber aus praktischen Erwägungen in vielen Fällen unerlässlich. Ausnahmen bilden lediglich grundlegende Ontologien wie die in [Kapitel 3.3](#) beschriebene SUMO Ontologie, die explizit anwendungsunabhängig und domänenübergreifend konzipiert sind.

Als probates Mittel zur Fokussierung der Ontologie auf Aspekte, die für eine bestimmte Anwendung relevant sind, hat sich der Einsatz sogenannter Competency Questions erwiesen. Dies sind Fragen, von denen erwartet wird, dass sie sich mit Hilfe der zu erstellenden Ontologie beantworten lassen. Sie entsprechen also zu erwartenden Anfragen, die von der entsprechenden Anwendung an das formale Modell gestellt werden und mit Hilfe üblicher Inferenzmethoden, wie den in [Kapitel 4.2.1](#) beschriebenen, beantwortet werden sollen. Ziel ist es, in dieser Phase eine möglichst vollständige Liste solcher Fragen zu erstellen, die dann später zur Überprüfung der erstellten Ontologie im Hinblick auf die Abdeckung der relevanten Aspekte sowie Korrektheit und Vollständigkeit der gelieferten Antworten verwendet werden können.

Um das Prinzip zu veranschaulichen, nehmen wir das oft verwendete Beispiel einer Wein-Ontologie, die als Grundlage für ein Wein-Empfehlungssystem dienen soll. Typische Competency Questions für eine solche Wein-Ontologie wären zum Beispiel die folgenden:

- Welche Eigenschaften sollten bei der Auswahl eines Weins berücksichtigt werden?
- Ist Bordeaux ein Rotwein oder ein Weißwein?
- Passt ein Cabernet Sauvignon zu Meeresfrüchten ?
- Welche Art von Wein passt am besten zu dunklem Fleisch?
- Welche sind gute Jahrgänge für einen Napa Zinfandel?

- Ändern sich die Eigenschaften eines Weines mit dem Jahrgang?
- ...

Eine Liste solcher Fragen bildet den Ausgangspunkt für die Erstellung einer Ontologie. Im Laufe der Entwicklung können Fragen hinzukommen, aber auch Fragen entfernt werden, die sich als nicht relevant oder in der gegebenen Sprache als nicht umsetzbar herausstellen. Ein Beispiel hierfür ist die Frage nach der Änderung der Eigenschaften eines Weins mit dem Jahrgang. Ihre Beantwortung erfordert streng genommen zeitliches Schließen, welches von den meisten Ontologiesprachen nicht unterstützt wird.

### **Wiederverwendung bestehender Ontologien**

Sobald der Fokus der zu erstellenden Ontologie feststeht, stellt sich als Nächstes die Frage, ob es möglich ist, bereits bestehende Ontologien wiederzuverwenden. Hierbei muss sowohl die Wiederverwendung einer kompletten Ontologie als auch ihrer Teile in Betracht gezogen werden. Die Vorteile einer Wiederverwendung liegen auf der Hand. Zum einen kann der Aufwand der Erstellung erheblich reduziert werden, und zum anderen kann häufig von bestehenden Erfahrungen in der Modellierung bestimmter Domänen profitiert und typische Fehler bei der Modellierung können vermieden werden. Ein ebenfalls nicht zu unterschätzender Vorteil der Wiederverwendung von Ontologien ist die sich hieraus ergebende Einheitlichkeit der Beschreibung einer bestimmten Domäne, welche es einfacher macht, Daten und Informationen zwischen Anwendungen auszutauschen. Aus diesem Grund basiert die grundlegende Idee der Verwendung von Ontologien im Semantic Web auf der Wiederverwendung

von Konzepten durch Verweise auf entsprechende Definitionen, welche die Einheitlichkeit und Interoperabilität der Beschreibungen sicherstellen.

Diesen offensichtlichen Vorteilen stehen erhebliche Probleme gegenüber, die eine Wiederverwendung schwierig machen. So entsprechen bestehende Ontologien in den seltensten Fällen den konkreten Anforderungen einer neuen Anwendung. In diesem Fall muss die bestehende Ontologie entsprechend erweitert und angepasst werden. Oft ist dies aufwändiger als eine Ontologie komplett neu zu erstellen, da die Anpassung einer bestehenden Ontologie zunächst ihre genaue Analyse voraussetzt und potenziell mehr Konflikte auftreten, die gelöst werden müssen. Es muss also abgewogen werden, ob die zu erwartenden Vorteile im Hinblick auf die Standardisierung der Modelle den zusätzlichen Aufwand rechtfertigen. Die in [Kapitel 3](#) beschriebenen Ontologien sind Beispiele von Ontologien, die häufig wiederverwendet werden, da sie in vielen Fällen einen echten Mehrwert bilden. Allerdings erfolgt hier eher eine Anpassung der Anwendung auf die Ontologie als umgekehrt. Ontologieteile, deren Wiederverwendung sich lohnt, sind Modelle, die sich nicht direkt auf eine bestimmte Domäne beziehen, sondern eher grundlegende Konzepte formalisieren. Ein Beispiel hierfür ist die Beschreibung von Maßen und Skalen in SUMO, die als Grundlage für die Definition von konkreten Maßeinheiten in anderen Ontologien eingesetzt werden kann.

Potenzielle Kandidaten für eine Wiederverwendung können auf unterschiedliche Weise identifiziert werden. Die Popularität des Semantic Web und der Sprache OWL spielt hierbei eine wichtige Rolle. Als erste direkte Auswirkung wurden an unterschiedlichen Stellen Bibliotheken von Ontologien in OWL und RDF-Schema eingerichtet, in denen nach relevanten Modellen gesucht werden kann. Diese Bibliotheken, die zum Beispiel im Rahmen des DAML-Projektes und der Entwicklung des Protégé

Tools als OWL-Editor entwickelt wurden, besaßen jedoch zunächst nur eingeschränkte Suchunterstützung. Neuerdings stehen mit Systemen wie Swoogle (<http://swoogle.umbc.edu/>) und Watson (<http://watson.kmi.open.ac.uk/>) dedizierte Suchmaschinen zur Verfügung, die es erlauben, auf der Basis von Begriffen nach Ontologien zu suchen, die diese Begriffe als Konzept- oder Relationennamen enthalten. Zusätzlich zu dieser Möglichkeit gibt es eine Reihe bekannter Ontologien wie die beschriebene SUMO-Ontologie, die aufgrund ihres grundlegenden Charakters dafür ausgelegt sind, wiederverwendet zu werden. Weitere Beispiele solcher Ontologien sind OpenCyc (<http://www.opencyc.org/>) und DOLCE (<http://www.loa-cnr.it/DOLCE.html>). Eine weitere interessante Quelle für die Wiederverwendung von ontologischem Wissen entsteht derzeit durch die Migration von seit langem bestehenden und über Jahre hinweg entwickelten und gepflegten Ordnungssystemen wie Thesauri, von Klassifikationssystemen und bibliographischen Systematiken ins Web, die oft mit deren Darstellung in RDF einhergeht.

### **Identifikation relevanter Begriffe**

Da Ontologien, wie im ersten Teil dieses Buches ausführlich diskutiert, die in einem Anwendungsgebiet verwendete Terminologie beschreibt und mit einem konzeptuellen und formalen Modell unterlegt, bilden diese Begriffe einen guten Ausgangspunkt für die systematische Entwicklung einer Ontologie. Die Identifikation relevanter Begriffe kann hierbei auf der Grundlage ganz unterschiedlicher Quellen erfolgen. Der klassische Ansatz ist die Durchführung von Interviews mit Experten aus der Anwendungsdomäne. Im Kontext des hier beschriebenen Vorgehens würde der Schwerpunkt eines solchen Interviews allerdings

auf der Erstellung der oben erwähnten Competency Questions liegen, da dies, wie beschrieben, gleichzeitig die Fokussierung auf relevante Begriffe unterstützt. Dementsprechend sind die Competency Questions natürlich eine hervorragende Quelle relevanter Terme. Weitere Quellen für relevante Terme sind Dokumente aus bzw. über den Anwendungsbereich sowie bestehende Datenbanken und andere Datenquellen, die im Zusammenhang mit der geplanten Anwendung von Bedeutung sind.

Bei der Analyse textueller Quellen, also primär der Competency Questions und relevanter Dokumente, sind vor allem Substantive von Interesse, da diese potenziell relevante Konzepte beschreiben. Von Interesse sind jedoch auch Verben, die möglicherweise relevante Relationen wiedergeben. Adjektive wiederum können Hinweise auf mögliche Werte charakteristischer Eigenschaften geben und daher für die Definition notwendiger und hinreichender Bedingungen von Konzepten relevant sein. Stehen strukturierte Informationsquellen wie Datenbanken zur Verfügung, so können deren strukturelle Informationen als Grundlage für die Identifikation wichtiger Begriffe dienen. Im Falle einer Datenbank bilden die Namen von Tabellen, deren Spalten sowie die in den Tabellen enthaltenen Werte potenziell relevante Begriffe, die bei der Erstellung der Ontologie berücksichtigt werden sollten. Dies ist insbesondere dann wichtig, wenn die späteren Instanzen der Ontologie zum Teil oder vollständig aus diesen Informationsquellen stammen, was häufig der Fall ist, wenn bereits Informationssysteme vorhanden sind, die jedoch bisher keine Ontologien verwenden.

Im Falle unseres Wein-Beispiels könnten wir neben den oben beschriebenen Competency Questions Informationen aus Wein-Führern sowie Datenbanken von Winzern und Weinhändlern als Grundlage für die Identifikation von Begriffen verwenden. Begriffe, die auf diese Art identifiziert werden könnten, wären unter anderen die folgenden:

- wine, grape, winery, location,
- wine color, wine body, wine flavor, sugar content
- white wine, red wine, Bordeaux wine
- food, seafood, fish, meat, vegetables, cheese

Wie die gezeigten englischsprachigen Begriffe verdeutlichen, ist ein wichtiger Aspekt, der in dieser Phase der Ontologieerstellung berücksichtigt werden muss, die Mehrsprachigkeit, die einer besonderen Behandlung bedarf. Es ist zwar häufig möglich, eine einheitliche konzeptuelle Struktur zu definieren und Mehrsprachigkeit durch die Verwendung unterschiedlicher Namen für Konzepte, Relationen und Instanzen auszudrücken. Es gibt jedoch auch Bereiche, in denen unterschiedliche Sprachen auch konzeptuelle Unterschiede mitbringen, die nicht ohne Weiteres abgebildet werden können.

## **Festlegung der Klassenhierarchie**

Der wesentliche Schritt bei der Formalisierung von Ontologien bildet nun die Abbildung der gesammelten Terme in ein geeignetes formales Modell. Besondere Bedeutung kommt hierbei der Gestaltung der Konzepthierarchie zu, die ja, wie im ersten Teil dieses Buches ausführlich diskutiert, das zentrale Element einer Ontologie bildet. Dementsprechend erfolgt die Festlegung der Konzepthierarchie in der Regel als erster Schritt der Formalisierung. Die entstandene Konzepthierarchie bildet dann die Grundlage für weitere Formalisierungsschritte.

Bei der Erstellung der Konzepthierarchie sind im Prinzip zwei Fragen zu beantworten:

1. Welche Terme repräsentieren Klassen von Objekten?
2. Zwischen welchen Klassen besteht eine Subsumptionsbeziehung?

Die Beantwortung dieser Fragen wird durch die Rückbesinnung auf die in [Kapitel 1](#) vorgestellten Prinzipien von Kategoriensystemen erleichtert. Dort hatten wir Kategorien als Mengen von Objekten mit gemeinsamen Eigenschaften beschrieben. Um die Frage nach der Eignung eines Terms als Konzept zu beantworten, sind also relevante Terme dahingehend zu prüfen, ob sie Mengen von Objekten mit gemeinsamen Eigenschaften beschreiben. Wenn wir also für die Terme „Wein“ und „rot“ entscheiden wollen, ob diese Konzepte beschreiben, können wir zur Überprüfung die folgenden Sätze bilden:

- „Die Menge aller Weine.“
- „Die Menge aller Rote.“

Der erste Satz macht Sinn und ist ein Indiz dafür, dass Wein tatsächlich ein potenzielles Konzept beschreibt. Der zweite Satz ist nicht ohne Weiteres sinnvoll und daher ein Indiz dafür, dass Rot eher nicht für ein Konzept steht. An dieser Stelle muss erwähnt werden, dass es sich hierbei nur um Faustregeln handelt und insbesondere die Frage, ob ein Begriff wie „rot“ als Kategorie aufgefasst werden kann, in der Philosophie nicht unumstritten ist. Für den praktischen Einsatz hat sich das beschriebene Vorgehen in jedem Fall als sinnvoll erwiesen.

Die Frage nach der Existenz von Subsumptionsbeziehungen lässt sich auf ähnliche Weise überprüfen. Wie bereits im Kontext von Kategoriensystemen diskutiert, steht Subsumption für eine Teilmengenbeziehung zwischen den zu den jeweiligen Konzepten gehörenden Objektmengen. Wollen wir zum Beispiel für die als potenzielle Konzepte identifizierten Terme „Bordeaux“ auf der einen sowie „Wein“ und „Weinregion“ auf der anderen Seite bestehende Subsumptionsbeziehungen bestimmen, können wir zur Überprüfung die folgende Sätze bilden:

- „Alle Bordeaux sind Weine.“
- „Alle Bordeaux sind Weinregionen.“

Der erste Satz ist wahr und daher ein Indiz dafür, dass tatsächlich eine Subsumptionsbeziehung zwischen den entsprechenden Konzepten besteht. Bei der Betrachtung des zweiten Satzes fällt auf, dass dieser nicht wahr ist, da Bordeaux zwar eine Weinregion ist, nicht aber die Instanzen des Konzeptes Bordeaux und somit keine Subsumptionsbeziehung besteht. Eine weitere Erkenntnis, die aus der Betrachtung dieses Satzes erwächst ist, dass der Name Bordeaux uneindeutig ist, da das Konzept Weine aus der Region Bordeaux beschreibt. Eine korrektere Bezeichnung, die auch direkt erkennen lässt, dass keine Subsumptionsbeziehung zum Konzept „Weinregion“ vorliegt, wäre „Bordeaux-Wein“. Das Ergebnis dieses Schrittes ist eine Menge von Konzepten, die in eine Subsumptionshierarchie eingegliedert sind und die Grundlage für die nachfolgenden Schritte bilden. Abbildung 5.1 zeigt beispielhaft einen Ausschnitt aus einer Wein-Ontologie, die auf diese Weise entstanden sein könnte.

## **Definition von Relationen**

Die Identifikation relevanter Relationen ist der nächste Schritt, nachdem die Klassenhierarchie festgelegt wurde. Dieser Schritt ist notwendig, da die Kenntnis der zur Verfügung stehenden Relationen Voraussetzung für die Formalisierung der Klassen durch notwendige und hinreichende Bedingungen im nächsten Schritt ist. Grundsätzlich ist das Vorgehen bei der Identifikation von Rollen ähnlich wie bei der Identifikation von Klassen: Es muss für die verbleibenden relevanten Terme entschieden werden, welche für Relationen stehen. Hierfür gibt es allerdings keinen ähnlich intuitiven Test wie für Konzepte und Subsumptionsbeziehungen. Auch finden sich in den relevanten Termen nicht immer Terme für alle Relationen, die für eine vollständige Beschreibung der Konzepte notwendig sind. Ein nützliches





**Abb. 5.1** Teil der Klassenhierarchie einer Wein-Ontologie. Konzepte sind als Ovale dargestellt. Pfeile zwischen Konzepten repräsentieren eine Subsumptionsbeziehung zwischen den Konzepten in Pfeilrichtung

Vorgehen bei der Identifizierung ist hierbei, sich an folgenden Fragen zu orientieren:

1. Welche Eigenschaften haben alle Objekte einer Klasse gemeinsam?

2. Welche Eigenschaften unterscheiden die Objekte einer Klasse von denen anderer Klassen?
3. Welche Eigenschaften unterscheiden die Objekte einer Klasse?

Diese Fragen haben ihren Ursprung wiederum in den in [Kapitel 1.2](#) vorgestellten Prinzipien, da sie sich an den differentiae, also denjenigen Eigenschaften, die Objekte eines Typs unterscheiden, orientieren. Die ersten beiden der genannten Fragen zielen direkt auf diese Eigenschaften ab, da sie danach fragen, was Klassen nach innen gemeinsam haben und was sie nach außen unterscheidet. Zusätzlich erhalten wir so bereits einen Hinweis darauf, bei der Definition welcher Klassen die entsprechende Relation eine Rolle spielt. Im Fall der Klasse Rotwein wäre eine solche Eigenschaft, die alle Objekte der Klasse gemeinsam haben, die Farbe „Rot“, die sie gleichzeitig von den Objekten anderer Klassen – zum Beispiel Weißwein – unterscheidet. Wir können hieraus ableiten, dass die Relation „hat-Farbe“ relevant ist. Hierbei geht man am besten von oben nach unten in der Klassenhierarchie vor, um Redundanzen zu vermeiden. So ist der Aggregatzustand flüssig zwar auch eine Eigenschaft aller Objekte in der Klasse Rotwein, diese haben sie jedoch mit Objekten vieler anderer Klassen gemein. Das bedeutet zwar, dass der Aggregatzustand eine relevante Eigenschaft sein kann, sie spielt aber bei der Definition der Klasse Rotwein eher keine Rolle, da sie normalerweise von einer allgemeineren Klasse vererbt werden würde.

Die letzte Frage behandelt Eigenschaften, die nicht direkt einen definierenden Charakter haben, jedoch für die Beantwortung von Anfragen an die Ontologie und insbesondere den darin beschriebenen Objekten von Interesse ist. Man unterscheidet in diesem Zusammenhang in der Philosophie auch zwischen den folgenden Arten von Eigenschaften:

**Intrinsische Eigenschaften:** sind untrennbar mit einem Objekt verbunden und definieren dieses. Im Falle eines Weines wären dies etwa die Farbe und der Aggregatzustand. Würden sich diese ändern, würde das Objekt seinen Charakter verlieren.

**Extrinsische Eigenschaften:** werden dem Objekt verliehen, und können sich über die Zeit ändern. Für einen Wein wäre dies zum Beispiel der Preis oder die Abfüllungsart. Ändern sich diese Eigenschaften, so hat dies keinen Einfluss auf das Objekt an sich.

Nachdem auf diese Art eine Grundmenge von Relationen identifiziert wurde, muss diese näher beschrieben werden. Hierbei ist zunächst die Beziehung der gefundenen Relationen zu den im Vorfeld identifizierten Klassen zu bestimmen. Dies erfolgt in der Regel durch die in den [Kapiteln 4.1.1](#) und [4.2.1](#) beschriebenen Domain- und Range Einschränkungen. Für jede Relation ist also festzulegen, welche Art von Objekten diese potenziell verbinden kann. Hierbei ist stets die allgemeinste Klasse zu wählen, um nicht unnötigerweise Möglichkeiten auszuschließen. Kommen mehrere Klassen als Domäne oder Range einer Relation in Betracht, so sind die jeweilige Semantik der verwendeten Sprache und leider zum Teil auch die Eigenschaften des verwendeten Tools zu berücksichtigen, da diese zu unerwünschten Effekten führen können. Nennt man in OWL etwa zwei verschiedene Konzepte, so bedeutet dies, dass nur Objekte, die gleichzeitig Instanzen beider Konzepte sind, in Betracht kommen. Sollen Objekte beider Klassen berücksichtigt werden, so muss dies explizit durch eine Disjunktion modelliert werden. Einige Editoren behandeln die Eingabe mehrerer Konzepte jedoch irreführenderweise wie eine Disjunktion.

Wie bei den Konzepten sind auch Subsumptionsbeziehungen zwischen Relationen von Bedeutung. Auch hier kann die

Existenz einer solchen Relation getestet werden, indem ein Satz gebildet wird, der die Natur der Subsumptionsrelation als Teilmengenbeziehung ausdrückt. Allerdings ist es hierbei oft schwieriger zu entscheiden, ob der entsprechende Satz immer wahr ist oder nicht. Wollen wir feststellen, ob „liegt in“ eine Teilrelation der Relation „Teil von“ ist, können wir dies überprüfen, indem wir mit Hilfe der jeweiligen Konzepte, die domain und range „liegt in“ Relation bilden – nehmen wir an, dies wären „Weinregion“ und „Land“ –, den folgenden Satz bilden:

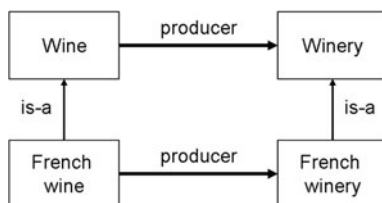
Eine Weinregion, die in einem Land liegt, ist automatisch immer Teil dieses Landes.

In diesem Fall würden wir den Satz als wahr klassifizieren und somit einen Hinweis auf die Existenz einer Subsumptionsbeziehung bekommen. Wie wir schon in [Kapitel 3.1](#) beschrieben haben, ist es vor allem im Hinblick auf die Teil-Ganzes-Beziehung nicht immer leicht zu entscheiden, ob diese zwischen zwei Objekten gilt oder nicht. Weitere Hinweise auf mögliche Fehler bei der Bestimmung von Subsumptionsbeziehungen ergeben sich aus dem Vergleich der jeweiligen Domain- und Range Einschränkungen, da diese für die Teilrelation natürlich jeweils Unterklassen der entsprechenden Konzepte der allgemeineren Relation sein müssen.

## Formalisierung von Klassen

Der aufwändigste und schwierigste Schritt bei der Erstellung einer Ontologie ist die Formalisierung von Klassen und Relationen. Nicht zuletzt deshalb wird dieser Schritt häufig ausgelassen oder nur in sehr beschränktem Umfang durchgeführt. Stattdessen werden sogenannte „light-weight“ Ontologien verwendet. Diese bestehen in der Regel nur aus einer

Konzepthierarchie sowie Relationen, für die jeweils Domain und Range Einschränkungen angegeben werden. Im Gegensatz hierzu stehen sogenannte „heavy-weight Ontologien“, in denen mit Hilfe formaler Logik die Bedeutung von Konzepten exakt spezifiziert wird. Hierzu können zum Beispiel die in [Kapitel 4.2.2](#) beschriebenen Operatoren zur Erstellung von Klassendefinitionen verwendet werden. Ausgangspunkt für die Festlegung dieser Einschränkungen können die im vorangegangenen Abschnitt angestellten Überlegungen zur Beschreibung von Gemeinsamkeiten und Unterschieden der Objekte einer Klasse dienen. Da die Bedeutung der entsprechenden Konstrukte bereits in den [Kapiteln 2.2](#) und [4.2.1](#) beschrieben wurden, soll an dieser Stelle nicht mehr genauer auf die Modellierung konkreter Einschränkungen eingegangen werden.



**Abb. 5.2** Beispiel für das Prinzip der Spezialisierung von Einschränkungen bei Unterklassen

Abbildung [5.2](#) zeigt ein generelles Muster, welches bei der Einschränkung von Eigenschaften zu beachten ist. Analog zur oben beschriebenen Konsistenzbedingung ist hierbei sicherzustellen, dass die in einer Klasse eingeführte Einschränkung mit der der Oberklassen kompatibel ist. Insbesondere ist sicherzustellen, dass der Konzeptausdruck, der mögliche Werte einer Eigenschaft einschränkt, spezieller ist als entsprechende

Einschränkungen, die in Oberkonzepten definiert sind. In der Abbildung wird der Wert der Eigenschaft „producer“ von Objekten des Typs „Wine“ auf das Konzept „winery“ eingeschränkt. Die Unterklasse „French Wine“ schränkt die Eigenschaft „producer“ ebenfalls ein. Da die Einschränkung durch das Konzept „french-winery“ erfolgt, welches eine Unterklasse von „winery“ ist, entsteht hier kein Konflikt. Das Problem der Einschränkungen von Eigenschaften werden wir im Detail nochmals im folgenden Kapitel diskutieren, wo unter anderem typische Muster für die Definition von Wertebereichen beschrieben werden.

Die Verwendung der in [Kapitel 4.2.1](#) beschriebenen Inferenzmethoden kann eine wertvolle Unterstützung in dieser Phase der Formalisierung darstellen, da diese es ermöglichen, Konflikte im erstellten Modell zu finden. Diese äußern sich in der Regel durch unerfüllbare Konzepte, die mithilfe der entsprechenden Methode gefunden werden können. Weitere Hinweise auf mögliche Modellierungsprobleme bietet die Berechnung impliziter Subsumptionsbeziehungen. So kann eine falsche Modellierung zum Beispiel auch dazu führen, dass Subsumptionsbeziehungen zwischen Konzepten gefunden werden, die nicht intendiert sind. In diesem Fall ist die Formalisierung der entsprechenden Konzepte zu überprüfen. Weitergehende Möglichkeiten der Analyse formaler Beschreibungen werden im Zusammenhang mit Ontologie-Editoren in [Kapitel 5.2](#) beschrieben.

### ***5.1.2 Beschreibungsmuster***

Wie bereits erwähnt, haben alle vorgestellten Ontologiesprachen ihre Stärken und Schwächen und erfüllen nicht immer alle Anforderungen des Anwendungsgebietes. In diesem Fall ist es jedoch häufig möglich, den gewünschten Effekt durch eine

geschickte Kombination vorhandener Operatoren zu erreichen. Lange Zeit wurden solche „workarounds“ eher negativ und als Indiz für Schwächen der verwendeten Sprache angesehen. Im Zuge der Standardisierung von OWL als Ontologiesprache für das Semantic Web hat sich diese Einstellung geändert und die Entwicklung sogenannter „ontology patterns“, also Modellierungsmuster für bestimmte Probleme, hat sich zu einem eigenen Forschungsbereich entwickelt. Diese Entwicklung wurde durch die Erkenntnis befördert, dass es für Anwender oft schwer ist, vorhandene Sprachkonstrukte zu verstehen und optimal einzusetzen. Daraus ergibt sich eine weitere Verwendung von Modellierungsmustern im Sinne einer „best practice“ Spezifikation, die Benutzern, aber auch Entwicklern von Editoren Hinweise auf die optimale Darstellung eines bestimmten Aspektes in OWL gibt. Im Folgenden werden einige Modellierungsmuster vorgestellt, die die üblichen Probleme bei der Verwendung von OWL als Ontologiesprache ergeben. Im Kontext des oben beschriebenen Vorgehens bei der Entwicklung von Ontologien würde die Anwendung dieser Muster in der Formalisierungsphase erfolgen, auch wenn sich zwei der Muster auf die Beschreibung von Relationen beziehen. Da es sich bei diesen Mustern jedoch um klassische „workarounds“ handelt, kann ihre Anwendung als Teil der Formalisierung gesehen werden. Bevor wir uns diesen Mustern zuwenden, werden wir zunächst zwei Muster vorstellen, die sich als „best practices“ bei der Formalisierung von Konzepten herausgebildet haben und sich daher direkt auf den entsprechenden Formalisierungsschritt beziehen.

### **Beschreibung von Wertebereichen**

Wie im Verlauf dieses Buches mehrfach dargestellt, werden Klassen in OWL vor allem durch die Beziehung zu anderen

Objekten oder präziser durch die Einschränkung der möglichen Typen von Objekten in bestimmten Relationen beschrieben. Dieses Vorgehen ist so lange intuitiv, wie die entsprechenden Objekte klar identifizierbar sind. Die „friends“-Relation ist ein gutes Beispiel hierfür. Problematischer sind Situationen, in denen die Eigenschaften von Objekten einer Klasse durch die Zuweisung bestimmter konkreter Werte für bestimmte Eigenschaften beschrieben werden sollen. Beispiele hierfür finden sich ebenfalls in Abb. 2.3, nämlich in Form der grauen Hautfarbe eines Elefanten. Da „grau“ nicht ohne weiteres als ein spezielles Objekt der Anwendungsdomäne zu verstehen ist, ist die korrekte Modellierung der Tatsache, dass die Haut von Elefanten grau ist, nicht direkt offensichtlich.

Als offiziellen Mechanismus zur Darstellung konkreter Werte verfügt OWL über die sogenannten Datatype Properties, die Objekte mit konkreten Werten verbinden. Die Verwendung von Datatype Properties ist jedoch nicht immer unproblematisch. Zum einen werden Datentypen von existierenden Systemen nicht immer ausreichend unterstützt, so dass sich auf technischer Ebene Probleme ergeben. Zum anderen erfordert die Definition komplexer Datentypen die Verwendung von XML-Schema-Datentypen, also einer weiteren, nicht logischen Sprache, die auf vollkommen anderen Modellierungsprinzipien basiert und dadurch die Erstellung des Modells erschwert. Aus diesen Gründen ist es sinnvoll, auch bei der Modellierung komplexer Wertebereiche die direkt in OWL vorhandenen Konstrukte zu verwenden.

Prinzipiell bietet OWL zwei Möglichkeiten der Modellierung von Werten und Wertebereichen. Die erste Möglichkeit ist die Einschränkung von Eigenschaften auf konkrete Objekte, die den entsprechenden Wert darstellen. Für unser Beispiel der Hautfarbe wäre die entsprechende Einschränkung die folgende:

```
restriction(skin value(grey))
```



Der entsprechende Wertebereich möglicher Hautfarben würde in diesem Fall durch eine Aufzählung der möglichen Farben dargestellt. Eine entsprechende Definition der Klasse „Hautfarbe“ wäre zum Beispiel folgende:

```
Class(SkinColor complete
      oneOf(grey, pink, red, blue, green))
```

Zusätzlich würde man die entsprechende Relation, in diesem Fall die Relation „skin“, so definieren, dass sie funktional ist und die Aufzählung der relevanten Werte als range hat. Hierdurch wird sichergestellt, dass immer nur ein Wert aus dem Wertebereich zugewiesen wird.

```
ObjectProperty(skin range(SkinColor) Functional)
```

Ein wesentlicher Nachteil dieses Musters zur Beschreibung von Wertebereichen ist die Tatsache, dass sich auf diese Weise Zusammenhänge zwischen den verschiedenen Werten nicht darstellen lassen. Daher eignet sich dieses Muster nur für relativ einfache Wertebereiche, wie etwa das Geschlecht, welches die Werte „male“ und „female“ annehmen kann. Will man jedoch strukturierte Wertebereiche darstellen, ist eine andere Form der Modellierung notwendig. Probleme ergeben sich etwa bei der Unterscheidung verschiedener Grautöne („light-grey“ vs. „dark-grey“), die alle Spezialfälle von Grau sind, oder wenn wir bestimmte Werte zu Gruppen zusammenfassen wollen (zum Beispiel ungesunde Hautfarben, zu denen „red“, „blue“ und „green“ zählen).

Für solche Fälle wurden sogenannte Value-Partitions als Modellierungsmuster für die entsprechenden Wertebereiche vorgeschlagen. Das Prinzip der Value Partition sieht vor, Werte nicht durch konkrete Objekte, sondern durch Unterklassen des Wertebereiches zu definieren. Um unterschiedliche Werte voneinander abzugrenzen wird gefordert, dass die Unterklassen eine Partition

des Wertebereiches bilden, dass sie also paarweise disjunkt sind und den gesamten Wertebereich abdecken. Diese Forderung lässt sich durch entsprechende Axiome in OWL darstellen, wie das folgende Beispiel der Modellierung von Hautfarben als Value-Partition zeigt:

```
DisjointClasses(grey pink red blue green)
EquivalentClass(SkinColor
    UnionOf(grey pink red blue green))
```

Die Beschreibung einer Klasse mithilfe der entsprechenden Werte kann nun durch die folgende Einschränkung realisiert werden:

```
restriction(skin someValuesFrom(grey))
```

Durch die Definition der entsprechenden Klasse als Partition des Wertebereichs wird ein Verhalten erreicht, welches weitestgehend dem der Verwendung konkreter Objekte entspricht. Ein Vorteil dieser Form der Modellierung ist, dass sich der Wertebereich leicht weiter strukturieren und detaillieren lässt. So lässt sich der jetzt als Konzept modellierte Wert „grey“ verfeinern, indem einfach Unterklassen definiert werden, die wiederum Partitionen des zu verfeinernden Wertes bilden. Für den erwähnten Fall der Grautöne ergäben sich folgende zusätzliche Definitionen:

```
DisjointClasses(light-grey dark-grey)
EquivalentClass(grey
    UnionOf(light-grey dark-grey))
```

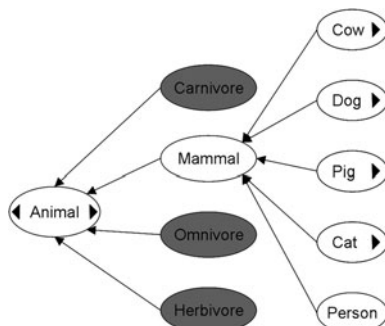
Auf der anderen Seite lassen sich auch die Werte red blue und green zum Wert „unhealthySkinColor“ auf die gleiche Weise zusammenfassen, ohne dass die ursprünglichen Definitionen geändert werden müssen. Die notwendige zusätzliche Definition ist die folgende:

```
EquivalentClass(unhealthySkinColor  
                UnionOf(red blue green))
```

Diese Möglichkeit der Strukturierung ist oft von Vorteil, da sich hierarchische Wertemengen, wie in Abb. 5.2 dargestellt, gut dazu eignen, in einer Hierarchie angeordnete Konzepte konsistent zu definieren.

## Definierte Konzepte und Closure Axiome

Die Verwendung von Konzepten zur Beschreibung von Wertebereichen zeigt, dass nicht alle Konzepte in einem Modell den gleichen konzeptuellen Status besitzen. Insbesondere können wir drei Arten von Konzepten unterscheiden. Neben den im vorangegangenen Kapitel diskutierten Konzepten zur Beschreibung von Werten und Wertebereichen – auch „modifier“ genannt – können wir zusätzlich zwischen zwei Arten von Konzepten unterscheiden, die Mengen von Objekten der Domäne beschreiben. Zum einen sind dies primitive Konzepte. Diese entsprechen im Wesentlichen den Konzepten aus der in [Kapitel 5](#) beschriebenen Klassenhierarchie, die das Rückgrat der zu erstellenden Ontologie bilden. Hierbei wird oft angenommen, dass diese Hierarchie einen Baum bildet, also keine Mehrfachvererbung auftritt. Hinzu kommen definierte Konzepte, die Objekte der Domäne aufgrund ihrer Eigenschaften zusätzlich in – oft orthogonal zur Haupthierarchie verlaufenden – Klassen einordnen. Diese Klassen werden zunächst nicht in die Klassenhierarchie integriert, sondern parallel zu dieser möglichst vollständig definiert. Abbildung 5.3 zeigt dieses Vorgehen am Beispiel einer Tier-Ontologie. Die Basishierarchie unterscheidet zwischen verschiedenen Tierarten (Katzen, Hunde, Schweine etc.). Zusätzlich sind die Klassen



**Abb. 5.3** Klassenhierarchie vor der Formalisierung; definierte Konzepte sind dunkler dargestellt

Fleischfresser, Pflanzenfresser und Allesfresser definiert. Diese sind nicht in die Hierarchie integriert, sondern durch notwendige und hinreichende Bedingungen beschrieben.

Eine Verbindung zwischen den Konzepten der Haupthierarchie und den definierten Konzepten ergibt sich auf der Grundlage der formalen Definitionen aus der Semantik der Ontologie und kann durch geeignete Inferenzmethoden wie den in [Kapitel 4.2.1](#) beschriebenen berechnet werden. Der Vorteil dieser Aufteilung liegt zum einen darin, dass die Struktur der Haupthierarchie relativ einfach bleibt und daher leichter zu erweitern und zu überprüfen ist. Zum anderen kann durch die nachträgliche Klassifizierung der definierten Konzepte in die Basishierarchie die Korrektheit und Vollständigkeit der Definitionen der beteiligten Konzepte überprüft werden. Wird ein Konzept wie Fleischfresser durch die Inferenzmaschine nicht an die Stelle in der Hierarchie eingefügt, an der man es erwarten würde, so sind die Definitionen nicht angemessen.

In unserem Beispiel würden wir jetzt zunächst gemäß dem allgemeinen Vorgehen bei der Erstellung von Ontologien die ver-

wendeten Relationen sowie die Konzepte der Hierarchie genauer beschreiben. Wir beschränken uns in diesem Beispiel auf die Verwendung der Relation „eats“, die wie folgt definiert werden kann:

```
ObjectProperty(eats domain(Animal) range(living_thing))
```

Auf der Grundlage dieser Relation können wir nun sowohl die einfachen Konzepte aus der Hierarchie als auch die definierten Konzepte im Hinblick auf die jeweiligen Essgewohnheiten genauer beschreiben. Die definierten Klassen, die jeweils Tiere mit bestimmten Gewohnheiten beschreiben, können wir in OWL wie folgt darstellen:

```
Class(Herbivore complete(Animal
    restriction(eats allValuesFrom(Plant))))
Class(Carnivore complete (Animal
    restriction(eats allValuesFrom(Animal))))
Class(Omnivore complete(Animal
    restriction(eats someValuesFrom(Plants))
    restriction(eats someValuesFrom(Animal)))
```

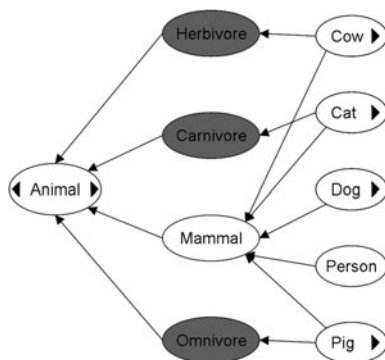
Zusätzlich formalisieren wir die einfachen Konzepte aus der Basis-Hierarchie durch die folgenden OWL-Definitionen:

```
Class(cow partial (Animal
    restriction(eats someValuesFrom(plants))
    restriction(eats allValuesFrom(plants))))
Class(cat partial (Animal
    restriction(eats someValuesFrom(Animal))
    restriction(eats allValuesFrom(Animal)))

Class(pig partial (Animal
    restriction(eats someValuesFrom(plants))
    restriction(eats someValuesFrom(Animal)))
```

Diese Definitionen zeigen auch den Einsatz sogenannter „Closure-Axiome“, einem weiteren wichtigen Modellierungsmuster. Closure Axiome schränken den Wertebereich einer bestimmten Eigenschaft ein. Die Definition des Konzeptes „Cow“

besteht, wie oben dargestellt, zum einen aus der Forderung, dass alle Kühe Pflanzen fressen. Zusätzlich besagt die zweite Einschränkung, dass Kühe ausschließlich Pflanzen fressen. Letztere Definition bildet das Closure Axiom, welches notwendig ist, um abzuleiten, dass es sich bei Kühen um Pflanzenfresser handelt, was nicht alleine durch die Tatsache, dass Kühe Pflanzen fressen gegeben ist. Die hier vorkommende Kombination der beiden Einschränkungstypen ist häufig eine geeignete Form der Modellierung, wenn nicht ganz eindeutig ist, welche Einschränkung die Realität am besten widerspiegelt.



**Abb. 5.4** Klassenhierarchie aus Abb. 5.3 nach der Klassifikation

Abbildung 5.4 zeigt die Klassenhierarchie unseres Beispiels nach der Klassifikation. Wie in der Abbildung gut zu sehen ist, wurden die definierten Klassen in die Hierarchie eingefügt. Katzen wurden korrekt als Fleischfresser, Kühe als Pflanzenfresser

und Schweine als Allesfresser klassifiziert, was ein Indiz, jedoch keine Garantie dafür ist, dass die Modellierung angemessen ist.

## N-stellige Relationen

Eine wesentliche Einschränkung von Beschreibungslogiken ist die Beschränkung auf zweistellige Relationen, während in der Prädikatenlogik Prädikate beliebiger Stelligkeit verwendet werden können, um Relationen zwischen Objekten zu beschreiben. Da auch in der Praxis mehrstellige Relationen auftreten, ist ihre angemessene Darstellung in OWL eine weitere relevante Fragestellung im Hinblick auf Modellierungsmuster. Ein Beispiel für eine mehrstellige Relation ist der Kauf einer bestimmten Ware. Die Objekte, die in einer entsprechenden Relation zueinander stehen, sind neben dem Käufer und dem Verkäufer auch das gekaufte Produkt sowie unter Umständen Randbedingungen des Kaufes wie Kaufpreis und Datum.

Eine sehr direkte und einfache Art, solche Relationen, die sich auf mehr als zwei Objekte beziehen, darzustellen, ist die Modellierung als Klasse. Sie hat für jede Dimension der Relation eine spezielle Eigenschaft, die sie mit den entsprechenden Objekten verbindet. Ein konkreter Verkauf würde entsprechend als Instanz einer Klasse „Verkauf“ modelliert werden und wäre über spezielle Relationen mit den Objekten, die Käufer, Verkäufer, Produkt usw., verbunden. In OWL könnte ein solches Modell wie folgt aussehen:

```
Class(Purchase partial(  
    restriction(buyer someValueFrom(LegalPerson))  
    restriction(seller someValueFrom(LegalPerson))  
    restriction(item someValueFrom(Product))  
    restriction(price someValuesFrom(Real))  
))
```

Ein konkreter Verkaufsvorgang wird als Instanz der oben beschriebenen Klasse dargestellt. Durch die Modellierung als Klasse kann eine solche Instanz nicht mehr als eine einzige Relationsinstanz wiedergegeben werden. Vielmehr ist eine Menge von Relationsinstanzen notwendig, welche die Verkaufsinstanz mit den anderen beteiligten Objekten in Beziehung setzt.

```
Individual(purchase42 type(Purchase)
           value(buyer buyer42)
           value(seller seller42)
           value(item item42)
           value(price price42))
```

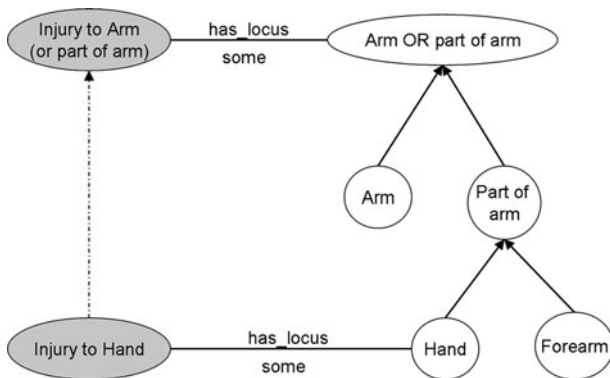
Diese Darstellung bietet im Prinzip alle Möglichkeiten einer Wiedergabe als vierstellige Relation, lediglich Anfragen an das Modell werden komplizierter, da diese die Struktur der Modellierung als Klasse abbilden müssen.

## Teil-Ganzes-Relationen

Wie wir bereits in [Kapitel 3.1](#) festgestellt haben, ist die Teil-Ganzes-Relation aufgrund ihrer unterschiedlichen Varianten schwer formal zu fassen. Da Teil-Ganzes-Relationen jedoch in vielen Anwendungsbereichen wie der Medizin oder der Geografie eine wichtige Rolle spielen, ist die angemessene Darstellung der Beziehungen zwischen Objekten und ihren Teilen in OWL ebenfalls ein Modellierungsproblem, für das Muster vorgeschlagen wurden. In der Regel werden hierbei Teil-Ganzes-Relationen betrachtet, welche die Eigenschaft der Transitivität aufweisen und daher folgende als transitive Relation in OWL modelliert werden. Hierdurch lässt sich etwa das folgende Prinzip ausdrücken: Dadurch, dass der Arm Teil des menschlichen Körpers ist und die Hand ein Teil des Arms, lässt sich korrekterweise ableiten, dass auch die Hand ein Teil des Körpers darstellt.



Ein Problem ist hierbei die Tatsache, dass diese einfache Modellierung der Teil-Ganzes-Relation bestimmte Ableitungen nicht unterstützt. Beispiele für solche gewünschten, jedoch nicht unterstützten Ableitungen, ist etwa die Aussage, dass, obwohl eine Hand kein menschlicher Körper ist, eine Verletzung der Hand auch eine Verletzung des Körpers darstellt; eine in der Medizin häufig benötigte Ableitung. Abbildung 5.5 zeigt eine Möglichkeit zur Modellierung der Teil-Ganzes-Relation, die dieses Problem behebt.



**Abb. 5.5** Modellierung einer anatomischen Teil-Ganzes-Relation

Dieses Muster führt zusätzliche Konzepte ein, die jeweils ein Körperteil sowie seine Teile umfassen. Eine bestimmte Verletzung wird jetzt nicht als Verletzung des entsprechenden Körperteils, sondern als Verletzung des Körperteils oder seiner Teile modelliert. Hierdurch lässt sich, wie in der Abbildung dargestellt, eine Subklassenbeziehung zwischen der Verletzung eines Körperteils und der Verletzung eines anderen Körperteils,

welches ein Teil des erstgenannten darstellt, ableiten. Die Abbildung suggeriert, dass die neuen Konzepte explizit definiert und in die Hierarchie eingefügt werden müssen, was die Übersichtlichkeit erheblich beeinträchtigen würde. Für dieses Problem gibt es zwei Lösungen. Die erste besteht darin, dass die neuen Konzepte gemäß dem oben dargestellten Prinzip der Trennung von primitiven und definierten Konzepten separat als definierte Konzepte dargestellt und lediglich im Rahmen des logischen Schließens zur Laufzeit in die Konzepthierarchie eingefügt werden. Zudem besteht die Möglichkeit, die Vereinigung eines Körperteils und seiner Teile nicht als explizite Klassen, sondern lediglich als komplexe Einschränkung der entsprechenden Eigenschaft einer Verletzung darzustellen. Die Definition der verschiedenen Verletzungen wäre die folgende:

```
Class(InjuryToHand complete(  
  Injury  
  restriction(has_locus unionOf(  
    Hand  
    restriction(is-part-of someValuesFrom(Hand))))))
```

Aufgrund der Komplexität dieser Definition ist es notwendig, den Benutzer bei der Erstellung entsprechender Modelle zu unterstützen. Wie dies geschehen kann, werden wir im folgenden Abschnitt über Ontologie-Editoren diskutieren.

## 5.2 Ontologie-Editoren

Ontologie-Editoren sind Systeme, die den Benutzer beim Erstellen von Ontologien unterstützen. Sie sind mit modernen Textverarbeitungssystemen vergleichbar, die den Benutzer bei der Erstellung von Dokumenten unterstützen: Es besteht zwar die

Möglichkeit, OWL-Ontologien direkt in der auch in diesem Buch verwendeten Sprache auszudrücken, dies würde jedoch der Verwendung eines ASCII Editors bei der Erstellung von Texten entsprechen und ist nur für sehr kleine Modelle, wie die in diesem Buch gezeigten Beispiele, sinnvoll. Größere und komplexere Modelle lassen sich in diesem Fall nur noch schwer überblicken und warten. Ontologie-Editoren unterstützen den Benutzer dabei, auch große und komplexe Modelle zu beherrschen: Sie geben einen Überblick über ein bestehendes Modell in Form geeigneter Visualisierungen, die von den Details des Modells abstrahieren und so eine globale Sicht ermöglichen. Des Weiteren bieten Editoren oft alternative Formen der Manipulation von Definitionen, zum Beispiel als vordefinierte Formulare für bestimmte Beschreibungsmuster, an. Häufig halten Editoren zusätzlich eine Unterstützung durch Methoden des logischen Schließens bereit, wie sie in [Kapitel 4.2.1](#) beschrieben wurden. Eine weitere denkbare Unterstützung bezieht sich auf das in [Kapitel 5.1](#) beschriebene Vorgehen bei der Ontologieerstellung. Eine solche Prozessunterstützung wird jedoch in der Regel nicht angeboten. Der Grund hierfür ist die Tatsache, dass es sich bei dem beschriebenen Vorgehen um einen stark idealisierten Prozess handelt und das Vorgehen in der Realität sehr viel weniger strukturiert ist. Daher beschränken sich bestehende Editoren darauf, die einzelnen Schritte zu unterstützen, ohne dem Benutzer ein bestimmtes Vorgehen vorzuschreiben.

Im Folgenden werden wir die drei genannten Aspekte der Visualisierung, der Manipulation von Definitionen sowie der Inferenzunterstützung, die von den meisten Editoren unterstützt werden, genauer betrachten. Wir orientieren uns hierbei an dem Protégé-System, welches den zurzeit weltweit am meisten benutzten Editor für OWL-Ontologien darstellt und alle genannten Unterstützungen anbietet.

### **5.2.1 Visualisierung von Ontologien**

Die Visualisierung komplexer Ontologie-Modelle ist, wie angedeutet, eine wichtige Funktion von Ontologie-Editoren. Sie soll dem Benutzer einen Überblick über das Gesamtmodell oder zumindest einen Teil des Modells geben. Da die Wiedergabe aller Details den Benutzer überfordern würde, abstrahieren entsprechende Darstellungen vom eigentlichen Modell, indem sie nur bestimmte Elemente einer Ontologie zeigen. Elemente, die dargestellt werden, sind in der Regel Klassen, Relationen und Objekte. Hierbei kann der Benutzer oft wählen, welche Aspekte wiedergegeben werden sollen. Grundlage der Darstellung sind zumeist gerichtete Graphen mit Knoten- und Kantenbezeichnungen. Hierbei werden, wie in semantischen Netzen, Klassen und Objekte in der Regel als Knoten und Relationen als Kanten gezeigt, so dass der Benutzer im Prinzip ein semantisches Netz sieht, hinter dem sich komplexere Definitionen verbergen, die nicht dargestellt werden. Oft können zusätzliche Informationen, vor allem über Klassen, durch die Auswahl des entsprechenden Knotens angezeigt werden. Im Folgenden zeigen wir anhand einiger Beispiele dieses Prinzip der Darstellung unterschiedlicher Aspekte einer Ontologie als Graph.

#### **Klassen- und Relationshierarchien**

Die wohl gebräuchlichste Visualisierung einer Ontologie betrifft die Klassenhierarchie. Diese lässt sich, da sie einen Verband bildet, durch einen gerichteten azyklischen Graphen darstellen. Abbildung 5.6 zeigt die Visualisierung der Klassenhierarchie einer einfachen Ontologie aus dem Bereich des Journalismus. An der Spitze der Hierarchie steht das allgemeine Konzept `OWL:Thing`. Direkte Subklassen, `Person`, `Autor`, `Content` usw. sind durch

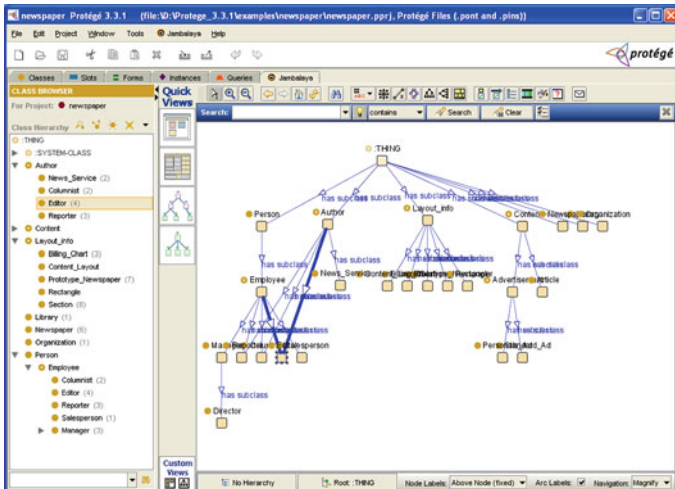


Abb. 5.6 Oberfläche des Protégé-Editors mit Darstellung der Klassenhierarchie mit Hilfe des Jambalaya Plugins

gerichtete Kanten mit dieser Klasse verbunden. Die Namen der Konzepte sind jeweils über dem entsprechenden Knoten dargestellt. In der linken Hälfte der Hierarchie sind Klassen mit mehreren Oberklassen wiedergegeben. Hierbei handelt es sich um Klassen, die spezielle Aufgaben in einem Verlag beschreiben, die jeweils als Unterklasse von „Person“ und von „Autor“ gesehen werden können.

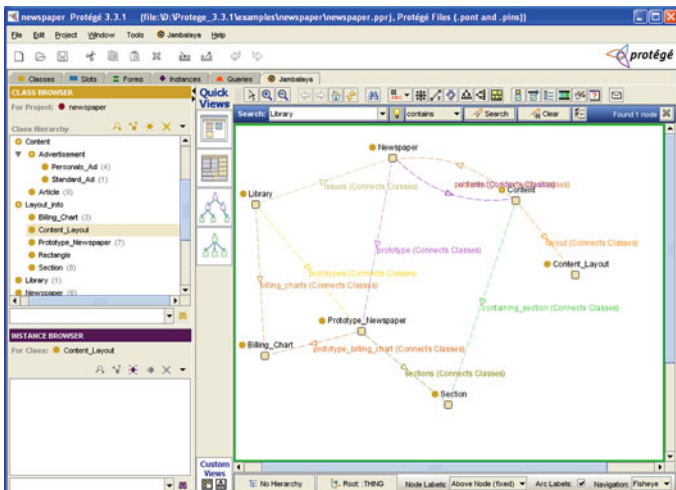
Da Bereiche mit Mehrfachvererbung, wie schon dieses einfache Beispiel zeigt, eine übersichtliche Darstellung der Hierarchie erschweren, wird häufig eine alternative Wiedergabe von Vererbungshierarchien verwendet, die ebenfalls in Abb. 5.6 auf der linken Seite des Bildschirms zu sehen ist. Hierbei wird die Klassenhierarchie, die, wie oben erwähnt, eigentlich ein

gerichteter azyklischer Graph ist, als Baum dargestellt. Dies geschieht, indem Teile der Hierarchie, die auf unterschiedlichen Wegen vom allgemeinsten Konzept aus erreicht werden können, mehrfach im Baum gezeigt werden. Hierbei wird der Baum in Anlehnung an die übliche Darstellung von Ordnern eines Dateisystems in einem vertikalen Menü angezeigt (siehe linke Seite des Bildschirms in Abb. 5.6). In der Abbildung wird dieses Prinzip am Beispiel des Konzeptes Editor deutlich. Wie man sehen kann, ist Editor sowohl als direktes Unterkonzept von „Employee“ als auch von „Author“ im Menü dargestellt. Dies entspricht den beiden im Graph auf der rechten Seite der Abbildung hervorgehobenen Kanten, die zum Editor Knoten hinführen. In diesem Fall besitzt Editor keine weiteren Unterklassen. Diese würden dann auch an beiden Stellen der Baumstruktur erscheinen.

## Domänen-Relationen

Eine alternative Darstellungsform von Ontologien, die häufig verwendet wird, stellt die beschriebenen Relationen in den Vordergrund. Wie im Fall der Vererbungshierarchie werden diese als Kanten eines Graphen dargestellt, in welchem die vorhandenen Klassen die Knoten bilden. Die Verbindung zwischen Knoten und Kanten entspricht hierbei den festgelegten „domain“ und „range“ Einschränkungen in der Definition der entsprechenden Relationen. Das Ergebnis ist ein Graph, der alle Eigenschaften eines semantischen Netzes aufweist. Abbildung 5.7 zeigt einen entsprechenden Graphen für die bereits in Abb. 5.6 dargestellte Ontologie aus dem Bereich des Journalismus.

Die Abbildung zeigt die Relationen, die potenziell zwischen Bibliotheken, Zeitschriften und deren Inhalten bestehen können: Die Tatsache, dass Bibliotheken bestimmte Zeitschriften anbieten, wird durch die Issues Relation dargestellt, während die



**Abb. 5.7** Oberfläche des Protégé-Editors mit Darstellung eines Teils der Relationen zwischen Klassen mit Hilfe des Jambalaya Plugins

Tatsache, dass ein Exemplar einer Zeitschrift vorhanden ist, durch die Relation „prototypes“ zu einer Instanz der Klasse „Prototype-Newspaper“ wiedergegeben wird. Diese ist wiederum durch die prototype Relation mit der Klasse Newspaper verbunden. Solch komplexe Zusammenhänge zwischen der Darstellung einer Zeitung als abstraktes Konzept und der Klasse, welche die Menge der konkreten Zeitschriftenexemplare beschreibt, lassen sich in dieser Sicht gut erkennen.

Wie man an dieser Abbildung außerdem sehr schön sehen kann, ergänzen sich die beiden Visualisierungen, da sie komplementäre Aspekte der jeweiligen Ontologien darstellen. Im Prinzip besteht auch die Möglichkeit, die beiden Visualisierungen zu mischen, indem entweder die Darstellung in Abb. 5.6 um

Domänen-Relationen oder die in Abb. 5.7 um Vererbungsrelationen erweitert wird. Ein solches Mischen der Relationen führt jedoch in der Regel dazu, dass die entstehende Visualisierung sehr unübersichtlich wird, und es sollte daher nur bei sehr kleinen Ontologien verwendet werden. Entsprechende Visualisierungsmechanismen bieten daher oft die Möglichkeit, die Wiedergabe auf ein bestimmtes Konzept zu fokussieren und sie auf die direkte Umgebung dieses Konzeptes einzuschränken. Hierzu wird in der Regel die Länge von Wegen im Graphen angegeben, die bei der Darstellung noch berücksichtigt werden sollen.

## Komplexe Definitionen

Eine der größten Herausforderungen bei der Visualisierung von Ontologien ist die Darstellung komplexer Definitionen, wie etwa Konzeptdefinitionen in Beschreibungslogiken oder gar prädikatenlogische Formeln. Prinzipiell lassen sich logische Formeln bis zu einem bestimmten Grad in Form von Graphen wiedergeben – etwa indem spezielle Knoten eingeführt werden, die logische Operatoren repräsentieren. Prominentes Beispiel hierfür sind sogenannte konzeptuelle Graphen, welche prädikatenlogische Ausdrücke in Form spezieller geschachtelter Graphen darstellen können und zum Teil auch logische Inferenzen auf der Basis von Graphalgorithmen erlauben. Auch für diese Formalismen gilt jedoch, dass sie sehr schnell unübersichtlich werden und sich nur für sehr kleine Probleme eignen.

Insgesamt kann man feststellen, dass sich die graphbasierte Repräsentation logischer Formeln nicht durchgesetzt hat. Statt dessen werden in der Regel quasi-textuelle Darstellungen komplexer Formeln verwendet. Diese wenden oft eine vereinfachte Sprache zur Beschreibung logischer Operatoren an, die sich an einer natürlichsprachlichen Beschreibung der entsprechenden



Operatoren orientieren. Zusätzlich werden diese textuellen Beschreibungen häufig durch einfache grafische Symbole ergänzt, welche die Zuordnung eines vereinfachten Sprachbegriffs zum entsprechenden logischen Operator erleichtern. Der Protégé Editor etwa verwendet seine eigene Sprache zur Darstellung von Konzeptdefinitionen, mit deren Hilfe sich komplexe logische Ausdrücke, die der Sprache OWL entsprechen, beschreiben lassen. Abbildung 5.9 zeigt die quasi-textuelle Darstellung des Konzeptes *Giardiniera*. Hierbei werden die vorhandenen Einschränkungen auf Eigenschaften jeweils in einer Zeile dargestellt. Jeder Zeile ist ein Zeichen vorangestellt, welches die Art der Einschränkung symbolisiert. Logische Operatoren werden durch farblich hervorgehobene Schlüsselworte wiedergegeben. Die Schlüsselworte sind hierbei so gewählt, dass sie sich an umgangssprachlichen Ausdrücken orientieren. So steht etwa der Ausdruck: „hasTopping some Olivetopping“ für die Einschränkung  $\exists \text{hasTopping.OliveTopping}$  sowie „hasTopping only (Leektopping or ... or PeperoniTopping)“ für die Einschränkung  $\forall \text{hasTopping} . (\text{Leektopping} \sqcup \dots \sqcup \text{PeperoniTopping})$ . Hierbei wird zudem zwischen direkt für das angezeigte Konzept spezifizierten und solchen Einschränkungen unterschieden, die von Oberkonzepten vererbt und ebenfalls dargestellt werden, um eine möglichst vollständige Sicht auf die Konzeptdefinition zu erhalten. Dies kann zwar logisches Schließen nicht ersetzen, führt bei eher einfachen Ontologien jedoch häufig zu einer einigermaßen vollständigen Sicht auf ein bestimmtes Konzept.

Dieser Ansatz, komplexe logische Ausdrücke darzustellen, hat zum einen den Vorteil, dass es sich um eine sehr kompakte Repräsentation handelt und die Definition eines Konzeptes in der Regel vollständig dargestellt werden kann, zum anderen bietet die textuelle Darstellung sehr direkte Möglichkeiten zur Interaktion und Manipulation der entsprechenden Definitionen, die im nachfolgenden Kapitel behandelt werden.

### ***5.2.2 Manipulation von Definitionen***

Die Erstellung und Manipulation von Ontologien ist die zentrale Funktion von Ontologie-Editoren und genießt daher naturgemäß besondere Aufmerksamkeit. Im Gegensatz zum bisher diskutierten Problem der Visualisierung bestehen die Anforderungen hier nicht nur darin, dem Benutzer einen guten Überblick über ein Modell zu verschaffen, sondern zusätzlich auch die konkreten Definitionen in einer Form darzustellen, die eine intuitive und effiziente Manipulation ermöglicht. Hiermit eng verknüpft ist die Notwendigkeit, nur Manipulationen zuzulassen, welche die Integrität des Gesamtmodells nicht beeinträchtigen. Dies ist vor allem dann relevant, wenn Elemente aus dem Modell gelöscht werden sollen, die eventuell noch an anderer Stelle verwendet werden. Außerdem ist sicherzustellen, dass eventuell vorhandene unterschiedliche Sichten auf eine Ontologie miteinander synchronisiert sind. Zu diesem Zweck basieren Editoren in der Regel auf einem zentralen Datenmodell, in dem neben den eigentlichen Definitionen der Ontologie zusätzliche Informationen zur Verwaltung und Darstellung der Definitionen gespeichert sind. Die verschiedenen Sichten auf das Modell werden aus diesem gemeinsamen Modell generiert, welches stets konsistent gehalten wird.

Im Folgenden diskutieren wir die gängigsten Ansätze zur Manipulation von Definitionen in Ontologie-Editoren. Dies sind zum einen die Verwendung vorgefertigter Formulare für bestimmte Elemente wie Klassen oder Relationen, zum anderen besteht in Editoren wie Protégé die Möglichkeit, logische Formeln auf der Grundlage einfacher textueller Formate direkt einzugeben. Weit weniger verbreitet ist eine grafische Manipulation von Definitionen. Diese Option werden wir daher nur kurz ansprechen.

## Formulare und Muster

Die üblichste Form der Manipulation von Definitionen ist die Verwendung vordefinierter Formulare für bestimmte Elemente der Ontologie. In der Regel wird zwischen Konzepten, Relationen und Instanzen unterschieden, für die jeweils getrennte Arbeitsbereiche bestehen. Einstiegspunkt ist meist eine baumartige Darstellung der vorhandenen Elemente. In Abb. 5.6 und 5.7 ist jeweils am linken Bildschirmrand eine entsprechende Darstellung der Konzepthierarchie zu sehen. Abbildung 5.8 zeigt Relationen, die in einer Sub-Superrelations-Hierarchie dargestellt werden. Diese Hierarchie dient als interaktiver Einstiegspunkt in die Modellierung. Insbesondere können hier mithilfe von Kontextmenüs Operationen, wie etwa das Einfügen oder Löschen einer Relation bzw. Klasse, auf der bestehenden

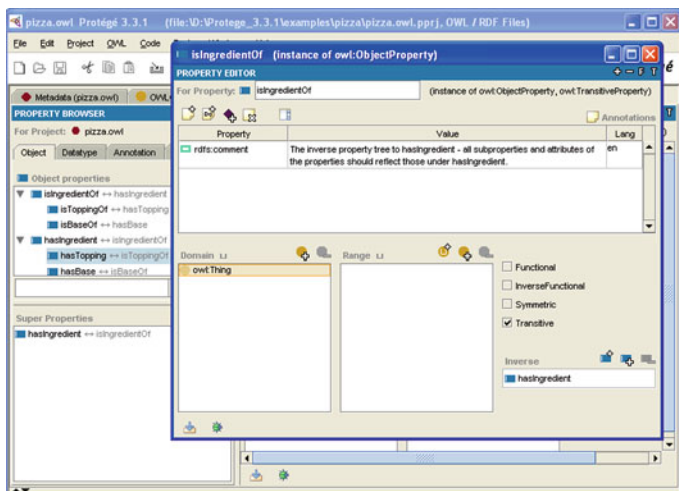


Abb. 5.8 Formular zur Definition einer Relation in Protégé

Hierarchie ausgeführt werden. Das Einfügen eines neuen Elements führt hierbei in der Regel dazu, dass ein spezialisiertes Menü geöffnet wird, in dem das neu zu erstellende Element genauer spezifiziert werden kann. Abbildung 5.8 zeigt als Beispiel die Einfügung einer neuen Relation in eine bestehende Relationshierarchie.

Dieses Beispiel bietet sich an, da sich die meisten Eigenschaften, die eine Relation bestimmen, effizient in einem Formular spezifizieren lassen. Dies können zum einen mögliche allgemeinere Relationen sein, die sich leicht aus einem entsprechenden Menü auswählen lassen, oder aber mathematische Eigenschaften, die man durch das Markieren von Check-Boxen erfassen kann.

Die Verwendung von Formularen eignet sich vor allem auch zur Unterstützung des Benutzers durch Modellierungsmuster wie die in [Kapitel 5.1.2](#) beschriebenen. So verfügt der Protégé Editor über spezielle Formulare zur Spezifikation von Mustern, insbesondere von Wertebereichen und n-stelligen Relationen auf der Grundlage vorhandener Konzepte.

Bei der Definition von Relationen und insbesondere bei der Betrachtung von „domain“ und „range“ Einschränkungen wird allerdings auch ersichtlich, dass sich auf diese Weise nicht beliebig komplexe Ausdrücke modellieren lassen. Einfache Einschränkungen auf eine Menge von Konzepten (die implizit als Disjunktion interpretiert werden) lassen sich noch durch Auswahl aus einem Menü beschreiben, sollen jedoch komplexe Konzeptausdrücke als domain oder range einer Relation gewählt werden, stoßen Formular-basierte Ansätze an ihre Grenzen. Insbesondere die Möglichkeit, Ausdrücke beliebig zu schachteln, ist in diesem Zusammenhang problematisch, da eine solche Schachtelung zwar durch den wiederholten Aufruf von Untermenüs abgebildet werden kann, dies jedoch die meisten Benutzer überfordert. Aus diesem Grund wird hier idealerweise auf

die bereits angesprochene Möglichkeit der direkten Manipulation quasi-textueller Ausdrücke zurückgegriffen.

### **Text-basierte Eingabe**

Vereinfachte textuelle Sprachen wurden oben bereits als Möglichkeit zur kompakten Darstellung komplexer Definitionen vorgestellt. Aufgrund dieser kompakten Darstellung einerseits und der Effizienz der direkten Manipulation von Text andererseits stellt die Text-basierte Eingabe einen idealen Ansatz zur Manipulation komplexer Definitionen dar. Voraussetzung ist jedoch die angemessene Unterstützung von ungeübten Benutzern. Während geübte Benutzer nach einer kurzen Eingewöhnungsphase in der Regel in der Lage sind, Definitionen direkt in der oben vorgestellten vereinfachten OWL-Notation einzugeben, sind für andere zusätzliche Hilfen notwendig. Diese sind zum Beispiel eine automatische Syntaxüberprüfung und Highlighting, die dem Benutzer direktes Feedback hinsichtlich Fehler in der Notation gibt. Zusätzlich bietet Protégé Kontextmenüs an, aus denen oft benötigte Elemente wie bestimmte logische Operatoren und Relationsnamen ausgewählt werden können. Der entsprechende Text wird dann direkt an der aktuellen Cursor Position in die textuelle Beschreibung eingefügt; hierdurch können auch Tippfehler bei der Eingabe reduziert werden.

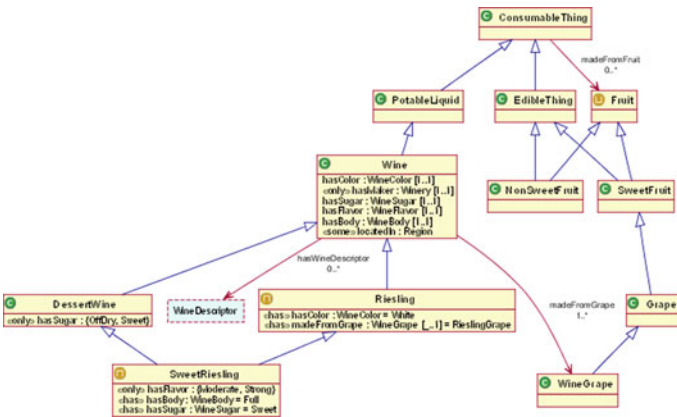
Abbildung 5.9 zeigt, wie ein Teil einer Klassendefinition in Protégé mit Hilfe des Kontextmenüs manipuliert wird. Der manipulierte Teil der Definition der Klasse Pizza „Giardeniere“ ist eine Einschränkung der Eigenschaft `hasTopping`, die fordert, dass eine Beziehung zu einem Objekt vom Typ „MozzarellaTopping“ bestehen muss. Die vereinfachte textuelle Darstellung dieser Einschränkung in Protégé ist: `hasTopping some MozzarellaTopping`. Durch das Anwählen dieser



Protégé jedoch den großen Nachteil, dass der Benutzer mit verschiedenen Darstellungen der gleichen Definition konfrontiert wird. Neben dem logischen Symbol, welches als Icon verwendet wird, muss auch die Bezeichnung von Operatoren in OWL auf der einen und in der vereinfachten Sprache auf der anderen Seite verstanden werden, um effektiv arbeiten zu können. Dies überfordert nicht selten den Benutzer und führt dazu, dass nur in relativ wenigen Ontologien komplexe Definitionen zu finden sind.

## Grafische Manipulation

Eine naheliegende Ergänzung der beschriebenen Interaktionsmöglichkeiten wäre die direkte Manipulation von grafischen Darstellungen einer Ontologie. Diese Option wird in bestehenden Editoren allerdings nur in sehr begrenztem Maße eingesetzt. Am nächsten kommt dieser Option die Möglichkeit, Klassen bzw. Relationshierarchien auf der Basis von Kontextmenüs zu editieren. So können bei Protégé einzelne Elemente in der Darstellung der Konzepthierarchie ausgewählt und bestimmte Operationen wie das Einfügen einer Unterklasse direkt aus einem Kontextmenü ausgewählt werden. Darüber hinausgehende Interaktionsmöglichkeiten, beispielsweise die Definition von Relationen durch drag and drop, sind in bestehenden Editoren nicht realisiert. Es bestehen jedoch Untersuchungen, die darauf abzielen, Ontologien in der Unified Modelling Language UML darzustellen. Dies bietet die Möglichkeit, entsprechende UML-Editoren zu nutzen, die zum Teil weiterführende Möglichkeiten der grafischen Interaktion bieten. Abbildung 5.10 zeigt beispielhaft die Darstellung einer OWL-Ontologie in einem UML-verwandten graphischen Modell, welches in UML zwei Editoren verarbeiten kann.



**Abb. 5.10** Darstellung einer OWL-Ontologie in UML als Grundlage einer grafischen Modellierung

Obwohl die Verbindung zwischen OWL und UML in einer Vielzahl von Arbeiten thematisiert und oft als wichtiger Schritt in Richtung der Benutzerunterstützung angesehen wird, ist unklar, ob eine grafische Modellierung in UML wirklich einen Vorteil gegenüber den beschriebenen Interaktionsmöglichkeiten bietet. UML Werkzeuge, die eine grafische Modellierung unterstützen, stoßen bei größeren Modellen schnell an ihre Grenzen und eignen sich eher für das Fine Tuning einzelner Klassenbeschreibungen als zur Erstellung größerer Modelle.

### 5.2.3 Inferenzunterstützung

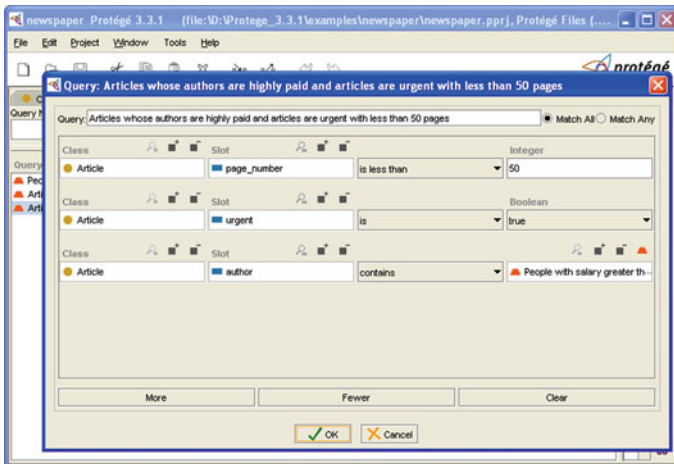
Wie im Zusammenhang mit der Formalisierung von Ontologien schon ausgiebig besprochen, bietet die Verwendung logischer Sprachen als Grundlage für die Modellierung von Ontologien die Möglichkeit, das Erstellen komplexer Modelle durch logische



Inferenzdienste zu unterstützen. Hierbei spielen vor allem zwei Grundprinzipien eine Rolle: die Überprüfung der Vollständigkeit und inhaltliche Korrektheit des Modells durch die Ableitung und Überprüfung von implizitem Wissen und die Prüfung der logischen Konsistenz der Gesamtmodells. Beide Prinzipien, die im Folgenden genauer beschrieben werden, sind Bestandteil üblicher Ontologie-Editoren, sofern diese Logik-basierte Ontologiesprachen wie OWL unterstützen.

### Anfragebearbeitung

Die Möglichkeit, ein bestehendes Modell durch Anfragen zu überprüfen, wurde bereits im Zusammenhang mit dem Vorgehen bei der Erstellung von Ontologien erläutert. So wurden in [Kapitel 5.1.1](#) „Competency Questions“ als Mittel zur Fokussierung des Modells auf einen relevanten Ausschnitt des Anwendungsgebiets vorgestellt. Wurden die entsprechenden Fragen im oben beschriebenen Prozess eher informell genutzt, kann ein Editor die Überprüfung der in den Competency Questions beschriebenen Anforderungen operationalisieren. Zu diesem Zweck verfügen Editoren wie Protégé über eine eigene Sprache, mit deren Hilfe sich Anfragen an das Ontologie-Modell formulieren und auswerten lassen. Ihre Ergebnisse können dann mit den erwarteten Ergebnissen verglichen werden, um die Vollständigkeit und Korrektheit des Modells zu überprüfen. Zu diesem Zweck wurden in der Vergangenheit häufig Anfragemodelle speziell für das in dem entsprechenden Editor verwendete Datenmodell umgesetzt. Mit der fortschreitenden Standardisierung von Ontologien im Zuge des Semantic Web wird seit kurzem häufig die Anfragesprache SPARQL eingesetzt. [Abbildung 5.11](#) zeigt das in den Protégé Editor integrierte Anfrageinterface, in dem einfache konjunktive Anfragen auf der Basis der in der



**Abb. 5.11** Formular zur Erstellung von Anfragen auf Basis der Ontologie in Protégé

Ontologie enthaltenen Konzepte und Relationen formuliert werden können.

In der Regel besteht die Möglichkeit, formulierte Anfragen zu speichern. Dies hat zwei wesentliche Vorteile. Zum einen können hierdurch die zur Überprüfung des Modells verwendeten Competency Questions zu Dokumentationszwecken zusammen mit der Ontologie gespeichert und bei einer späteren Erweiterung oder Veränderung des Modells erneut verwendet werden, um die konsistente Weiterentwicklung zu unterstützen. Zum anderen können gespeicherte Anfragen anstelle von Konzepten in neuen Anfragen verwendet werden. Dies erleichtert die Formulierung komplexerer Anfragen. Abbildung 5.11 zeigt eine solche zusammengesetzte Anfrage. Basierend auf einer Anfrage nach Personen mit einem bestimmten Mindesteinkommen fragt die gezeigte Anfrage nach Zeitungsartikeln, die weniger als 50 Seiten haben, den Status „eilig“ aufweisen und von einem Autoren

geschrieben wurden, der ein Ergebnis der gespeicherten Anfrage ist.

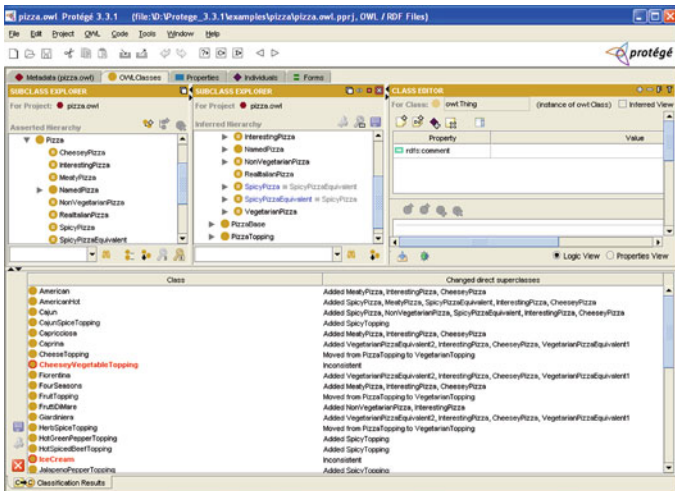
## Klassifikation und Konsistenzprüfung

Die Berechnung der implizierten Klassenhierarchie sowie die Überprüfung der Erfüllbarkeit von Klassendefinitionen sind, wie in [Kapitel 4.2.1](#) beschrieben, typische Inferenzdienste für Beschreibungslogiken. Die enge Beziehung zwischen OWL und diesen Logiken wird von Editoren ausgenutzt, indem die entsprechenden Inferenzdienste als Teil der Funktionalität des Editors angeboten werden. Hierfür wird aus dem im Editor gespeicherten Modell die Wissensbasis einer Beschreibungslogik generiert, die semantisch der modellierten Ontologie entspricht, und an eine externe Inferenzmaschine zur Überprüfung geschickt. Hierfür stehen standardisierte Schnittstellen zur Verfügung.<sup>1</sup> Abbildung 5.12 zeigt das Resultat der Überprüfung einer Ontologie durch eine externe Inferenzmaschine. Es werden weitere gefundene Subsumptionsbeziehungen sowie die Unerfüllbarkeit von Konzeptdefinitionen angezeigt. Oben links werden außerdem die explizit modellierte sowie die durch logisches Schließen berechnete Klassenhierarchie gegenübergestellt. Veränderungen in der berechneten Hierarchie sind hierbei farbig hervorgehoben.

Wie bereits erwähnt, bieten diese Inferenzdienste ebenfalls eine wertvolle Unterstützung für den Benutzer an. Die Berechnung der Konzepthierarchie ermöglicht es, erwartete und tatsächliche Subsumptionsbeziehungen zu vergleichen und hier-

---

<sup>1</sup> Es kommt leider immer wieder vor, dass in der Kommunikation zwischen Editor und Inferenzmaschine Informationen verlorengehen, so dass auf die im Editor gezeigten Ergebnisse nicht immer Verlass ist. Es ist jedoch damit zu rechnen, dass diese Probleme über kurz oder lang behoben werden.



**Abb. 5.12** Ergebnisse der Überprüfung einer Ontologie in Protégé durch eine externe Inferenzmaschine

durch eventuelle Modellierungsprobleme aufzudecken. Zudem ermöglicht sie das in [Kapitel 5.1.2](#) beschriebene Vorgehen der Trennung von atomaren und definierten Konzepten. Unerfüllbare Konzepte sind zudem meist ein Hinweis auf Fehler in der Modellierung und bieten somit ebenfalls Anhaltspunkte für die Verbesserung des Modells.

## Erklärung und Diagnose

Eine Schwachstelle bei der Verwendung von Unerfüllbarkeit als Hinweis auf mögliche Modellierungsprobleme ist die Tatsache, dass der Modellierungsfehler, der die Ursache für die

Unerfüllbarkeit darstellt, sich keineswegs in der Definition des unerfüllbaren Konzeptes befinden muss, sondern an einer beliebigen Stelle im Gesamtmodell. Um den Benutzer bei der Behebung von Fehlern zu unterstützen, ist die Identifikation der Ursache für die Unerfüllbarkeit daher eine entscheidende Funktion. Bei der Identifikation spielen zwei verwandte Probleme eine wichtige Rolle. Dies ist zum einen die Generierung von Erklärungen für abgeleitetes Wissen. Solche Erklärungen lassen sich im Prinzip aus dem logischen Beweis ableiten, indem die am Beweis beteiligten Definitionen in Form eines Ableitungspaths gespeichert und dem Benutzer dann in geeigneter Form präsentiert werden. Abbildung 5.13 zeigt die Darstellung

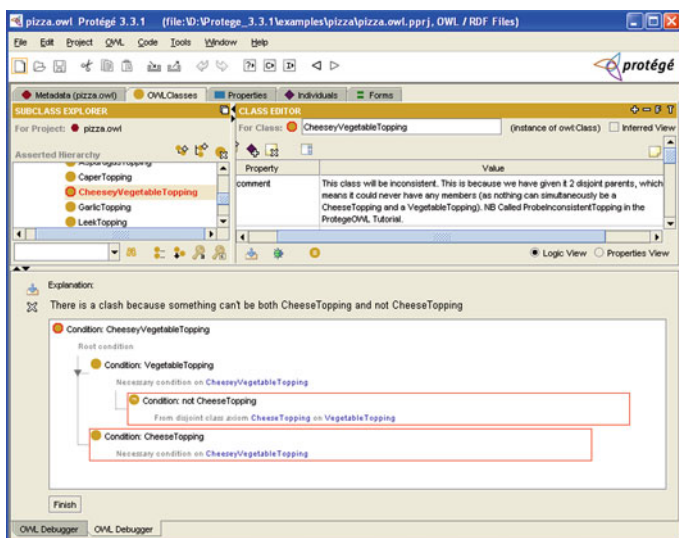


Abb. 5.13 Erklärung für die Unerfüllbarkeitsableitung des Konzeptes „CheesyVegetableTopping“

einer entsprechenden Ableitung der Unerfüllbarkeit des Konzeptes „CheesyVegetableTopping“ in Protégé. Das Problem liegt demnach darin, dass die Instanzen des entsprechenden Konzepts gleichzeitig vom Typ VegetableTopping und CheeseTopping sein müssen. Dies wiederum ist nicht möglich, da diese beiden Konzepte als disjunkt definiert sind. Die Ursache ist also entweder in der Definition des Konzeptes selbst oder aber in der Disjunktheit der Konzepte VegetableTopping und CheeseTopping zu finden.

Solche Erklärungen helfen dem Benutzer nicht nur dabei, die Ontologiesprache und deren Logik besser zu verstehen, sie bilden auch die Grundlage, die Ursache eines Problems zu identifizieren. Letzteres ist im Prinzip ein Diagnoseproblem. Unerfüllbare Konzepte bilden Symptome, deren Ursache gefunden werden sollen. Dies geschieht in der Regel in zwei Schritten. Zunächst werden sogenannte Konfliktmengen für jedes Symptom bestimmt. Diese sind minimale Teilmengen der Definitionen einer Ontologie, die ein Konzept unerfüllbar machen. In den meisten Fällen gibt es mehrere solcher Mengen. Aufgrund ihrer Eigenschaft, minimale Mengen darzustellen, kann die Unerfüllbarkeit des betroffenen Konzeptes dadurch beseitigt werden, dass aus jeder Konfliktmenge mindestens ein Element entfernt wird. Als Diagnose wird nun eine minimale Menge von Elementen bezeichnet, deren Entfernung aus der Ontologie alle unerfüllbaren Konflikte erfüllbar macht. Eine solche Diagnose kann auf der Grundlage der zuvor berechneten Konfliktmengen durchgeführt werden. Der entsprechende Algorithmus ist seit langem bekannt und wurde für den Spezialfall der Diagnose von Ontologien angepasst. Die praktische Anwendung dieser Methode wird durch zwei Probleme erschwert. Zum einen sind sowohl die Bestimmung der Konfliktmengen als auch die Berechnung ihrer Diagnose sehr komplexe Probleme, für die keine effizienten Algorithmen existieren, was vor allem bei großen und

komplexen Ontologien schnell problematisch wird. Zum anderen entspricht die minimale Diagnose nicht immer den tatsächlichen Problemen, so dass die Gefahr besteht, dass korrekte Definitionen aus der Ontologie entfernt werden, falsche jedoch nicht. An beiden Problemen wird im Rahmen aktueller Forschung intensiv gearbeitet.

# **Teil III**

## **Anwendungen**



Das allgemeine Interesse an maschinenlesbaren Ontologien, das zur Entwicklung der im vorangegangenen Teil beschriebenen Methoden und Technologien geführt hat, lässt sich nur dadurch erklären, dass diese einen konkreten Nutzen bei der Verarbeitung von Information darstellen. Er liegt, wie bereits angedeutet, in der Tatsache, dass Tätigkeiten, die traditionell als intellektuelle Aufgabe des Benutzers angesehen wurden, mehr und mehr vom Computer übernommen werden. Je anspruchsvoller die Aufgabe, desto höher sind hierbei die Anforderungen an das Verständnis für die Bedeutung von Informationsinhalten. Ontologien spielen in diesem Zusammenhang als Träger von Bedeutung eine zentrale Rolle. In diesem abschließenden Teil des Buches werden einige typische Anwendungsfelder von Ontologien in der Informationsverarbeitung vorgestellt. Gemeinsames Element ist hierbei die erwähnte Verlagerung der Interpretation von Informationen vom Benutzer in das System. Je nach Art und Struktur der zu interpretierenden Informationen kommen hierbei explizite Repräsentationen lexikalischer oder formaler Semantik zum Tragen. Kapitel 6 beschäftigt sich mit dem verwandten Thema der Datenintegration. Hierbei geht es nicht um den Entwurf neuer, sondern um die nachträgliche Verbindung bestehender Systeme. Das Hauptproblem hierbei ist die korrekte Abbildung und Übertragung versteckter Annahmen über den Kontext und somit die Bedeutung von Informationen in den verschiedenen Systemen. Ontologien dienen hierbei der expliziten Beschreibung dieser individuellen Bedeutung sowie oftmals auch als übergeordnetes Modell, in dem Daten aus unterschiedlichen Systemen vereint werden. In der Informationssuche, die im Kapitel 7 behandelt wird, werden Ontologien ebenfalls zum Zweck der Mediation zwischen unterschiedlichen Darstellungen von Informationen verwendet. Hierbei gilt es nicht, eine gemeinsame Darstellung von Informationen aus unterschiedlichen Systemen zu finden, sondern die Beschreibungen von Benutzerinteressen

(in Form von Anfragen) und Informationen (in Form von Metadaten) miteinander in Beziehung zu setzen. Obwohl die hierbei zu betrachtenden Informationen meist eine einfachere Struktur aufweisen als im Fall der Datenintegration, sind die bestehenden Probleme jedoch meist die gleichen.

# Kapitel 6

## Datenintegration

Eine der wichtigsten Anwendungen von Ontologien in der modernen Informationsverarbeitung ist die semantische Datenintegration. Die Integration von Daten aus heterogenen Quellen zur gemeinsamen Nutzung ist eines der zentralen Probleme der Informationsverarbeitung und tritt in der Praxis in unterschiedlichsten Formen und Zusammenhängen auf. Hierbei treten häufig Probleme auf unterschiedlichen Ebenen auf, welche die Nutzung beeinträchtigen. In der Regel unterscheidet man zwischen den folgenden Arten von Problemen, die bei der Integration auftreten können:

**Syntaktische Konflikte:** Konflikte auf der syntaktischen Ebene entstehen, wenn Informationsquellen unterschiedliche Sprachen bzw. Datenmodelle verwenden, um Informationen darzustellen. So kommt es zum Beispiel in der Praxis häufig vor, dass Informationen aus relationalen Datenbanken mit anderen Formaten wie Tabellenkalkulationen oder Textdateien kombiniert werden sollen.

**Strukturelle Konflikte:** Strukturelle Konflikte ergeben sich, wenn Informationsquellen unterschiedliche Strukturen, also zum Beispiel unterschiedliche Relationen verwenden, um dieselben Daten darzustellen. Dies kann auch dann zu Problemen führen, wenn die Informationsquellen das gleiche Datenmodell, etwa das relationale Modell, verwenden.

**Semantische Konflikte:** Semantische Konflikte entstehen, wenn Informationsquellen unterschiedliche Begriffe verwenden, um die gleichen Elemente eines Anwendungsgebietes zu beschreiben oder die gleichen Begriffe für unterschiedliche Aspekte verwenden. Spezialfälle hiervon sind die Verwendung von Fachausdrücken und Abkürzungen, deren Bedeutung sich nicht direkt erschließt oder Raum für Interpretation lässt. Auch die Verwendung unterschiedlicher Skalen und Maßeinheiten fällt in diesen Bereich.

Zur Lösung syntaktischer Konflikte wurden in der Vergangenheit eine Reihe von Technologien entwickelt, die in der Praxis erfolgreich angewendet werden. Hierzu gehören standardisierte Objektmodelle und Sprachen wie XML, die als gemeinsame Basis für die einheitliche syntaktische Darstellung von Informationen verwendet werden können. In produktiven Systemen werden Informationsquellen unter Verwendung sogenannter „Wrapper“ gekapselt. Sie stellen eine Schnittstelle zur Verfügung, über die auf Daten einer Quelle zugegriffen werden kann als wären diese in einem gemeinsamen Datenformat (zum Beispiel XML oder RDF) gespeichert. Hierdurch entfällt das Problem syntaktischer Konflikte weitestgehend.

Ontologien spielen daher eher bei der Lösung struktureller und semantischer Konflikte in der Datenintegration eine Rolle. Im Folgenden werden zunächst Probleme, die bei der Integration heterogener Datenquellen auf der strukturellen und semantischen Ebene auftreten, diskutiert. Anschließend beschreiben wir

die Rolle von Ontologien bei der Lösung dieser Probleme. Das Kapitel schließt mit einem konkreten Beispiel eines kommerziellen Systems zur Datenintegration auf der Basis von Ontologien.

## 6.1 Heterogenität von Informationsquellen

Um unterschiedliche Konflikte, die bei der Integration von Daten auftreten können, besser zu verstehen, betrachten wir ein konkretes Szenario, in dem Informationen unterschiedlicher Reiseveranstalter integriert werden sollen, um die Buchung von Unterkünften zu erleichtern. Wir gehen davon aus, dass Konflikte auf der syntaktischen Ebene gelöst wurden und die Informationen der unterschiedlichen Quellen in einem gemeinsamen Datenformat vorliegen. Als Datenformat verwenden wir eine objektzentrierte Darstellung, in der Objekte in Klassen eingeteilt sind und durch Attribute mit entsprechenden Werten aus vorgegebenen Wertebereichen beschrieben werden. Diese Darstellung ist sehr allgemein und daher in der Lage unterschiedlichste konkrete Datenmodelle wiederzugeben. Eine relationale Datenbank kann etwa in diesem Modell abgebildet werden, indem die vorhandenen Tabellen als Klassen und die Zeilen einer Tabelle als Objekte der Klasse aufgefasst werden. Die Spalten der Tabelle entsprechen den Attributen mit entsprechenden Wertebereichen. Einträge in der Tabelle stehen dann für die konkreten Werte einzelner Attribute für das durch die Zeile repräsentierte Objekt.

Abbildung 6.1 illustriert die Abbildung zwischen dem verwendeten Datenmodell und Tabellen einer relationalen Datenbank. Es ist jedoch wichtig festzuhalten, dass alle im Folgenden anhand des Objektmodells beschriebenen Konflikte nicht nur in relationalen Datenbanken, sondern auch in anderen Datenmodellen auftreten können, die sich auf eine entsprechende objektzentrierte Darstellung abbilden lassen. Anhand des angenommenen



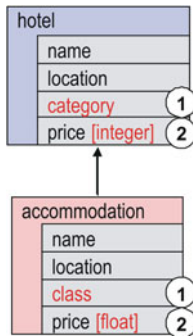
**Abb. 6.1** Abbildung einer relationalen Datenbank auf ein einfaches Objektmodell

Objektmodells lassen sich nun die unterschiedlichen Konflikte, die bei der Integration auftreten können, anschaulich machen. Im Folgenden beschreiben wir häufig vorkommende Konflikte auf der strukturellen und semantischen Ebene anhand des genannten Szenarios einer Integration von Informationen unterschiedlicher Reiseveranstalter.

### 6.1.1 Strukturelle Heterogenität

Strukturelle Konflikte entstehen, wenn auf semantischer Ebene äquivalente Informationen durch unterschiedliche Objektstrukturen dargestellt werden. Dies führt dazu, dass bei der Integration der entsprechenden Quellen unter Umständen relevante Informationen nicht erkannt werden und somit im gemeinsamen Modell nicht zur Verfügung stehen. Hierbei können sich Konflikte aus unterschiedlichen Situationen ergeben, die dazu führen, dass äquivalente Informationen nicht erkannt werden.

Die einfachste Art von Konflikten kann bereits zwischen einzelnen Attributen eines Objektes auftreten. So können Attribute,



**Abb. 6.2** Mögliche Konflikte zwischen einzelnen Datenelementen

die im Prinzip die gleiche Eigenschaft eines Objektes beschreiben, unterschiedliche Namen besitzen. Zudem können sie unterschiedliche Wertebereiche aufweisen. Zusätzliche Konflikte können auftreten, wenn für Attributwerte zusätzliche Bedingungen spezifiziert sind. Beispiele hierfür sind Schlüsselattribute, Default-Werte oder Integritätsbedingungen. Abbildung 6.2 illustriert diese Art einfacher Konflikte am Beispiel von zwei konkreten Objektschemata.

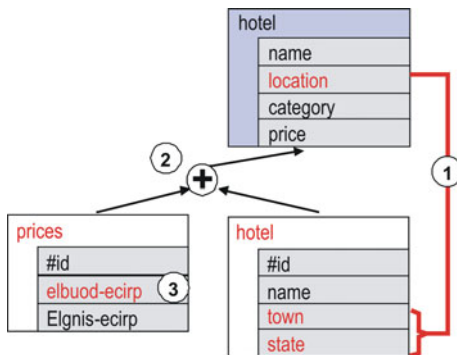
Im Beispiel besteht ein Konflikt zwischen den Eigenschaften, welche die Qualitätskategorie und den Preis eines Hotels bzw. einer Unterkunft im Allgemeinen beschreiben. Während die Kategorie in der einen Informationsquelle mit dem Attribut „category“ beschrieben ist, wird die gleiche Information in der anderen Quelle durch das Attribut „class“ dargestellt. Einen Konflikt zwischen Wertebereichen finden wir im Hinblick auf den Preis, der im Beispiel einmal als „integer“ und einmal als „float“ angegeben wird. Integritätskonflikte können sich in diesem Beispiel ergeben, falls eine der Informationsquellen den Namen der Unterkunft, die andere jedoch die Adresse als Schlüssel verwendet.

Weitere Probleme können entstehen, wenn zusätzliche Integritätsbedingungen angegeben sind. So könnte es zum Beispiel sein, dass in einer der Quellen eine Abhängigkeit zwischen der Kategorie und dem Preis eines Hotels gefordert wird, die in anderen Quellen so nicht besteht.

Komplizierter werden die Probleme, wenn relevante Informationen in mindestens einer der Informationsquellen durch mehr als ein Objekt dargestellt werden. In diesem Fall können nicht nur Unterschiede in der Darstellung einzelner Objekteigenschaften zu Konflikten führen, sondern auch Abweichungen in der Verteilung der Informationen über unterschiedliche Objekte. Probleme entstehen hier zum Beispiel, wenn Informationen, die in einer Quelle als ein einzelnes Attribut dargestellt werden, in einer anderen auf mehrere Attribute verteilt sind. Auf der Ebene ganzer Objekte können ähnliche Situationen auftreten, in denen die Informationen, die in einer Quelle als einzelnes Objekt repräsentiert sind, in einer anderen auf unterschiedliche, über Attribute verbundene Objekte verteilt sind. Außerdem kann es vorkommen, dass bestimmte Informationen zu einem Objekt, die in einer Informationsquelle vorhanden sind, in einer anderen fehlen, so dass Anfragen nach den entsprechenden Informationen an diese Informationsquelle keine Ergebnisse liefern können.

Abbildung 6.3 illustriert diese Art des Konfliktes anhand des Hotel-Beispiels. In dem oben angezeigten Objektschema „Hotel“ wird die Adresse des Objektes in Form eines einzelnen Attributs „location“ dargestellt, während dieselbe Information im unten dargestellten, ebenfalls mit „hotel“ bezeichneten Schema über die zwei Attribute „town“ und „state“ verteilt ist. Außerdem ist in diesem Schema der Preis nicht direkt als Attribut bei dem entsprechenden Objekt gespeichert, sondern in einem eigenen Objekt, welches über eine gemeinsame ID mit dem entsprechenden Hotelobjekt verbunden ist. In diesem Preisobjekt





**Abb. 6.3** Mögliche Konflikte zwischen mehreren Datenelementen

befinden sich zusätzliche Informationen (zum Beispiel über Steueranteile), die in der Darstellung des Preises als einzelnes Attribut nicht vorhanden sind.

### 6.1.2 Semantische Heterogenität

Weitere Probleme bei der Datenintegration sind Unterschiede in der intendierten Interpretation der Daten in einer Informationsquelle. So können Informationen aus unterschiedlichen Quellen selbst dann, wenn sie scheinbar den gleichen Sachverhalt ausdrücken, dennoch eine unterschiedliche Intention haben. Hierdurch kann es bei der Integration mehrerer Quellen zu einer falschen Interpretation bestimmter Informationen kommen, welche das Ergebnis einer Anfrage verfälschen. Dieser Effekt wird auch als semantische Heterogenität bezeichnet und äußert sich auf unterschiedliche Weise. Grundsätzlich können wir zwischen zwei Ebenen von Konflikten unterscheiden, die auf semantische Heterogenität zurückzuführen sind. Zum einen treten Probleme

auf der Ebene von Attributwerten auf, zum anderen auf der Ebene von Konzepten, denen Datenobjekte zugeordnet sind.

Auf der Ebene von Attributwerten kommt es beispielsweise häufig vor, dass numerische Werte mit unterschiedlichen Maßen und Skalen angegeben werden. Je nach gewählter Skala sind solche Unterschiede nicht immer leicht zu sehen. Ein einfaches Beispiel ist die Angabe von Preisen auf der Grundlage unterschiedlicher Währungen. In unserem Hotel-Beispiel kann es vorkommen, dass in einer Datenquelle die Hotelpreise in der Landeswährung angegeben werden. In einer anderen Datenquelle kann das gleiche Angebot in Dollar oder Euro angegeben sein, je nachdem, wo die entsprechende Unterkunft angeboten wird. In Fällen, in denen der Umrechnungskurs deutlich vom Faktor eins abweicht, werden solche Diskrepanzen leicht entdeckt. Ist der Wechselkurs jedoch nahe bei eins, etwa wie der Umrechnungskurs zwischen Euro und Dollar kurz nach der Einführung des Euros, so kann die unterschiedliche Grundlage der Angaben leicht übersehen werden.

Ähnliche Probleme ergeben sich bei der Spezifikation der Hotelkategorie, die auf der Grundlage unterschiedlicher Kriterien erfolgen kann. So entspricht die mit der Kategorie „Drei Sterne“ bezeichnete Kategorie in einem bestimmten Veranstalterkatalog nicht unbedingt der Kategorie „Drei Sterne“ des entsprechenden Landes. Ein wirklicher Vergleich ist in diesem Fall nur auf der Grundlage der tatsächlich angebotenen Leistungen möglich, die nicht unbedingt in der Informationsquelle angegeben sind. Eine besondere Situation ergibt sich hier, wenn keine Eins-zu-Eins-Beziehung zwischen den verwendeten Kategorien möglich ist. So werden in Deutschland Hotelkategorien ja bekanntlich mit einem bis fünf Sternen angegeben, in Frankreich geht die Skala nur bis vier Sterne (Das Hotel „Burj al Arab“ in Dubai gibt sich selbst übrigens sieben Sterne). Auch in diesem Fall ist ein wirklicher Vergleich nur auf der Grundlage der den Kategorien

unterliegenden Kriterien sowie der tatsächlichen Eigenschaften konkreter Hotels möglich.

Eine ähnliche Situation ergibt sich auf der Ebene von Objektklassen, die in unserem Modell ja beispielsweise Tabellen einer Datenbank repräsentieren. Hier verbindet sich mit einer Objektklasse in der Regel eine bestimmte Erwartung an die enthaltenen Objekte. Diese Erwartung kann sich von Datenquelle zu Datenquelle unterscheiden, auch wenn die Klassen auf den ersten Blick so aussehen, als würden sie die gleichen Objekte beschreiben. So ist es möglich, dass es in mehreren Quellen eine Tabelle „Hotels“ gibt, die jeweils als eine Objektklasse dargestellt wird. Es kann nun vorkommen, dass diese Klasse in einer Quelle nur Hotels im üblichen Sinne enthält, während andere Quellen auch Hotel-ähnliche Einrichtungen wie Jugendherbergen, Pensionen oder auch Campingplätze in der entsprechenden Klasse aufführt. In diesem Fall kann die Anfrage nach einem Hotel mit bestimmten Eigenschaften unerwünschte Ergebnisse, nämlich zum Beispiel einen Campingplatz als Ergebnis zurückliefern. Solche Unterschiede hinsichtlich der erwarteten Objektarten in einer Klasse können die folgenden Ausprägungen besitzen:

**Subsumption:** Subsumption ist der einfachste Fall von Unterschieden hinsichtlich der potenziell enthaltenen Objekte. Im Fall des angeführten Beispiels, in dem eine Datenquelle auch Jugendherbergen als Hotels einstuft und eine andere nicht, sind die potenziell enthaltenen Objekte der letztgenannten Quelle eine Teilmenge der potenziell in der erstgenannten enthaltenen.

**Überlappung:** Subsumption ist ein Spezialfall der Überlappung der Objektmengen, in dem eine Objektmenge vollständig in der anderen enthalten ist. Ein allgemeinerer Fall wäre, wenn sich die Objektmengen überschneiden. Dies

geschieht zum Beispiel dann, wenn beide Quellen Hotels im üblichen Sinn beschreiben und in der einen Quelle zusätzlich Jugendherbergen und in der anderen zusätzlich Campingplätze enthalten sind.

**Disjunktheit:** In selteneren Fällen kann es auch vorkommen, dass die Objektmengen von scheinbar äquivalenten Klassen sogar disjunkt sind. Dies kann zum Beispiel vorkommen, wenn in beiden Informationsquellen eine Tabelle „Unterkunft“ besteht, in einer Datenquelle jedoch ausschließlich Jugendherbergen und in der anderen ausschließlich Campingplätze als Einträge vorgesehen sind.

Neben diesen Problemen kann es auch vorkommen, dass sich Informationen widersprechen. Bei einer Integration der Daten ergibt eine Anfrage an das integrierte Modell in diesem Fall unter Umständen sich widersprechende Aussagen. Solche Inkonsistenzen können unterschiedliche Ursachen haben. So ist es zum Beispiel möglich, dass die Daten zu unterschiedlichen Zeitpunkten erhoben wurden und sich der Zustand der Welt inzwischen geändert hat.

## 6.2 Ontologien für die Datenintegration

Ontologien spielen eine wichtige Rolle bei der Unterstützung der Datenintegration und insbesondere bei der Detektion und Behebung der oben genannten Probleme bei der Integration heterogener Datenbestände. Aus diesem Grund sind Ontologien seit den achtziger Jahren Bestandteil vieler Datenintegrationsmethoden. Dabei übernehmen sie im Wesentlichen drei Aufgaben: Zum einen bieten sie eine anwendungsunabhängige und damit neutrale Struktur zur Beschreibung von Informationen in unterschiedlichen Informationsquellen und werden daher oft als

globales Datenmodell verwendet. Zum anderen werden formale Definitionen in Ontologien für implizite Annahmen bezüglich der Interpretation von Informationen aus unterschiedlichen Quellen verwendet. Schließlich können die in diesem Buch vorgestellten Inferenzmethoden für Ontologien eingesetzt werden, um Inkonsistenzen im integrierten Modell zu identifizieren, die zu beheben sind, bevor das integrierte Modell genutzt werden kann. Im Folgenden werden diese drei Funktionen genauer diskutiert.

### ***6.2.1 Neutrale Darstellung von Domänenstrukturen***

Wie in den Beispielen in [Kapitel 6.1.1](#) gezeigt, unterscheiden sich Informationsquellen häufig signifikant bezüglich der Strukturen, wie Informationen über einen bestimmten Anwendungsbereich dargestellt werden. Diese Unterschiede ergeben sich in der Regel aus den unterschiedlichen Anforderungen, die an die Anwendung gestellt werden, für welche die Daten ursprünglich gesammelt und gespeichert wurden. Je nach Sichtweise auf die Daten und Arten der zu erwartenden Anfragen ist die Verwendung unterschiedlicher Strukturen sinnvoll. Sollen solche, speziell für eine Anwendung optimierten Daten integriert werden, so kann dies am besten auf der Basis eines gemeinsamen, neutralen Modells der Anwendungsdomäne geschehen, welches jenseits der spezifischen Anforderungen einer Anwendung relevante Elemente einer Domäne darstellt und dabei die zu integrierenden Aspekte der beteiligten Datenquellen abdeckt. Ontologien, wie wir sie im zweiten Teil dieses Buches kennen gelernt haben, eignen sich hervorragend zur Darstellung eines solchen gemeinsamen Modells.

Der übliche Ansatz der Datenintegration auf der Basis von Ontologien besteht also darin, ein neutrales Modell der betrachteten Domäne zu entwickeln, in dem die konkreten Informationen der einzelnen Quellen dargestellt werden können. Das Vorgehen hierbei entspricht im Wesentlichen dem in [Kapitel 5](#) erläuterten Vorgehen. Die Aufgabe wird einfacher, wenn von vornherein bekannt ist, welche Informationsquellen integriert werden sollen. In diesem Fall dienen die darin enthaltenen Informationen als Referenz für die Vollständigkeit des Modells. Oft ist es jedoch so, dass nicht von Anfang an alle zu integrierenden Informationen bekannt sind, so dass auch in diesem Fall ein gewisses Maß an Allgemeingültigkeit notwendig ist, um diese unbekannten Quellen abdecken zu können. Häufig ist auch eine Überarbeitung der Ontologie im Nachhinein notwendig. Nehmen wir als Beispiel die in [Abb. 6.2](#) und [6.3](#) dargestellten Objektmodelle für Unterkünfte. Um diese unterschiedlichen Modelle zu integrieren, würden wir in diesem Fall zunächst die in den verschiedenen Quellen enthaltenen Objektklassen als Konzepte in eine neue Ontologie übernehmen. Um Klassen aus unterschiedlichen Quellen mit gleichem Namen zu unterscheiden, werden hierbei in der Regel Namenszusätze vergeben. In unserem Beispiel ergeben sich hierdurch zunächst die folgenden Konzepte:

- Quelle1-Hotel
- Quelle2-Accomodation
- Quelle3-Hotel
- Quelle3-Preis

In einem nächsten Schritt werden nun neutrale Klassen eingeführt, über welche der Zugriff auf die integrierten Informationen erfolgen soll. Um sicherzustellen, dass die entsprechenden Objekte aus den unterschiedlichen Quellen auch als Antworten auf Anfragen an dieses neutrale Modell erscheinen, werden die

neutralen Konzepte als Oberkonzepte dieser quellenspezifischen Klassen modelliert. In unserem Beispiel ergäbe sich folgendes initiales OWL-Modell:

```
Class(Quelle1-Hotel)
Class(Quelle2-Accommodation)
Class(Quelle3-Hotel)
Class(Quelle3-Preis)
SubClassOf(Hotel Accommodation)
SubClassOf(Quelle2-Accommodation Accommodation)
SubClassOf(Quelle1-Hotel Hotel)
SubClassOf(Quelle3-Hotel Hotel)
SubClassOf(Quelle3-Preis Preis)
```

Dieses initiale Modell lässt noch alle Freiheiten bezüglich möglicher semantischer Konflikte zwischen den unterschiedlichen Hotel-Konzepten, da die entsprechenden Konzepte Quelle1-Hotel und Quelle3-Hotel theoretisch sowohl äquivalent als auch überlappend oder disjunkt sein können. Eine Festlegung, die bereits getroffen wurde, ist, dass die Klasse Quelle2-Accommodation nicht spezieller sein kann als die Klassen Quelle1-Hotel und Quelle3-Hotel. Diese Festlegung haben wir hier aus dem allgemeingültigen Verständnis der Relation zwischen den Begriffen „Hotel“ und „Accommodation“ hergeleitet. Es muss jedoch auch erwähnt werden, dass es Fälle geben kann, in denen das tatsächliche Verhältnis zwischen Klassen diesem Verständnis zuwiderläuft.

In ähnlicher Weise werden nun in einem zweiten Schritt die Attribute der beteiligten Objektklassen auf geeignete Relationen in der gemeinsamen Ontologie abgebildet. Aus Platzgründen beschränken wir uns hier auf die Attribute der in Abb. 6.2 dargestellten Klassen. Hierbei werden idealerweise alle Relationen zunächst als „Object Properties“ definiert. Dies eröffnet die Möglichkeit, Attributwerte in einem komplexen Objekt darzustellen, was häufig von Vorteil ist, wenn diese in den Quellen unterschiedlich strukturiert sind. Durch die Übernahme dieser

Attribute und die Definition neutraler Relationen ergibt sich folgendes Modell:

```
ObjectProperty(Quelle1-Name
    domain(Quelle1-Hotel) super(Name))

ObjectProperty(Quelle2-Name
    domain(Quelle2-Accommodation) super(Name))

ObjectProperty(Quelle1-location
    domain(Quelle1-Accommodation) super(has-Location))

ObjectProperty(Quelle2-location
    domain(Quelle1-Hotel) super(has-Location))

ObjectProperty(Quelle1-class
    domain(Quelle1-Hotel) super(Rating))

ObjectProperty(Quelle2-category
    domain(Quelle2-Accommodation) super(Rating))

ObjectProperty(Quelle1-Price
    domain(Quelle1-Hotel) super(Price))

ObjectProperty(Quelle2-Price
    domain(Quelle1-Hotel) super(Price))
```

Als nächster Schritt folgt die Modellierung von Attributwerten als komplexe Objekte. Ein gutes Beispiel hierfür ist der Preis, der ja auch in der dritten Informationsquelle als ein solches Objekt dargestellt war. Hierzu werden eine Reihe von Relationen eingeführt, welche das Konzept „Price“ als Domäne haben und dieses mit den unterschiedlichen, in den verschiedenen Quellen vorkommenden Attributwerten, die einen Bezug zum Preis haben, verbinden. Je nachdem ob der entsprechende Wert ein konkreter Datenwert oder aber eine Referenz ist, werden hierzu Object- oder Datatype Properties verwendet. In unserem Fall könnten dies zum Beispiel Relationen für Brutto- und Nettopreis sein:



```
DatatypeProperty(net-price domain(Price))  
DatatypeProperty(gross-price domain(Price))
```

Diese Unterscheidung verschiedener möglicher Interpretationen der Attributwerte macht es möglich, die Werte unterschiedlicher Quellen der jeweils korrekten Interpretation zuzuordnen und die Werte einer Datenquelle der jeweiligen Relation als Wert zuzuweisen. Um die Einheitlichkeit der Darstellung zu gewährleisten, ist es zusätzlich sinnvoll, wenn immer dies möglich ist, die fehlenden Informationen aus den vorhandenen abzuleiten. Hierzu ist allerdings oft Hintergrundwissen – in unserem Fall der Mehrwertsteuersatz – erforderlich. Außerdem sind bei der Übertragung von Werten aus den Quellen in das gemeinsame Modell Konflikte auf der Ebene von Datentypen zu beseitigen, was jedoch oft durch eine einfache Transformation möglich ist.

### ***6.2.2 Explizite Darstellung von Annahmen***

Ein weiterer Vorteil ein integriertes Modell als Ontologie darzustellen ist die Möglichkeit, Annahmen, die von einzelnen Informationsquellen getroffen werden, im gemeinsamen Modell explizit wiederzugeben und zu verwenden, um das Verhältnis zwischen Inhalten aus diesen Quellen klar zu machen. Ein gutes Beispiel hierfür ist das Rating einer Unterkunft, welches ja schon im Zusammenhang mit Problemen bei der Datenintegration diskutiert wurde. Nehmen wir einmal an, dass die zu integrierenden Datenquellen unterschiedliche Bewertungssysteme zur Festlegung der Qualität eines Hotels verwenden – zum Beispiel Sterne und Schlüssel. In den konkreten Datensätzen sind diese in den Attributen *class* bzw. *category* vermerkt. Das Problem besteht nun darin, die entsprechenden Bewertungen auf einer einheitlichen Skala abzubilden, um eine Vergleichbarkeit herzustellen.

Wie bereits erwähnt, besteht das Problem vor allem darin, dass die Bewertungen oft nur vor dem Hintergrund bestimmter Annahmen darüber möglich sind, was es bedeutet, wenn ein Hotel eine bestimmte Anzahl Sterne oder Schlüssel besitzt. Diese Annahmen lassen sich in einer gemeinsamen Ontologie explizit machen. Nehmen wir an, ein Hotel bekommt dann zwei Schlüssel verliehen, wenn es über einen Pool und einen eigenen Parkplatz verfügt. Diese Information lässt sich in das gemeinsame Modell einbringen, indem ein passendes Konzept für Hotels dieser Kategorie eingeführt wird, welches eine entsprechende Definition besitzt.

```
Class(TwoKeys complete(Accommodation
    restriction(has-facility
        someValuesFrom Pool)
    restriction(has-facility
        someValuesFrom Garage)))
```

Diese Definition ermöglicht es nun zum einen, für ein gegebenes Hotelobjekt aufgrund von dessen Beschreibung zu entscheiden, ob es zu der Kategorie TwoKeys gehört. Eine Anfrage nach Hotels dieser Kategorie liefert alle Hotelobjekte, die einen Pool und eine Garage besitzen, auch wenn diese nicht explizit als zu dieser Kategorie gehörig gekennzeichnet sind – natürlich vorausgesetzt, die entsprechenden Beschreibungen in den anderen Informationsquellen wurden in das gemeinsame Modell abgebildet, so dass sich die entsprechenden Objekte im Hinblick auf die Erfüllung der in der Definition erwähnten Restriktionen überprüfen lassen.

Darüber hinaus ist ein Vergleich häufig auch möglich, wenn für die zu bewertenden Objekte keine explizite Beschreibung der vorhandenen Einrichtungen (Pool, Garage), sondern nur eine abstrakte Bewertung zum Beispiel in Form von Sternen vorliegt – vorausgesetzt, die mit dieser Bewertung verbundenen Annahmen sind ebenfalls durch entsprechende Definitionen im Modell

enthalten. Nehmen wir an, ein Hotel der Kategorie „ein Stern“ zeichnet sich dadurch aus, dass es mindestens zwei „Facilities“ besitzt. Dies lässt sich ebenfalls in Form einer entsprechenden Klasse beschreiben.

```
Class(OneStar complete(Accommodation
    restriction(has-facility mincardinality(2))

ObjectProperty(has-facility range(Facility))

Class(Facility complete(unionOf(
    Pool Garage Restaurant)))
DisjointClasses(Pool Garage Restaurant)
```

Aus dieser Definition der Klasse „OneStar“ kann nun automatisch geschlossen werden, dass alle Hotels der Kategorie „TwoKeys“ automatisch auch der Kategorie „OneStar“ angehören, da diese per Definition alle geforderten Bedingungen erfüllen. Eine Verbindung zu den in den Informationsquellen enthaltenen Daten kann nun dadurch hergestellt werden, dass die Ausprägung der Attributwerte mit den neu eingeführten Klassen in Verbindung gesetzt wird. Dies könnte zwar auch durch entsprechende Axiome erfolgen, es ist aus Performanzgründen jedoch sinnvoller, Objekte bei der Übersetzung in das gemeinsame Modell bereits der entsprechenden Klasse explizit zuzuordnen. Ein Hotel der Kategorie „Zwei Schlüssel“ würde in diesem Fall nicht einfach dem Konzept Hotel, sondern zusätzlich dem Konzept TwoKeys zugeordnet.

### 6.2.3 Konsistenzprüfung

Ein weiterer Vorteil, Ontologien zur expliziten Modellierung von Annahmen in unterschiedlichen Informationsquellen zu verwenden, ist die Möglichkeit, solche Annahmen auf versteckte Widersprüche zu überprüfen. Zudem können Fehler aufgedeckt

werden, die durch eine nicht-korrekte Abbildung der Daten auf das gemeinsame Modell entstehen. Als Beispiel nehmen wir die Frage, ob für eine bestimmte Unterkunft ein Rabatt gegeben werden kann. Dies kann durch ein Konzept „liable-for-discount“ dargestellt werden, in das alle Unterkünfte fallen, für die ein Rabatt gewährt werden kann. Angenommen, in einer Informationsquelle wird für Konferenzhotels ein Rabatt gewährt. Dies kann durch die folgende Definition dargestellt werden:

```
Class(conference-Hotel
      restriction(has-facility
                  someValuesFrom(ConferenceFacility)))

SubClassOf(conference-Hotel liable-for-discount)
```

Nun kann es vorkommen, dass es in einer anderen Informationsquelle die Kategorie Luxus-Hotel gibt, für die kein Rabatt gegeben werden kann und deren Definition unter anderem folgende Beschreibung enthält:

```
Class(luxury-Hotel
      ...
      restriction(has-facility
                  someValuesFrom(ConferenceRoom))
      ...)

SubClassOf(luxury-Hotel
            complementOf(liable-for-discount))
```

Angenommen, „ConferenceRoom“ ist als Spezialfall von „Conference Facility“ definiert, dann wird die Klasse „luxury-hotel“ unerfüllbar, da Objekte der Klasse Luxushotel aufgrund ihrer Ausstattung auch gleichzeitig Konferenzhotels und somit Instanzen des Konzeptes „liable-for-discount“ sind. Auf der anderen Seite sind Luxushotels aber als nicht rabattfähig definiert. Dieser Konflikt kann mithilfe der in [Kapitel 5.2](#) erwähnten Methoden zur Diagnose von Ontologien identifiziert und behoben werden. Dies kann zum Beispiel dadurch geschehen, dass

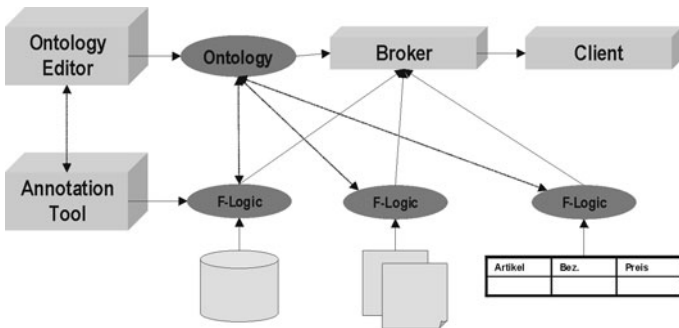
Luxushotels explizit aus den Rebattregelungen ausgenommen werden:

```
SubClassOf(  
    intersectionOf(conference-Hotel  
        complementOf(luxury-hotel))  
    liable-for-discount)
```

Auch in diesem Fall wird der Schritt des logischen Schließens nicht zur Laufzeit des Systems, sondern vorab in einem Validierungsschritt durchgeführt, um eine effiziente Auswertung von Anfragen zu ermöglichen.

## 6.3 Beispiel Ontobroker

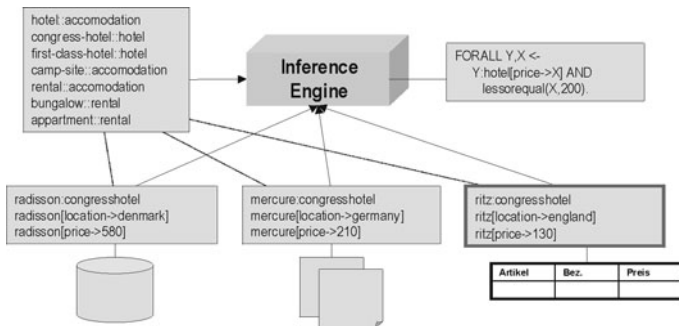
Ein konkretes Beispiel eines kommerziellen Systems, das Ontologien für die Datenintegration verwendet, ist Ontobroker der Firma Ontoprise. Dieses System implementiert im Prinzip die im vorangegangenen Kapitel beschriebenen Ideen, verwendet jedoch nicht OWL, sondern F-Logic aus [Kapitel 4.1.2](#) als Grundlage für das gemeinsame Datenmodell. Ableitungen und Konsistenzregeln werden in Ontobroker dementsprechend in Form von F-Logic Regeln definiert. Abbildung 6.4 zeigt die generelle Architektur des Ontobroker-Systems. Das Gesamtsystem besteht aus einer Reihe von Komponenten, die bei der Integration von Informationen auf der Grundlage von Ontologien eine Rolle spielen: Ein spezieller Editor für F-Logic Ontologien hilft dem Benutzer bei der Erstellung des gemeinsamen Modells. Ein Annotationswerkzeug unterstützt die Übersetzung von Dateninhalten in dieses gemeinsame Modell, und der Broker ermöglicht die Beantwortung komplexer Anfragen an das gemeinsame Modell. Ein spezieller Client vereinfacht die Kommunikation mit dem Broker und kann für bestimmte Anwendungen durch spezifische Clients ersetzt werden.



**Abb. 6.4** Generelle Architektur des Ontobroker-Systems

Abbildung 6.5 veranschaulicht die Funktionsweise des Systems anhand eines Beispiels, welches sich ebenfalls mit Hotels beschäftigt. Hier werden Informationen aus unterschiedlichen Informationsquellen – dies können sowohl Datenbanken als auch Webseiten oder Tabellenkalkulationsprogramme sein – mit Hilfe eines Wrappers nach F-Logic übersetzt. Die Zielstruktur dieser Daten orientiert sich hierbei an einer zuvor manuell erstellen Ontologie. Diese ist minimiert ebenfalls in der Abbildung dargestellt. In diesem Fall besteht die Ontologie lediglich aus einer einfachen Konzepthierarchie. Objekte sind diesen Klassen zugeordnet und besitzen zusätzlich die Eigenschaften „price“ und „location“. Anfragen an das Modell werden in Form komplexer F-Logik Regeln gestellt. In unserem Beispiel wird nach einem Hotel zu einem Preis von 200 oder weniger gefragt. Das Ergebnis dieser Anfrage in unserem Beispiel wäre das Ritz in England, da dieses als Kongresshotel automatisch auch vom Typ Hotel ist und der mit 130 angegebene Preis unter der geforderten Grenze von 200 liegt.

Eine Besonderheit von F-Logic und dem Ontobroker-System ist die Möglichkeit, komplexe Berechnungen auf Datenwerten

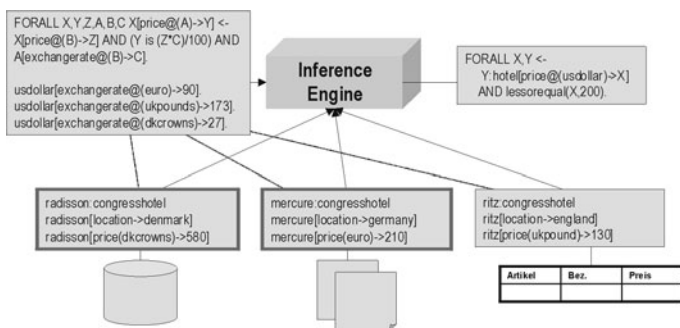


**Abb. 6.5** Beispiel für die Integration von Hotelinformationen mit F-Logic

vorzunehmen. Diese Fähigkeit, welche die meisten Ontologie-basierten Systeme nicht besitzen, ist im Kontext der Datenintegration häufig eine wichtige Funktion, um numerische Werte, die auf der Basis unterschiedlicher Skalen festgelegt wurden, entsprechend umzurechnen. In unserem Beispiel ist dies im Hinblick auf den Preis notwendig, da alle Länder, in denen die angezeigten Hotels liegen, unterschiedliche Währungen besitzen. So ist der Preis für das Ritz in Englischen Pfund angegeben, für das Mercure in Euro und für das Radisson in Dänischen Kronen. Angenommen, die Anfrage bezog sich auf einen Preis in US-Dollar, so müssen die in den Daten vermerkten Werte noch in Dollar umgerechnet werden, um ein falsches Ergebnis zu vermeiden.

Abbildung 6.6 zeigt eine erweiterte Version des Beispiels, welches nun auch die Umrechnung der Preise enthält. F-Logik bietet die Möglichkeit, Relationen zu parametrisieren. Damit können wir das Attribut „Preis“ mit der entsprechenden Währung, in der der Preis angegeben ist, parametrisieren. Außerdem können wir die Ontologie um eine Regel zur Umrechnung einer Währung in eine andere erweitern. Diese Regel

arbeitet auf einer Liste von Umrechnungskursen. In der Abbildung sind diese Umrechnungskurse explizit angegeben, in einer realen Anwendung würde man in der Regel die jeweils tagesaktuellen Kurse als eine eigene Informationsquelle ins System einbinden und als Grundlage der Umrechnung benutzen.



**Abb. 6.6** Erweitertes Beispiel mit Transformation der Preise

Die Umrechnung der Kurse in unserem Beispiel hat einen signifikanten Einfluss auf das Ergebnis der gezeigten Anfrage. Statt des Ritz ist nun das Radisson in Dänemark Ergebnis der Anfrage, da 580 Dänische Kronen umgerechnet weniger als 200 Dollar sind, während sowohl 210 Euro als auch 130 Pfund jeweils einem Dollarbetrag von mehr als 200 entsprechen.



## Kapitel 7

# Informationssuche

Neben strukturierten Daten, zum Beispiel aus Datenbanken, wie sie im Kapitel über Datenintegration behandelt wurden, gewinnen unstrukturierte Dokumente als Informationsquelle zunehmend an Bedeutung. Dies hat vor allem zwei Ursachen. Zum einen stehen in Form des World Wide Web inzwischen riesige Mengen an Dokumenten zu fast jedem denkbaren Thema zur Verfügung, zum anderen gibt es Angebote klassischer Bibliotheken zunehmend auch in elektronischer Form. Sie sind so für eine breite Masse potenzieller Nutzer zugänglich, die nun aus dem Angebot vieler verschiedener Bibliotheken wählen können. Infolge dieses riesigen Angebots an Dokumenten bekommt das Auffinden relevanter Dokumente eine besondere Bedeutung. Als Lösung wurden in der Vergangenheit Suchmaschinen entwickelt, die in der Lage sind, sehr große Dokumentbestände auf der Grundlage bestimmter Schlüsselwörter zu durchsuchen und Dokumente hinsichtlich ihrer Relevanz zu ordnen. Aufgrund der bereits im ersten Teil dieses Buches diskutierten Ambiguitäten von Begriffen enthalten die Ergebnisse einer solchen Suche oft Fehler oder relevante Dokumente werden nicht gefunden, da

diese zwar das entsprechende Thema adressieren, jedoch andere Begriffe verwenden als die Anfrage. In beiden Fällen kann die Verwendung spezieller Ontologien helfen, die Ergebnisse zu verbessern, indem zum einen ein standardisiertes Vokabular verwendet wird, um Anfrage und Dokumente zu vergleichen, und zum anderen Informationen über Synonyme bei der Suche berücksichtigt werden können.

Im Folgenden beschreiben wir kurz den üblichen Ansatz bei der Suche nach Dokumenten, das sogenannte Vektorraum-Modell, in dem Anfragen und Dokumente als Begriffsvektoren dargestellt und mit Hilfe von Vektoroperationen verglichen werden. Hierbei gehen wir auch auf Probleme ein, die durch synonyme und homonyme Begriffe bei der Suche entstehen. Anschließend beschreiben wir die Verwendung von Thesauri als spezielle Form von Ontologien für die Lösung dieser Probleme und stellen auch ein konkretes System vor, welches Ontologien bei der Suche nach relevanten Dokumenten einsetzt.

## **7.1 Klassische Dokumentensuche**

Das Problem der Suche nach relevanten Dokumenten beschäftigt Forscher aus dem Bereich des „Information Retrieval“ seit den siebziger Jahren. Im Laufe der Zeit wurden eine Reihe unterschiedlicher Ansätze entwickelt, die sich durch die Art der Formalisierung von Dokumenten und Anfragen sowie die Relevanz eines Dokumentes im Hinblick auf eine gegebene Anfrage unterscheiden. Die unterschiedlichen Modelle darzustellen, würde den Rahmen dieses Buches sprengen und uns vom eigentlichen Thema der Verwendung von Ontologien bei der Dokumentensuche wegführen. Wir beschränken uns daher auf die Erläuterung des Vektorraum-Modells als populärstem Modell und auf

die übliche Bestimmung der Relevanz eines Dokuments auf der Grundlage des sogenannten TF-IDF Wertes.

### ***7.1.1 Relevanz von Dokumenten***

Das Grundprinzip der Dokumentensuche besteht darin, bei gefundenen Dokumenten abzuschätzen, wie relevant diese für eine gegebene Anfrage sind. Die Dokumente werden dann gemäß der ermittelten Relevanz sortiert und dem Benutzer als Ergebnis präsentiert. Da häufig keine zusätzlichen Informationen über ein Dokument vorliegen als der darin enthaltene Text, wird die Relevanz in der Regel auf dessen Grundlage ermittelt. Klassische Ansätze nehmen hierbei die einzelne Wörter als Basis der Abschätzung. Das einfachste denkbare Modell ist die Abfrage, ob ein bestimmtes Wort in einem Dokument vorhanden ist. Die Anfrage wäre demnach ein einzelnes Wort. Die Relevanz eines Dokumentes hätte einen Wert von 1, wenn dieses Wort im Dokument vorkommt, und 0, wenn es nicht vorkommt. Ergebnisse der Anfrage wären in diesem Fall alle Dokumente mit einer Relevanz von 1. Dieses recht einfache Modell weist eine Reihe von Problemen auf, die dazu geführt haben, dass Erweiterungen und Verbesserungen entwickelt wurden. Zunächst einmal müssen im Text enthaltene Wörter identifiziert werden. Diese Aufgabe, die auch als Tokenization bezeichnet wird, erscheint zunächst trivial, kann aber im Falle von Eigennamen oder zusammengesetzten Wörtern durchaus zu Fehlern führen, da Wörter nicht als solche erkannt oder aber an der falschen Stelle getrennt werden. Ein weiteres Problem stellen Beugungsformen dar. Wörter tauchen in Texten selten in ihrer Grundform auf, sondern verändern sich je nachdem ob sie im Singular oder Plural bzw. in unterschiedlichen Kasus verwendet werden. Auch wenn die Anfrage

in der Grundform gestellt wird, ist offensichtlich, dass auch Dokumente, die das entsprechende Wort in einer Beugungsform enthalten, relevant sind. Um dieses Problem zu lösen, werden Anfragen und Wörter im Text in der Regel auf ihren Wortstamm (vgl. Kapitel 1.3) reduziert und diese Stämme werden dann als Grundlage des Vergleichs herangezogen. Dieser Vorgang wird auch als „Stemming“ bezeichnet. Sowohl für Stemming als auch Tokenization wurden robuste Methoden entwickelt, die in der Lage sind, diese Probleme für gängige Sprachen weitestgehend zu lösen.

Es ist also ohne weiteres möglich, aus einer Dokumentenmenge alle Dokumente herauszusuchen, die ein bestimmtes Wort enthalten, und diese dem Benutzer zu präsentieren. Dies ist jedoch nur bedingt nützlich, da es zum einen sehr viele Dokumente geben kann, die ein bestimmtes Wort enthalten, zum anderen kann ein Dokument, nur weil es ein bestimmtes Wort enthält, nicht unbedingt als relevant im Hinblick auf den entsprechenden Begriff bezeichnet werden. Oft kommt es vor, dass bestimmte Begriffe nur am Rande erwähnt werden, ohne wirklich das Thema eines Dokumentes zu bestimmen. Was in diesem einfachen Modell bisher fehlt, ist die Möglichkeit, unterschiedliche Stufen von Relevanz zu unterscheiden und so Dokumente entsprechend zu ordnen.

Übliche Ansätze zur Bestimmung der Relevanz verwenden das Kriterium der Häufigkeit, mit der ein bestimmter Begriff in einem Dokument auftritt, als Grundlage. Diese Häufigkeit wird auch als „term frequency“ oder kurz TF bezeichnet. Hierbei wird angenommen, dass ein Begriff, der nur am Rande von Bedeutung ist, weniger häufig erwähnt wird als ein zentraler Begriff. Diese einfache Verwendung der „term frequency“ als Relevanzmaß ist jedoch problematisch, da sie aufgrund ihrer Definition lange Dokumente gegenüber kürzeren bevorzugt. Es ist relativ einleuchtend, dass in einem mehrere hundert Seiten starken

Buch die meisten Begriffe häufiger auftreten als in einer einseitigen Zusammenfassung. Um diesen Vorteil auszugleichen, kann die Häufigkeit mit der Länge des Dokuments normalisiert werden. Das Ergebnis beschreibt die relative Häufigkeit, mit der der Suchbegriff im Dokument auftritt, bzw. die Wahrscheinlichkeit, dass man den Suchbegriff erhält, wenn man zufällig ein Wort aus dem Dokument auswählt. Das entsprechende Relevanzmaß, welches auch als „keyword density“ (KD) bezeichnet wird, ist wie folgt definiert:

$$KD_{kj} = \frac{TF_{kj}}{\sum_{i=1}^{|D|} tf_{ij}}$$

Hierbei ist  $tf_{ij}$  die term frequency des Begriffs  $i$  in Dokument  $j$ ,  $|D|$  ist die Anzahl der Dokumente in der Dokumentenbasis. Die Summe im Nenner beschreibt die Anzahl aller Wörter in dem Dokument  $j$  und somit dessen Länge.

Diese Vorgehensweise, die Relevanz zu bestimmen, löst zwar das Problem der Überbewertung relevanter Dokumente, es führt jedoch zu einer Fehleinschätzung der Relevanz eines Dokuments im Hinblick auf häufig verwendete Begriffe. So gibt es in jedem Anwendungsgebiet bestimmte Begriffe, die so generell sind, dass sie in den meisten Dokumenten erwähnt werden. Dies bedeutet jedoch, dass das einzelne Dokument weniger relevant wird – es beschäftigen sich ja so gut wie alle Dokumente mit diesem Begriff. Demgegenüber muss ein Dokument, welches sich als einziges mit einem Begriff beschäftigt, für diesen relevanter sein, da es keine andere Möglichkeit gibt, Informationen über diesen Begriff zu bekommen. Man kann also sagen, dass dieser Begriff charakteristisch für den Inhalt des Dokumentes ist, da es dieses von anderen Dokumenten unterscheidet. Um diesen Effekt ebenfalls bei der Bestimmung der Relevanz berücksichtigen zu können, wurde das sogenannte TF-IDF Maß entwickelt, welches folgendermaßen definiert ist:

$$tf - idf_{ij} = \frac{TF_{ij}}{\sum_{k=1}^{|D|} TF_{kj}} \cdot \log\left(\frac{|D|}{DF_i}\right)$$

Hierbei sind  $TF_i$  und  $|D|$  definiert wie bisher,  $DF_i$  bezeichnet die Anzahl der Dokumente, in denen der Begriff  $i$  vorkommt. Der Term, mit dem hier zusätzlich multipliziert wird, beschreibt, wie speziell der Begriff  $i$  ist. Hierzu wird die Anzahl der Dokumente durch die Anzahl derjenigen Dokumente geteilt, in denen der Begriff vorkommt. Ist dieser in fast allen Dokumenten enthalten, so ergibt sich ein Wert, der nahe bei eins ist. Dies bedeutet wiederum, dass der Logarithmus dieses Ausdrucks gegen null geht. Er geht jedoch gegen eins, wenn der entsprechende Begriff in sehr wenigen Dokumenten vorkommt. Der entsprechende Wert, der auch als „Inverse Document Frequency“ (IDF) bezeichnet wird, bestraft also Begriffe, die in vielen Dokumenten enthalten sind und belohnt solche, die selten vorkommen. Multipliziert mit der Dichte eines Begriffs in dem betrachteten Dokument, ergibt sich ein Maß, welches eine sehr gute Abschätzung der Relevanz eines Dokumentes ermöglicht. Es repräsentiert den heutigen Stand der Technik bei der Bewertung der Relevanz von Dokumenten. Tabelle 7.1 zeigt ein Beispiel für die Berechnung der Relevanz unterschiedlicher Begriffe in einer Dokumentenmenge.

Eine wichtige Erkenntnis hierbei ist, dass Begriffe, die in nahezu allen Dokumenten vorkommen, keine sinnvolle Abschätzung der Relevanz erlauben und daher bei der Dokumentensuche keine Hilfe sind. Solche Begriffe werden daher in der Regel mit Hilfe sogenannter Stop-Word Listen bereits vorher aus der Beschreibung von Dokumenten entfernt. Stop Word Listen enthalten Begriffe, die aus der Darstellung entfernt werden sollen. Dies geschieht dann in der Regel nach der Ermittlung der Wortgrenzen.

**Tabelle 7.1** Beispiel der Berechnung des TF-IDF Wertes für eine Dokumentenmenge

Query Q: 'gold silver truck'											
Document D <sub>1</sub> : 'Shipment of gold damaged by fire'											
Document D <sub>2</sub> : 'Delivery of silver arrived in a silver truck'											
Document D <sub>3</sub> : 'Shipment of gold arrived in a truck'											
	Häufigkeit							Relevanz			
Terms	Q	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	DF <sub>i</sub>	D/DF <sub>i</sub>	log(D/DF <sub>i</sub> )	Q	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>
a	0	1	1	1	3	1	0	0	0	0	0
arrived	0	0	1	1	2	1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	1	0	0	0	0	0
of	0	1	1	1	3	1	0	0	0	0	0
silver	1	0	2	0	1	3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	1.5	0.1761	0.1761	0	0.1761	0.1761

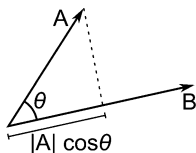
### 7.1.2 Das Vektorraum-Modell

Die oben beschriebene Bewertung der Relevanz eines Dokuments bezog sich bisher jeweils nur auf einen einzelnen Begriff als Anfrage. In der Realität lassen sich Themen, zu denen Dokumente gesucht werden, meist nicht durch einen einzigen Begriff beschreiben.<sup>1</sup> Dieses Problem lässt sich also nicht durch die Relevanz eines einzelnen Wortes lösen, vielmehr müssen Kombinationen von Begriffen betrachtet werden. Zu diesem Zweck wurde das Vektorraum-Modell als Grundlage für die Darstellung

<sup>1</sup> Allerdings zeigt das Beispiel von Internet-Suchmaschinen, dass meist auch nicht mehr als drei Begriffe als Anfrage verwendet werden.

und den Vergleich von Dokumenten und Anfragen entwickelt. In diesem Modell werden sowohl Dokumente als auch Anfragen als Vektoren in einem mehrdimensionalen Raum dargestellt. Die Dimensionen dieses Raumes entsprechen möglichen Begriffen, die in den Dokumenten vorkommen können, oder, genauer gesagt, deren Wortstämmen, zwischen denen ja, wie oben beschrieben, nicht unterschieden wird. Die Werte des Vektors, der ein Dokument beschreibt, entspricht den TF-IDF Werten des entsprechenden Begriffs. Vektoren, die Anfragen darstellen, enthalten für die Begriffe in der Anfrage einen Wert – normalerweise den Wert eins und für nicht vorkommende Begriffe den Wert null. Es besteht jedoch auch die Möglichkeit, explizit Werte für die Wichtigkeit der unterschiedlichen Begriffe anzugeben. Die Relevanz eines Dokumentes in Bezug auf eine Anfrage lässt sich nun durch die Distanz der Vektoren der Anfrage und des entsprechenden Dokumentes in dem Vektorraum bestimmen: Vektoren mit einer großen Entfernung stimmen nur wenig in der Bewertung der Begriffe überein; Vektoren, die nahe beieinander liegen, besitzen eine große Übereinstimmung.

Die Berechnung der Entfernung zweier Vektoren in einem Vektorraum kann im Prinzip auf unterschiedliche Weise erfolgen. Das gängigste Maß hierfür ist die Kosinus-Ähnlichkeit der Vektoren. Hierbei wird der Kosinus des Winkels zwischen den Vektoren im  $n$ -dimensionalen Raum als Abstandsmaß verwendet. Abbildung 7.1 zeigt das Prinzip der Abstandsbestimmung



**Abb. 7.1** Prinzip der Relevanzbestimmung von Dokumenten im Vektorraum-Modell



zwischen zwei Vektoren im dreidimensionalen Raum. Für den Fall, in dem Anfrage und Dokument als Vektoren  $a = (a_1 \cdots a_n)$  und  $b = (b_1 \cdots b_n)$ , wobei  $a_i$  und  $b_i$  jeweils die Relevanzbewertungen der einzelner Wörter in Anfrage und Dokument sind, kann folgende Formel zur Berechnung verwendet werden:

$$\cos(\theta) = \frac{a \cdot b}{|a||b|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}} \quad (7.1)$$

In dem Beispiel aus Abb. 7.1 ergeben sich durch die Anwendung des Kosinusmaßes, dass Dokument 2 am relevantesten in Bezug auf die Anfrage ist. Die Berechnung des konkreten Relevanzwertes sei dem Leser zur Übung überlassen.

Es ist wichtig festzuhalten, dass das Vektorraum-Modell in seiner allgemeinen Form eher wenig praktikabel ist, da wir es im Fall realer Dokumente mit sehr vielen möglichen Dimensionen zu tun haben und dies die Berechnung des Abstandes entsprechend aufwändig macht. In der Praxis gibt es unterschiedliche Möglichkeiten, um diese Komplexität zu reduzieren. Zum einen kann die Anzahl der Dimensionen reduziert werden, indem, wie bereits oben beschrieben, Begriffe mit einem sehr niedrigen TF-IDF Wert nicht berücksichtigt werden. Zum anderen ist, wie oben erwähnt, der Anfragevektor in der Regel eher niedrigdimensional, was den Aufwand der Abstandsberechnung erheblich reduziert. So ist diese Form die Relevanz zu bestimmen, eine Verallgemeinerung des einfachen Modells, welches im vorangegangenen Kapitel behandelt wurde. Insbesondere entspricht das Vektorraum-Modell dem oben beschriebenen für den Fall, in dem nur ein Begriff betrachtet wird und der Vektorraum folglich nur eine Dimension besitzt. Hinzu kommt eine Vielzahl

technischer Ansätze zur Beschleunigung der Berechnung, die hier nicht im Detail behandelt werden können.

### ***7.1.3 Probleme des klassischen Modells***

Dieses allgemein verwendete Modell für die Bewertung der Relevanz eines Dokuments hat zwei grundlegende Schwächen, die sich aus der Ambiguität der natürlichen Sprache ergeben. Das Modell ist in erster Linie anfällig für synonyme und homonyme Begriffe in Anfragen, was zu unkorrekten Ergebnissen bei der Relevanzbewertung führt.

**Einfluss von Synonymen:** Wie in Kapitel 1.3 dargestellt, sind Synonyme unterschiedliche Begriffe, die für das gleiche Konzept verwendet werden können. Da es sich um unterschiedliche Wörter mit unterschiedlichem Stamm handelt (Beispiel: Bank, Geldinstitut), werden diese an verschiedenen Stellen in der entsprechenden Vektordarstellung von Anfragen und Dokumenten gezeigt. In die Berechnung der Relevanz eines Dokumentes gehen synonyme Begriffe daher nicht angemessen ein. Dies zeigt das Beispiel, in dem nach „Bank“ gefragt wird, das Dokument jedoch ausschließlich das Wort Geldinstitut verwendet. In diesem Fall wird der Nenner des Bruches aus Gleichung 7.1 und somit auch der Relevanzwert null. Das entsprechende Dokument wird also nicht als relevant erkannt. Schuld hieran ist die Unfähigkeit des Modells, mit Synonymen angemessen umzugehen.

**Einfluss von Homonymen:** Homonyme (vgl. Kapitel 1.3) sind Wörter, die je nach Kontext unterschiedliche Bedeutungen haben können. Dies kann dazu führen, dass Dokumente fälschlicherweise als relevant angesehen werden, die in

Wahrheit jedoch nicht relevant für eine gegebene Anfrage sind. Dies passiert dann, wenn Anfrage und Dokumente zwar das gleiche Wort benutzen, die intendierten Bedeutungen jedoch unterschiedlich sind. Eine Anfrage nach einer Bank im Sinne eines Sitzmöbels wird zum Beispiel in der Regel auch Dokumente über Finanzinstitute zurückliefern. Dies liegt daran, dass eine fehlende Unterscheidung zwischen den unterschiedlichen Bedeutungen eines Wortes die Häufigkeit eines Begriffes, welche ja maßgeblich in die Berechnung des TF-IDF Wertes eingeht, künstlich erhöht. Ein weiterer Effekt ist die nicht angemessene Berücksichtigung seltener Bedeutungen eines Wortes, da durch die fehlende Unterscheidung der IDF-Wert für dieses Wort stark sinkt, sobald es viele Dokumente gibt, welche üblichere Bedeutungen des entsprechenden Wortes enthalten.

Ein weiteres Problem des Vektorraum-Modells ist das potenzielle Durcheinander von Abstraktionsebenen in Anfrage und durchsuchten Dokumenten. Hier treten, wenn auch in geringerem Maße, die gleichen Probleme auf, die wir auch schon im Zusammenhang mit Synonymen beschrieben haben. So sind Dokumente, die Unternehmen einer bestimmten Branche, etwa Banken, beschreiben, oft auch in Bezug auf eine allgemeine Anfrage nach Dokumenten zum Thema Unternehmen relevant. Aus den gleichen Gründen wie bei Synonymen wird diese Relevanz nicht angemessen im Modell abgebildet.

## 7.2 Thesaurus-basierte Suche

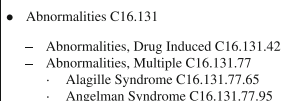
Im vorangegangenen Kapitel wurde beschrieben, wie sich Ontologien zur Integration von strukturierten Informationen verwenden lassen. Hierbei kamen vor allem Ontologien, die auf dem Prinzip der Logik-basierten Darstellung von Semantik beruhen,

zum Einsatz. Zur Lösung der oben erläuterten Probleme bei der Suche nach unstrukturierten Dokumenten spielen nun Ontologien, die eher in der Tradition der linguistischen Semantik stehen, eine Rolle. Dabei sind hier die sogenannten Thesauri von besonderer Bedeutung.

### ***7.2.1 Thesaurus-basierte Verschlagwortung***

Thesauri sind spezielle semantische Netze, die, ähnlich dem in Kapitel 3.1 beschriebenen WordNet-Netzwerk, die Bedeutung von Wörtern in Form von linguistischen Relationen zu anderen Wörtern beschreiben. Thesauri spielen im Bereich digitaler, aber auch klassischer Bibliotheken seit längerem eine wichtige Rolle bei der Erschließung von Fachinformationen. Dementsprechend bestehen bereits eine Reihe von Standards zur Beschreibung von Thesauri, unter anderem ein ISO-Standard, in dem die in einem Thesaurus verwendeten Relationen sowie deren konkrete Beschreibung festgelegt sind. Der Aufbau eines Thesaurus basiert analog zu WordNet auf der Beschreibung von Konzepten durch eine Menge von Wörtern, die in der Sprache verwendet werden können, um dieses Konzept zu beschreiben. Im Gegensatz zu WordNet, dessen Aufbau vom Konzept einer Synonymmenge ausgeht, in der alle Synonyme gleichberechtigt enthalten sind, weisen Thesauri eines der Synonyme als präferierten Begriff aus, der zu verwenden ist, um das Konzept zu beschreiben. Dabei werden besonders präferierte Begriffe als Schlagwörter zur Beschreibung des Inhalts von Dokumenten eingesetzt. Ergänzt wird dieser präferierte Begriff durch Informationen über Synonyme: allgemeinere, speziellere sowie verwandte Begriffe, die jedoch eine andere Bedeutung besitzen. Abbildung 7.2 zum Beispiel zeigt einen Ausschnitt der

Konzepthierarchie des MeSH-Thesaurus, eines der bekanntesten medizinischen Thesauri, der auch ein Teil des in Kapitel 3.2 vorgestellten UMLS-Metathesaurus ist. Die Abbildung zeigt das Konzept „Abnormalities“ sowie einige Unterkonzepte, insbesondere „Drug Induced Abnormalities“ und „Multiple Abnormalities“. Letzteres hat wiederum die Konzepte „Alagille Syndrome“ und „Angelman Syndrome“ als Unterkonzepte.

- 
- Abnormalities C16.131
    - Abnormalities, Drug Induced C16.131.42
    - Abnormalities, Multiple C16.131.77
      - Alagille Syndrome C16.131.77.65
      - Angelman Syndrome C16.131.77.95

**Abb. 7.2** Ausschnitt aus dem MeSH Thesaurus

In der Abbildung sind die Konzepte hierbei nur durch die präferierten Begriffe angegeben. Der Thesaurus enthält zu jedem Konzept jedoch noch weitere Informationen, welche dessen Bedeutung näher erklären. Tabelle 7.2 zeigt die entsprechenden Erklärungen für das Konzept Ethics.

Diese zusätzlichen Informationen sollen entscheiden, wann ein Dokument mit dem Schlagwort „Ethics“ annotiert wird und somit Ergebnis einer Suche nach dem Begriff Ethics sein soll. Traditionell dienen Thesauri als Unterstützung für die manuelle Verschlagwortung von Dokumenten durch Bibliothekare. Dementsprechend enthält die Definition eines Konzeptes eine natürlichsprachliche Erklärung, die sogenannte Scope Note. Sie ist eine Erklärung für den Bibliothekar, in welchen Fällen der Begriff zu verwenden ist. In der Abbildung wird hier auf zwei unterschiedliche Verwendungen, einmal im Sinne einer abstrakten Idee und einmal im Sinne einer wissenschaftlichen Forschungsrichtung, hingewiesen. Es kommt auch vor, dass an

**Tabelle 7.2** Beispiel eines MeSH Descriptors

MeSH Heading	Ethics
Tree Numbers	F01.829.500.519 K01.316 K01.752.256 N05.350
Scope Note	The philosophy or code pertaining to what is ideal in human character and conduct. Also, the field of study dealing with the principles of morality.
Entry Terms	Egoism Ethical Issues Metaethics Moral Policy Natural Law Situational Ethics
Allowable Qualifiers	CL HI
Unique ID	D004989

dieser Stelle auf andere Konzepte verwiesen wird, die eventuell anstelle des Begriffe verwendet werden sollen. Ein Beispiel ist das Konzept „child“, dessen Scope Note folgenden Text enthält:

A person 6 to 12 years of age. An individual 2 to 5 years old is CHILD, PRESCHOOL.

Hier wird nicht nur die Definition des Begriffs im Hinblick auf das Lebensalters der betrachteten Person angegeben, sondern auch auf das ebenfalls vorhandene Konzept „Preschool child“ hingewiesen. Außerdem enthält die Definition Informationen über Wörter, die möglicherweise in einem Dokument auftreten können und darüber, in welchem Fall jedoch der präferierte Begriff als Schlagwort verwendet werden soll. Die entsprechenden Begriffe, die in MeSH als Entry Terms bezeichnet werden, sind oft, jedoch nicht immer, Synonyme des

präferierten Begriffs. Dies zeigt sich sehr schön an dem Beispiel aus Abb. 7.2. Als Entry Terms finden wir hier zum Beispiel „Ethical Issues“, was als Synonym zu „Ethics“ gesehen werden kann, jedoch auch den Begriff „Egoism“, der zwar mit Ethik zu tun hat, jedoch streng genommen kein Synonym von „Ethics“ darstellt. Dieser scheinbare Fehler kann durch die Natur des Thesaurus als Handlungsanweisung für den annotierenden Bibliothekar erklärt werden. Thesauri vermischen also ontologische Prinzipien mit pragmatischen Entscheidungen, die sich aus den Notwendigkeiten der Anwendung ergeben. So hätte „Egoism“ auch als eigenes Konzept definiert werden können, es wurde jedoch entschieden, dass dieser Begriff nicht wichtig genug ist, um ein eigenes Konzept zu rechtfertigen. Einen weiteren Teil der Beschreibung, der durch die manuelle Verschlagwortung von Dokumenten motiviert ist, stellen die sogenannten „Qualifier“ dar. Diese sind eigene Kategorien, die angeben, in welchem Kontext der beschriebene Begriff in einem Dokument verwendet wird. Für das Konzept „Ethics“ aus unserem Beispiel sind zwei mögliche Quantifier vorgesehen, die durch ihre Abkürzungen CL bzw. HI angegeben sind. CL steht hierbei für den Qualifier „classification“, HI für „history“. Wird ein Dokument nun mit dem Konzept „Ethics“ und dem Qualifier „HI“ verschlagwortet, so bedeutet dies, dass in dem Dokument unter anderem das Thema Ethik aus historischer Sicht behandelt wird.

Üblicherweise wird die Verschlagwortung eines Dokuments, zumindest im Bereich wissenschaftlicher Publikationen, auf der Basis der Zusammenfassung (Abstract) durchgeführt. Abbildung 7.3 zeigt das Abstract eines Artikels aus der Zeitschrift „Journal of Hazardous Materials“. Das Dokument behandelt die potenzielle Gefährdung von Höhlenführern durch Radon und mögliche Maßnahmen zur Reduzierung dieser Gefährdung.

Tabelle 7.3(b) zeigt MeSH-Konzepte, mit denen das Dokument manuell annotiert wurde. Neben offensichtlichen

**Title** Analysis of the main factors affecting the evaluation of the radon dose in workplaces: the case of tourist caves.

**Authors** Carlos Sainz, Luis Santiago Quindós, Ismael Fuente, Jorge Nicolás and Luis Quindós

**Abstract** High concentrations of radon exist in several workplaces like tourist caves mainly because of the low ventilation rates existing at these enclosures. In this sense, in its 1990 publication, the ICRP recommended that high exposures of radon in workplaces should be considered as occupational exposure. In developed caves in which guides provide tours for the general public great care is needed for taking remedial actions concerning radon, because in some circumstances forced ventilation may alter the humidity inside the cave affecting some of the formations or paintings that attract tourists. Tourist guides can work about 1900 h per year, so the only option to protect them and other cave workers from radon exposure is to apply an appropriate system of radiation protection mainly based on limitation of exposure by restricting the amount of time spent in the cave. Because of the typical environmental conditions inside the caves, the application of these protecting actions requires to know some indoor air characteristics like particle concentration, as well as radon progeny behaviour in order to get more realistic effective dose values. In this work the results of the first two set of radon measurements program carried out in 10 caves located in the region of Cantabria Spain are presented.

**Journal** Journal of Hazardous Materials

**Abb. 7.3** Abstract eines Dokumentes als Grundlage für die Thesaurus-basierte Verschlagwortung

Konzepten wie Radon und Spain, die direkt im Abstract erwähnt werden, wurden eine Reihe von Konzepten als Schlagwörter gewählt, die dort nicht direkt vorkommen, sondern von den konkret erwähnten Themen abstrahieren. Beispiele hierfür sind „Radioactive Air Pollutants“ und „Leisure Activities“. Diese Begriffe zeigen klar, dass die manuelle Verschlagwortung eine Aufgabe darstellt, die eine intellektuelle Leistung erfordert. Verlangt ist nicht nur die Fähigkeit, von den konkreten Themen des Dokumentes abstrahieren zu können, sondern auch eine



**Tabelle 7.3** Automatisch zugeordnete vs. manuell selektierte Konzepte

Concepts (Rank)	Keywords
Radon (1)	Air Pollutants, Occupational
Ventilation (0,33)	Air Pollutants, Radioactive
Work (0,33)	Environmental Exposure
Workplace (0,33)	Humans
Affect (0,16)	Leisure Activities
Air (0,16)	Occupational Exposure
Health Services Needs and Demand (0,16)	Radon
Humidity (0,16)	Spain
Radiation Protection (0,16)	
Spain (0,16)	
Occupational Exposure (0,16)	

(a) Automatic

(b) Manual

detaillierte Kenntnis des verwendeten Thesaurus, um passende Konzepte finden zu können. Die manuelle Verschlagwortung ist aus diesem Grund sehr aufwändig.

Um den Aufwand der Verschlagwortung von Dokumenten zu reduzieren, wurden automatische Verfahren zur Bestimmung geeigneter Schlagwörter auf der Grundlage eines Thesaurus entwickelt. Diese Verfahren verwenden in der Regel Variationen der Relevanzmaße, die im vorangegangenen Kapitel beschrieben wurden. Wesentlicher Unterschied zum allgemeinen Modell für die Berechnung der Relevanz einzelner Begriffe ist die Beschränkung auf Konzepte aus dem Thesaurus – der Dokumentvektor beinhaltet nicht alle vorkommenden Wortstämme, sondern nur Konzepte aus dem Thesaurus. Außerdem wird bei der Bestimmung der „term frequency“ das Vorkommen der „entry terms“ sowie des präferierten Begriffs aufsummiert und diese ergeben zusammen die Häufigkeit des Konzeptes. Tabelle 7.3(a)

zeigt das Ergebnis einer solchen automatischen Verschlagwortung. Die Zahlen hinter den Begriffen geben die relative Häufigkeit des jeweiligen Begriffes an, ein weiteres mögliches Maß für die Relevanz. Bei der Bestimmung der Häufigkeit spielen hierbei jetzt zwei Mechanismen eine Rolle. Zum einen werden, wie im klassischen Modell, Wortstämme betrachtet (das Vorkommen des Begriffs „worker“ wird mit zum Konzept „work“ gezählt), zum anderen erfolgt die beschriebene Aufsummierung über die Entry Terms. So taucht der Begriff „Health Services Needs and Demand“ nicht im Abstract auf, wohl aber der Entry term „need“, der in der Form des Wortes „needed“ enthalten ist und zum entsprechenden Konzept dazugezählt wird.

### ***7.2.2 Anfrage-Normalisierung und -expansion***

Der Vorteil bei der Verwendung eines Thesaurus zur Verschlagwortung von Dokumenten liegt nun vor allem im Bereich der Dokumentensuche. Die Verschlagwortung auf der Grundlage von Konzepten hilft, mehr relevante Dokumente zu finden als dies auf der Grundlage von Wörtern möglich ist. Erreicht wird dies zum einen durch eine Normalisierung von Anfragen und Schlagwörtern sowie durch die Expansion der Anfrage.

Das Prinzip der Normalisierung wurde im Grundsatz bereits in Verbindung mit der Annotation von Dokumenten beschrieben. Hierbei werden, wie oben erläutert, Dokumente nicht durch im Dokument vorkommende Wörter, sondern durch die entsprechenden Konzepte beschrieben. Dies hat zum einen den Vorteil, dass, vorausgesetzt die Schlagwörter sind korrekt, das Thema eines Dokumentes eindeutiger beschrieben wird, da Konzepte eines Thesaurus im Gegensatz zu Wörtern eindeutig bestimmt sind. Zum anderen liegt der Vorteil darin, dass von

unterschiedlichen Möglichkeiten ein Konzept zu beschreiben, abstrahiert und durch eine normalisierte Beschreibung ersetzt wird. Der Vorteil dieser Normalisierung kommt nun vor allem dann zur Geltung, wenn wir das gleiche Prinzip auf Anfragen anwenden, die an eine Dokumentenmenge gestellt werden. Hierbei wird die Anfrage, die in der Regel aus einer Menge von Wörtern besteht, ebenfalls normalisiert, indem Wörter durch die entsprechenden Begriffe ersetzt werden. Diese lassen sich dann eindeutig mit den Schlagwörtern der Dokumente vergleichen.

Nimmt man das Dokument aus Abb. 7.3 als Beispiel, so könnte folgende Situation vorkommen. Nehmen wir an, ein Benutzer stellt die Anfrage: *Radon „Job Site“*. Vergleicht man die Suchwörter in dieser Anfrage mit dem MeSH Thesaurus, so stellt man fest, dass Radon dem MeSH Konzept mit der ID D011886 entspricht. „Job Site“ ist nicht selbst ein Heading, wird aber als Entry Term im Kontext des Konzeptes „workplace“ genannt. Dieses Konzept besitzt die eindeutige ID D017132. Hieraus ergibt sich eine normalisierte Version der Anfrage der Form: „D011886, D017132“. Diese Anfrage kann nun eindeutig mit der Beschreibung des Beispieldokumentes, welches aus den IDs der in Tabelle 7.3(a) aufgeführten Konzepte besteht und neben anderen auch die beiden in der normalisierten Anfrage vorhandenen Konzept IDs enthält, identifiziert werden. Anstatt die Wörter in der Anfrage mit den Wörtern im Dokument zu vergleichen, vergleicht man die IDs der entsprechenden Konzepte miteinander. Wie das Beispiel zeigt, führt dies zu exakteren Ergebnissen, da durch den Normalisierungsprozess Unterschiede in der Wortwahl wie in unserem Beispiel (job site vs. workplace) eliminiert werden. Diese Art der Expansion von Anfragen durch Unterkonzepte entspricht hierbei einer sehr eingeschränkten Form von Schließen über Vererbungsrelationen. Im Gegensatz zum logischen Schließen wird hierbei jedoch nicht die Zugehörigkeit von Objekten zu übergeordneten Konzepten

abgeleitet, sondern die Anfrage so modifiziert, dass auf explizit vorhandenen Informationen gearbeitet werden kann.

Dieser Normalisierungsprozess ist allerdings nur in der Lage, unterschiedliche Bezeichnungen des gleichen Konzeptes auszugleichen. Eine weitere wichtige Funktion von Ontologien bei der Suche liegt im Umgang mit unterschiedlichem Detailierungsgrad und inhaltlichen Zusammenhängen zwischen Konzepten in Anfrage und Dokumentenbeschreibung. Nehmen wir an, ein Anwender stellt die Anfrage: „Radioactive Air Pollutants“ „Job Site“ „Europe“. Unser Beispieldokument ist nach allgemeinem Verständnis ebenfalls relevant für diese Anfrage. Mit Hilfe der oben beschriebenen Normalisierung lässt sich jedoch nur „Job Site“ mit dem Dokument in Verbindung bringen, obwohl für das Dokument auch die beiden anderen Begriffe relevant sind. Betrachtet man die Hierarchie des MeSH Thesaurus, so stellt man fest, dass dort „Europe“ ein Oberbegriff von „Spain“ ist. Dies ist so zu interpretieren, dass Dokumente, die von Spanien handeln, auch relevant für Anfragen nach Europa sind. Diesen Zusammenhang kann man nun bei der Bearbeitung von Anfragen nutzen, indem man die Anfrage dadurch erweitert, dass man Unterbegriffe der enthaltenen Konzepte hinzufügt. In unserem Fall würde man unter anderem die Unterkonzepte von Europa, zu denen neben Spanien auch alle anderen europäischen Länder und einige Regionen wie „Skandinavien“ zählen, hinzufügen. Hierdurch werden auch Dokumente als relevant identifiziert, die sich mit diesen Ländern und dadurch indirekt mit Europa beschäftigen.

Noch komplizierter ist die Situation im Fall des Suchbegriffes „Radioactive Air Pollutants“. Im Beispiel wird mit „Radon“ ein Stoff erwähnt, der in die Klasse der radioaktiven Stoffe fällt, welche die Luft verunreinigen. Das Konzept „Radioactive Air Pollutants“ erscheint in der Definition von Radon unter der Rubrik „see also“. Wie man schön anhand der manuellen Annotationen

aus Tabelle 7.3(b) sehen kann, ist der ursprüngliche Zweck dieses Verweises, Bibliothekare bei der manuellen Annotation von Dokumenten auf möglicherweise geeignetere Schlagwörter, in diesem Fall Radioactive Air Pollutants anstelle von Radon, hinzuweisen. Diesen Zusammenhang, der wie Tabelle 7.3(b) bei der manuellen Verschlagwortung genutzt wird, kann man sich nun auch bei der Suche zunutze machen, in dem man auch „see also“ Relationen verwendet, um Anfragen zu erweitern. In diesem Beispiel würde man die Anfrage um Konzepte erweitern, welche durch eine „see also“ Relation mit einem der Anfragekonzepte verbunden sind.

Ein prinzipielles Problem bei der Erweiterung von Anfragen ist die Gefahr, die Anfrage zu sehr zu erweitern und dadurch viele nur sehr indirekt relevante Dokumente als Antwort zu erhalten. Um eine solche Verwässerung der Anfrage zu vermeiden, ist es sinnvoll, die Erweiterung nicht automatisch durchzuführen, sondern dem Benutzer Vorschläge zu unterbreiten, wie die Anfrage erweitert werden könnte, um bessere Ergebnisse zu erzielen.

## 7.3 Beispiel: Das DOPE-System

Ziel des DOPE-Projektes war der integrierte und benutzerfreundliche Zugriff auf medizinische Publikationen aus unterschiedlichen Quellen. Besonderes Augenmerk wurde hierbei auf die Möglichkeit einer themenbasierten Suche in den vorhandenen Dokumentenbeständen gelegt. Zu diesem Zweck wurde der von Elsevier entwickelte und vermarktete medizinische Thesaurus EMTREE verwendet. Er enthält ca. 250.000 Begriffe zu unterschiedlichen Aspekten medizinischer Forschung (Krankheiten, Symptome, Medikamente, Behandlungsmethoden etc.), die eine Grundlage für die gezielte Formulierung thematischer

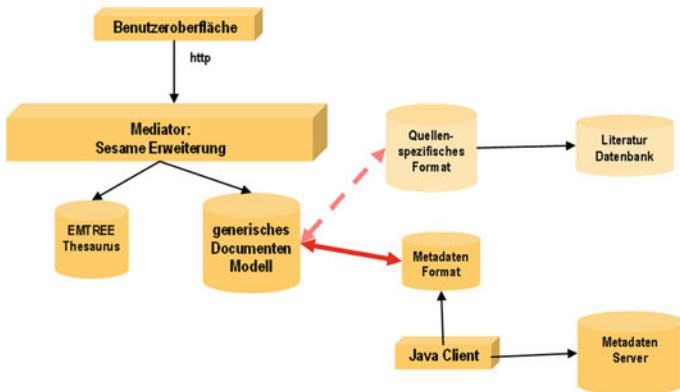
Anfragen bilden. In diesem Zusammenhang wurden in enger Kooperation mit den Entwicklern von EMTREE fundierte Erfahrungen mit dem Aufbau, der Wartung und der Verwendung großer Thesauri in Suchanwendungen gesammelt, die für eine erfolgreiche Bearbeitung des beantragten Projekts notwendig sind. Unter anderem wurde der EMTREE Thesaurus zum Zweck einer einfacheren Integration in Web-basierte Anwendungen in den W3C Sprachstandard RDF übersetzt. Hierfür wurde eine an der Vrije Universiteit Amsterdam entwickelte Methode zur systematischen Konvertierung von Begriffshierarchien verwendet. Um den Einsatz des Thesaurus zu vereinfachen, wurde das entsprechende RDF-Modell mit Hilfe der von der Firma Aduna entwickelten RDF-Datenbank Sesame über ein HTTP-basiertes Web Interface zur Verfügung gestellt. Über dieses Interface können flexibel Anfragen an den Thesaurus gestellt werden. Die automatische Ergänzung von Suchanfragen mit Unter- bzw. Oberbegriffen eines bestimmten Terms wird hierbei auf Grundlage der standardisierten Semantik der Sprache RDF automatisch durchgeführt.

Um die in EMTREE enthaltenen Begriffe als Grundlage für einen themenbasierten Zugriff auf Dokumente verwenden zu können, bestand die Notwendigkeit, die Dokumente mit den entsprechenden Begriffen zu indizieren. Damit die Skalierbarkeit und Erweiterbarkeit des Recherchesystems gewährleistet werden konnte, wurden Methoden zur automatischen Indizierung von Dokumentenbeständen mit Begriffen aus einem gegebenen Thesaurus untersucht. Grundlage der Untersuchung waren 500.000 Volltextartikel aus dem Bestand von Sciencedirect (Elseviers Online-Portal für wissenschaftliche Publikationen) sowie mehr als 10 Millionen Zusammenfassungen aus der Medline Datenbank, einer der wichtigsten Sammlungen medizinischer Publikationen. Die Indizierung wurde mit Hilfe eines Systems der Firma Collexis durchgeführt. Es verwendet die relative

Häufigkeit von Begriffen aus dem Thesaurus in den zu indizierenden Dokumenten, um die Relevanz eines Begriffs im Hinblick auf das Thema eines Dokuments zu bestimmen. In Experimenten auf vergleichbaren Datensätzen wurde gezeigt, dass sich mit diesem Verfahren der gleiche Grad der Vollständigkeit erreichen lässt wie mit den besten Methoden zur Freitextsuche. Durch die Einschränkung auf Thesaurusbegriffe lässt sich die Korrektheit der Suchergebnisse jedoch deutlich verbessern.

Im Rahmen des durch den Elsevier Verlag unterstützten DOPE-Projektes wurde ein System zum Thesaurus-basierten Zugriff auf indizierte Dokumentenbestände implementiert. Kern des Systems bildete eine erweiterte Version der Sesame Datenbank, in der unterschiedliche Thesauri und Metadaten verschiedener Dokumentenbestände verwaltet werden. Basierend auf der Benutzeranfrage wurden relevante Thesaurusbegriffe ermittelt und Metadaten der entsprechenden Dokumente an die Benutzeroberfläche zurückgegeben. Dokumente und deren Metadaten wurden hierbei nicht direkt im System verwaltet, sondern über ein generisches Dokumentenmodell von externen Quellen (Metadaten-Servern oder Literaturdatenbanken) bezogen. Abbildung 7.4 illustriert die Architektur des Systems, die eine einfache Erweiterung um neue Thesauri und Informationsquellen ermöglicht.

Ein weiterer Aspekt, der im Rahmen dieses Projekts eingehend untersucht wurde, ist die gezielte Unterstützung des Benutzers durch die Implementierung geeigneter Suchstrategien und graphischer Darstellung der Suchergebnisse. In diesem Zusammenhang wurde eine spezielle Benutzerschnittstelle für den Thesaurus-basierten Zugriff auf große Dokumentenbestände entwickelt und durch die User-Centered Design Group des Elsevier Verlags in Benutzerstudien evaluiert. Die in der entwickelten Benutzerschnittstelle realisierte Suchstrategie beruht darauf, dass der Benutzer zunächst den Kontext der Suche durch die



**Abb. 7.4** Architektur des DOPE-Systems

Angabe eines generellen Suchbegriffes (z.B. Malaria) einschränkt. Basierend auf den in Bezug auf dieses Thema relevanten Dokumenten schlägt das System dem Benutzer weitere Suchbegriffe zur Eingrenzung der Suche auf spezifischere Themen (z.B. Malaria in Verbindung mit einem speziellen Wirkstoff) vor, zu denen Dokumente vorhanden sind. Die entsprechenden Suchbegriffe werden hierbei dem Thesaurus entnommen und können durch Navigation in den entsprechenden Thesaurusstrukturen ausgewählt werden. Zur Visualisierung der Suchergebnisse in einer Kombination von Suchbegriffen wurde die von der Firma Aduna entwickelte Clustermap Technologie verwendet. Hierbei werden relevante Dokumente in einer interaktiven Karte mit zusätzlichen Interaktionsmöglichkeiten dargestellt. Abbildung 7.5 zeigt ein Suchergebnis mit vier Suchbegriffen im thematischen Kontext „Aspirin“.

In einer Studie wurde das beschriebene Suchinterface im Hinblick auf die Unterstützung des Benutzers bei der



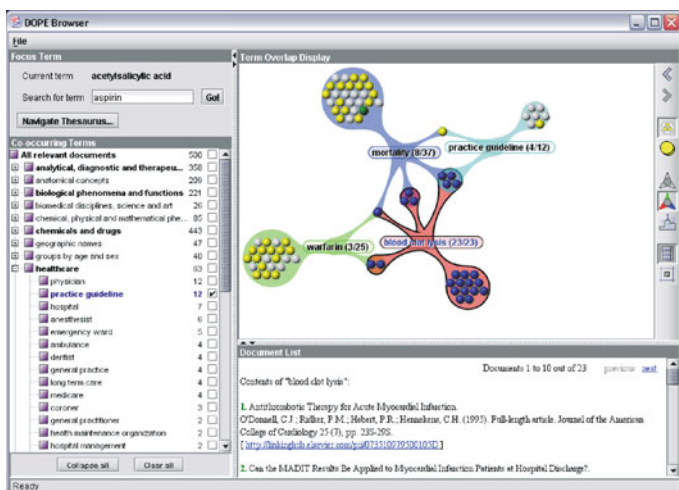


Abb. 7.5 Oberfläche des DOPE-Prototypen

themenbasierten Suche nach Informationen analysiert. Hierzu wurde Besuchern der Konferenz „Drug Discovery Technology 2003“ in Boston das System vorgeführt und den Probanden die Gelegenheit gegeben, das System selbstständig zu nutzen. Abschließend wurden die Probanden nach ihrer Meinung zu Benutzerführung und Ergebnisvisualisierung befragt. Die Mehrzahl der befragten Personen äußerte sich positiv über das System. Als besonders nützlich wurde die automatische Generierung von Vorschlägen zur Einschränkung der Anfrage sowie die grafische Darstellung der Suchergebnisse und der Beziehungen zu den verschiedenen Suchbegriffen erwähnt. Das Prinzip der Thesaurus-basierten Suche wurde als sehr nützlich zur Erschließung unbekannter Dokumentenbestände sowie zur Unterstützung der Lehre – insbesondere als Recherchesystem für Studenten und Doktoranden – eingeschätzt.

## Kapitel 8

# Zusammenfassung und Literatur

In diesem Buch haben wir uns mit Konzepten, Technologien und Anwendungen von Ontologien in der Informationsverarbeitung beschäftigt. Ziel war es hierbei, einen Einstieg in dieses spannende und komplexe Thema zu bieten und nicht den Stand der Technik in diesem Bereich umfassend darzustellen. Wir haben uns hierzu dem Thema Ontologien aus drei unterschiedlichen Richtungen genähert. Im ersten Teil des Buches haben wir uns aus der Sicht des Wissenschaftlers mit den grundlegenden Konzepten und Ideen bei der Verwendung von Ontologien in der modernen Informationsverarbeitung beschäftigt. In [Kapitel 1](#) wurden hierbei die Wurzeln moderner Ontologien in Disziplinen wie der Philosophie und der Linguistik aufgezeigt, um dem Leser ein Verständnis dafür zu vermitteln, welche Ideen zu der Entwicklung von Ontologien im modernen Sinne beigetragen haben und welche Konsequenzen dies für den aktuellen Stand der Technik hat. Hierbei haben wir besonderen Wert auf grundlegende Prinzipien von Kategoriensystemen, wie die Identifikation von Kategorien auf der Grundlage gemeinsamer Eigenschaften von Objekten, sowie die Vererbung von

Eigenschaften, welche sich auch in modernen Ontologieformalismen wiederfinden, gelegt. Dieser Bezug von Ontologien in der Informationsverarbeitung zu grundlegenden Ideen aus der philosophischen Ontologie konnte im Rahmen dieses Buches nur sehr oberflächlich behandelt werden. Eine umfassendere Behandlung dieses Themas findet sich in [Sowa, 2000], wo die ontologischen Grundlagen moderner Wissensrepräsentation aus der Sicht der Semiotik diskutiert werden. Eine detaillierte Diskussion über die Rolle formaler Ontologie für die moderne Informationsverarbeitung kann man in [Guarino, 1995] nachlesen. Als ein weiterer wichtiger Einfluss auf moderne Ontologien wurde die linguistische Semantik diskutiert, die nicht nur Ausgangspunkt vieler Überlegungen in der philosophischen Ontologie war, sondern auch wesentlich die Darstellung ontologischen Wissens in Form von semantischen Netzen und Thesauri beeinflusst hat. Auch das Thema der linguistischen Semantik konnte im Rahmen dieses Buches nur oberflächlich behandelt werden. Eine Diskussion von Semantik aus der Sicht der Linguistik findet sich in [Radford et al., 1999]. Die Nutzung linguistischer Semantik für die automatische Interpretation und Verarbeitung von Text im Zusammenhang mit der sogenannten Computerlinguistik wird in [Jurafsky and Martin, 2008] genauer beschrieben.

Die Betrachtung dieser beiden Wurzeln moderner Ontologien ist notwendig, um die unterschiedlichen Formen ihrer Darstellung zu verstehen, die parallel existieren und in Kapitel 2 vorgestellt werden. Hier finden wir semantische Netze als graphbasierte Repräsentation von Begriffen und deren Zusammenhängen, die heutzutage vor allem zur Darstellung linguistischer Relationen von Begriffen verwendet werden und somit in der Tradition der linguistischen Semantik stehen. Nicht umsonst finden semantische Netze heute vor allem in der Computerlinguistik ihre Anwendung. Unsere Betrachtung semantischer Netze in Kapitel 2.1 beschränkt sich vor allem auf strukturelle Eigenschaften,

ohne dabei allzu detailliert auf die Ausdrucksmächtigkeit sowie die Adäquatheit als Darstellung terminologischen Wissens einzugehen. Solche Betrachtungen finden sich in frühen Beiträgen aus dem Bereich der Wissensrepräsentation, von denen die wichtigsten in [Brachman and Levesque, 1985] zusammengefasst sind. In der Künstlichen Intelligenz wurde vor dem Hintergrund der Uneindeutigkeit von Beschreibungen und dem Fehlen klar definierter und verstandener Inferenzmethoden schon sehr früh eine stärkere Formalisierung semantischer Netze auf der Grundlage der Prädikatenlogik gefordert und vorangetrieben. Entsprechende Ansätze finden sich ebenfalls in [Brachman and Levesque, 1985]. Da sich das Prinzip der Formalisierung von ontologischem Wissen in Form von Konzepten, die durch notwendige und hinreichende Bedingungen auch die Grundlage für moderne Ontologiesprachen, insbesondere der Web Ontology Language, bildet, wurden dieser Ansatz und seine grundlegenden Ideen bereits in Kapitel 2 detailliert beschrieben. Wir sind hierbei davon ausgegangen, dass der Leser im Prinzip mit Prädikatenlogik erster Stufe vertraut ist. Daher wurde diese trotz ihrer zentralen Bedeutung nur sehr kurz und formal eingeführt. Eine sehr viel geeignetere Einführung in die Prädikatenlogik ist das ausgezeichnete Lehrbuch von Uwe Schöning [Schöning, 1995], das ausdrücklich als Lektüre zum Einstieg oder zur vertieften Auffrischung der Logik-Kenntnisse empfohlen wird.

Da die Beschreibungen in Kapitel 4 sich lediglich mit den Prinzipien der Darstellung von ontologischem Wissen anhand von Spielbeispielen beschäftigen, bieten diese kein wirkliches Bild realer Ontologien. Um zu zeigen, wie diese Ideen in der Praxis eingesetzt werden, um reales Wissen in praxisrelevanten Maßstäben darzustellen, haben wir uns in Kapitel 3 mit Beispielen realer Ontologien beschäftigt, die in der Praxis verwendet werden und deren Nutzen bewiesen ist. Hierbei war

es uns wichtig festzustellen, dass sowohl das Prinzip der Darstellung von Wissen in Form eines Graphen ohne formale Semantik als auch die Logik-basierte Spezifikation einen Nutzen hat und verwendet wird, um reale Ontologien zu bauen. Eine andere wichtige Unterscheidung, die hier getroffen wurde, ist die zwischen allgemeinen und domänenspezifischen Ontologien, die ein bestimmtes Anwendungsgebiet im Detail beschreiben. In Bezug auf anwendungsspezifische Ontologien ist hierbei das Gebiet der Medizin bzw. der Biomedizin das mit Abstand am besten entwickelte. Die Auswahl der im Detail vorgestellten Ontologien beruhte hierbei zum Teil auf deren Popularität, zum Teil aber auch auf der vorhandenen Dokumentation und meinem eigenen Wissen über Aufbau und Verwendung der entsprechenden Modelle. Im Bereich linguistisch motivierter Modelle, die als semantische Netze dargestellt werden, haben wir uns mit WordNet und UMLS die für wohl wichtigsten Modelle für die Analyse und Verarbeitung textueller Informationen entschieden. Aufgrund der häufigen Verwendung dieser Modelle bilden die Webseiten (<http://wordnet.princeton.edu/> und <http://www.nlm.nih.gov/research/umls/>) wohl die besten Informationsquellen hierzu. Die entsprechenden Publikationen ([Fellbaum, 1998] bzw. [Lindberg et al., 1993]) sind bereits etwas älter und daher wohl eher aus historischer Sicht interessant. Bei den Logik-basierten Ontologien haben wir uns auf die Vorstellung der SUMO-Ontologie [Niles and Pease, 2001] beschränkt. Hier bietet wiederum die entsprechende Webseite (<http://www.ontologyportal.org/>) die beste Informationsquelle. Die Auswahl dieser Ontologie als Beispiel kann hierbei diskutiert werden, da eine Reihe weiterer wichtiger Ontologien existieren, die ebenso gut hätten vorgestellt werden können. Allen voran sind dies die Cyc-Ontologie [Lenat and Guha, 1989], die seit kurzem in eingeschränkter Form frei verfügbar ist (<http://www.opencyc.org/>, <http://research.cyc.com>), sowie die DOLCE

Bibliothek allgemeiner Ontologien [Masolo et al., 2003], die im Rahmen des europäischen Projekts WonderWeb entwickelt wurde. Im Bereich domänenspezifischer Ontologien sind die sogenannten Open Biomedical Ontologies [Smith et al., 2007] eine sehr interessante Ressource, die unterschiedlichste Ontologien aus der Biomedizin zusammenfasst und zum Teil integriert. All diese Ontologien zu beschreiben hätte jedoch den Rahmen dieses Buches gesprengt.

Die zweite grundlegende Richtung, aus der wir uns in diesem Buch dem Thema Ontologien genähert haben, ist die aus der Sicht des Systementwicklers, der selbst eine Ontologie für einen bestimmten Zweck erstellen will. Diese Sicht ist Thema von Teil II des Buches. In diesem Teil haben wir uns in Kapitel 4 zunächst mit aktuellen Sprachen zur Repräsentation von Ontologien beschäftigt, wobei es gewisse Überschneidungen mit aktuellen Lehrbüchern zum Thema Semantic Web Technologien, insbesondere [Antoniou and van Harmelen, 2008] und [Hitzler et al., 2008], gibt. Im Unterschied zu den genannten Büchern haben wir den Fokus der Betrachtung auf die logischen Grundlagen aktuell verwendeter Sprachen wie RDF, OWL und F-Logic gelegt und den Aspekt der Einbettung in Web Technologien, insbesondere XML, der in den genannten Büchern sehr prominent ist, vollständig ausgeblendet. Es war uns wichtig zu zeigen, dass die genannten Sprachen, die zurzeit die wichtigsten Repräsentationsformalismen für Ontologien darstellen, als Teilsprachen der Prädikatenlogik mit unterschiedlichen Schwerpunkten gesehen werden können. Insbesondere haben wir versucht darzustellen, dass zwei spezifische Teilmengen der Prädikatenlogik, nämlich Logik-Programme und Beschreibungslogiken, die Grundlage der gebräuchlichen Sprachen bilden. Dabei haben wir gezeigt, wie sich RDF und F-Logic als Logik-Programme und OWL als Beschreibungslogik interpretieren lassen. Hat man diese beiden grundlegenden

Formalisten verstanden, so ist die Beherrschung der entsprechenden Ontologiesprachen kaum zusätzlicher Aufwand. Während RDF und OWL in den bereits genannten Lehrbüchern ausführlich behandelt werden, gibt es zu F-Logic weniger geeignete Quellen, da die aktuell verwendete Version der F-Logic sich doch erheblich von der ursprünglich vorgeschlagenen Version [Kifer et al., 1995] unterscheidet. Ein sehr praktisches Tutorial findet sich auf der Webseite der Firma Ontoprise.<sup>1</sup> Das Verhältnis der dort verwendeten Variante von F-Logic zur Prädikatenlogik ist in [de Bruijn and Heymans., 2008] beschrieben. Eine umfassende Behandlung von Beschreibungslogiken, welche die Grundlage für die Web Ontology Language bilden, ist [Baader et al., 2003]. Die Übersetzung von Beschreibungs- in Prädikatenlogik wurde in [Borgida, 1996] untersucht.

Als weiteren wichtigen Aspekt bei der Erstellung von Ontologien haben wir uns in Kapitel 5 mit dem Vorgehen und den Werkzeugen zur Erstellung von Ontologien beschäftigt. Diese ergänzen die Beschreibung der verwendeten Sprachen, da sowohl das Vorgehen bei der Erstellung als auch die Benutzung entsprechender Editoren ein gewisses Verständnis der verwendeten Ontologiesprache voraussetzen. Aufgrund der weiten Verbreitung, sowohl der Sprache als auch der entsprechenden Werkzeuge, haben wir uns hierbei auf Methoden und Werkzeuge beschränkt, welche die Web Ontology Language OWL und deren Besonderheiten und Möglichkeiten unterstützen. Grundlage der vorgestellten Prinzipien sind zum einen das ausgezeichnete Tutorial von Noy und McGuinness [Noy and McGuinness, 2005], welches ein grundlegendes Vorgehen bei der Erstellung von OWL Ontologien vorstellt, sowie die Ergebnisse des CO-ODE Projektes und insbesondere der in [Rector et al., 2004] beschriebenen Erkenntnisse über typische

---

<sup>1</sup> <http://www.ontoprise.de/de/deutsch/start/produkte/ontobroker.html>

Probleme und Fehler bei der Erstellung von Ontologien. Die Unterstützung der Ontologieerstellung durch Software-Werkzeuge wurde weitestgehend auf der Grundlage des Protege-Systems [Gennari et al., 2003] sowie des speziellen OWL-Plugins für dieses System [Knublauch et al., 2004] beschrieben. Diese Fokussierung auf OWL stellt eine gewisse Einschränkung der Allgemeingültigkeit der vorgestellten Methoden dar, bietet sich jedoch an dieser Stelle an. Für Leser, die an einer breiteren Diskussion um die Erstellung von Ontologien interessiert sind, sei das Buch von Gomez-Perez und anderen [Gomez-Perez et al., 2004] empfohlen.

Die dritte Richtung, aus der wir Ontologien betrachtet haben, ist die des Endanwenders, der mit bestehenden Ontologien die Funktionalität einer bestimmten Anwendung verbessern will. In diesem Kontext haben wir uns in Teil III des Buches mit typischen Anwendungen von Ontologien beschäftigt und deren Funktion und Nutzen diskutiert. Obwohl Ontologien inzwischen in viele unterschiedliche Anwendungsbereiche Einzug gehalten haben, beschränkte sich die Betrachtung auf zwei Gebiete, in denen Ontologien traditionell eingesetzt werden. In Kapitel 6 haben wir zunächst die Anwendung von Ontologien im Bereich der Datenintegration betrachtet. Ausgangspunkt war hierbei eine detaillierte Beschreibung möglicher Konflikte und Probleme, die bei der Integration heterogener Datenbestände auftreten können. Die Klassifikation dieser Konflikte sowie ein spezieller Ansatz zu ihrer Lösung findet sich in [Wache, 2003]. Im Anschluss haben wir die Verwendung von Ontologien in der Datenintegration beschrieben. Hierbei haben wir vor allem zwei Funktionen von Ontologien, nämlich als neutrale anwendungsunabhängige Datenstruktur und als Werkzeug zur Beschreibung impliziter Annahmen, in den Vordergrund gestellt, da diese am häufigsten auftreten. Der beschriebene Ansatz vereinfacht hierbei jedoch stark und klammert viele reale Probleme der Datenintegration



aus. Ein vollständigeres Bild der Verwendung von Ontologien bei der Datenintegration bieten neben der oben genannten Arbeit von Waché auch [Stuckenschmidt and van Harmelen, 2004] sowie [Visser, 2005], die sich mit der Verwendung von Semantic Web Technologien und insbesondere OWL zur Lösung von Integrationsproblemen beschäftigen. Der Fokus von [Stuckenschmidt and van Harmelen, 2004] liegt hierbei auf der Integration von Informationen in schwach strukturierten Umgebungen wie dem Web, [Visser, 2005] thematisiert neben der Integration von terminologischen auch räumliche und zeitliche Informationen, deren Integration spezifische Lösungen erfordert.

Als zweite typische Anwendung von Ontologien haben wir uns in Kapitel 7 mit der inhaltsbasierten Suche nach Dokumenten beschäftigt. Die Wahl dieses Themas ist zum einen darin begründet, dass bereits in den siebziger Jahren, also bevor das Thema Ontologien in der Künstlichen Intelligenz verstärkt Beachtung fand, Begriffsnetze, die semantischen Netzen ähneln, zur Unterstützung der Dokumentensuche eingesetzt wurden. Zum anderen macht diese Anwendung deutlich, dass beide Formen von Ontologie, die im ersten Teil des Buches eingeführt wurden, ihre Existenzberechtigung haben: Während in der Datenintegration meist Logik-basierte Ontologien zum Einsatz kommen, beruht die Ontologie-basierte Dokumentensuche zum Großteil auf Begriffsnetzen, die dem Prinzip eines semantischen Netzes folgen. In Kapitel 7 haben wir zunächst den Stand der Technik in Bezug auf die Relevanzbewertung von Dokumenten mit Hilfe des TF-IDF Maßes sowie des Vektorraum-Modells für Dokumente und Anfragen vorgestellt. Eine detailliertere Beschreibung der entsprechenden Technologien finden sich in entsprechenden Lehrbüchern (zum Beispiel [Grossman and Frieder, 1998]). Anschließend haben wir auf mögliche Probleme des Modells hingewiesen, welche darauf zurückzuführen sind, dass die Bedeutung von Begriffen

nicht ausreichend berücksichtigt wird, und haben erklärt, wie die Verwendung von Thesauri diese Probleme lösen kann. Als Beispiel haben wir hierbei den MeSH-Thesaurus, einen der meistgenutzten Modelle im Bereich der Medizin, verwendet. Eine genauere Beschreibung der Nutzung von MeSH für die Informationssuche ist [Coletti and Bleich, 2001]. Als ein weiteres konkretes Beispiel haben wir das DOPE-System betrachtet, welches den Zugang zu unterschiedlichen Dokumentenbeständen auf der Grundlage eines Thesaurus realisiert und hierzu zusätzlich auf Semantic Web Technologien wie RDF zurückgreift. Eine umfassendere Beschreibung dieses Systems findet sich in [Stuckenschmidt et al., 2004].

Abschließend ist festzuhalten, dass es aufgrund des Umfangs und der Natur dieses Buches nicht möglich und auch nicht sinnvoll war, einen umfassenden Überblick über Forschung und Technik im Umfeld von Ontologien zu geben. Diese Funktion wird von dem ebenfalls im Springer Verlag erschienenen Handbook of Ontologies [Staab and Studer, 2004] erfüllt. Die Beiträge dieses Handbuches decken eine Reihe von Themen ab, die hier nicht erwähnt wurden. Die zweite Auflage des Handbuches ist 2009 erschienen und bietet allen, die nach der Lektüre dieses Buches mehr über Ontologien wissen wollen, eine ideale Quelle für weitere Informationen.

## Literaturverzeichnis

- [Antoniou and van Harmelen, 2008] Antoniou, G. and van Harmelen, F. (2008). *A Semantic Web Primer*. MIT Press, 2nd edition edition.
- [Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook – Theory, Implementation and Applications*. Cambridge University Press.

- [Borgida, 1996] Borgida, A. (1996). On the relative expressiveness of description logics and predicate logics. *Artif. Intell.*, 82(1-2):353–367.
- [Brachman and Levesque, 1985] Brachman, R. and Levesque, H., editors (1985). *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos.
- [Coletti and Bleich, 2001] Coletti, M. and Bleich, H. (2001). Medical subject headings used to search the biomedical literature. *J Am Med Inform Assoc*, 8:317–323.
- [de Bruijn and Heymans., 2008] de Bruijn, J. and Heymans., S. (2008). On the relationship between description logic-based and f-logic-based ontologies. *Fundamenta Informaticae*, 82(3):213–236.
- [Fellbaum, 1998] Fellbaum, C., editor (1998). *WordNet – An Electronic Lexical Database*. MIT Press.
- [Gennari et al., 2003] Gennari, J., Musen, M., Ferguson, R., Grosso, W., Crubézy, M., Eriksson, H., Noy, N., and Tu, S. (2003). The evolution of protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123.
- [Gomez-Perez et al., 2004] Gomez-Perez, A., Corcho-Garcia, O., Fernandez-Lopez, M., Corcho, O., and Fernández-López, M. (2004). *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Springer-Verlag.
- [Grossman and Frieder, 1998] Grossman, D. A. and Frieder, O. (1998). *Information Retrieval: Algorithms and Heuristics*. Springer International Series in Engineering and Computer Science. Springer-Verlag.
- [Guarino, 1995] Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human and Computer Studies*, 43(5/6):625–640.
- [Hayes, 2004] Hayes, P. (2004). Rdf semantics. Recommendation, W3C.
- [Hitzler et al., 2008] Hitzler, P., Krötzsch, M., Rudolph, S., and Sure, Y. (2008). *Semantic Web – Grundlagen*. Springer-Verlag.
- [Jurafsky and Martin, 2008] Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing – An Introduction to Natural Language Processing, Linguistics, and Speech Recognition*. Prentice Hall Series in Artificial Intelligence. Prentice-Hall, 2nd edition.
- [Kifer et al., 1995] Kifer, M., Lausen, G., and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843.
- [Knublauch et al., 2004] Knublauch, H., Ferguson, R., Noy, N., and Musen, M. (2004). The protégé owl plugin: An open development environment for semantic web applications. In *The Semantic Web – ISWC 2004*, volume 3298 of *Lecture Notes in Computer Science*. Springer-Verlag.

- [Lenat and Guha, 1989] Lenat, D. and Guha, R. (1989). *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [Lindberg et al., 1993] Lindberg, D., Humphreys, B., and McCray, A. (1993). The unified medical language system. *Methods Inf Med.*, 32(4):281–291.
- [Masolo et al., 2003] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., and Schneider, L. (2003). The wonderweb library of foundational ontologies. WonderWeb Deliverable D17, ISTC-CNR.
- [Niles and Pease, 2001] Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In Welty, C. and Smith, B., editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*.
- [Noy and McGuinness, 2005] Noy, N. and McGuinness, D. (2005). Ontology development 101: A guide to creating your first ontology. Ksl technical report, Stanford University, KSL-01-05.
- [Radford et al., 1999] Radford, A., Atkinson, M., Britain, D., Clahsen, H., and Spencer, A. (1999). *Linguistics – An Introduction*. Cambridge University Press.
- [Rector et al., 2004] Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., and Wroe, C. (2004). Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *Proceedings of the European Conference on Knowledge Acquisition*, volume 3257 of *Lecture Notes on Computer Science*, pages 63–81.
- [Schöning, 1995] Schöning, U. (1995). *Logik für Informatiker*. Spektrum Akademischer Verlag, 4. Auflage.
- [Smith et al., 2007] Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L., Eilbeck, K., Ireland, A., Mungall, C., The OBI Consortium, Leontis, N., Rocca-Serra, P., Alan Ruttenberg, S.-A. S., Scheuermann, R., Shah, N., and abd Suzanna Lewis, P. W. (2007). The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25:1251–1255.
- [Sowa, 2000] Sowa, J. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA.
- [Staab and Studer, 2004] Staab, S. and Studer, R., editors (2004). *Handbook on Ontologies*. International Handbooks on Information Systems. Springer-Verlag.

- [Stuckenschmidt et al., 2004] Stuckenschmidt, H., de Waard, A., Bhogal, R., Fluit, C., Kampman, A., van Buel, J., van Mulligen, E., Broekstra, J., Crowlesmith, I., van Harmelen, F., and Scerri, T. (2004). Exploring large document repositories with rdf technology – the dope project. *IEEE Intelligent Systems*, 19(3):34–40.
- [Stuckenschmidt and van Harmelen, 2004] Stuckenschmidt, H. and van Harmelen, F. (2004). *Information Sharing on the Semantic Web*. Springer-Verlag.
- [Visser, 2005] Visser, U. (2005). *Intelligent Information Integration for the Semantic Web*, volume 3159 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.
- [Wache, 2003] Wache, H. (2003). *Semantische Mediation für heterogene Informationsquellen*. Dissertationen zur Künstlichen Intelligenz. DISKI Verlag.

# Sachverzeichnis

- Übersetzung
  - von F-logic, [121](#)
  - von Konzeptausdrücken, [137](#)
  - von OWL Relationen, [153](#)
- Ableitungsregel, [42](#), [46](#), [83](#), [109](#),  
[116](#), [123](#)
- Anfrageerweiterung, [252](#)
- Anfragen, [199](#)
- Anfragenormalisierung, [250](#)
- Antonym, [21](#), [63](#)
- Ausdrucksmächtigkeit, [83](#),  
[100](#), [130](#)
- Begriff
  - präferierter, [244](#)
- Beschreibungsmuster, [172](#)
- Closed World, [32](#)
- Competency Questions, [159](#)
- Diagnose, [204](#)
- differentiae, [14](#), [168](#)
- Disambiguierung, [70](#)
- Dokumentensuche, [234](#)
- DOPE, [253](#)
- Dreieck
  - semiotisches, [8](#)
- Eigenschaften
  - charakteristische, [29](#)
  - extrinsische, [169](#)
  - hinreichende, [14](#), [46](#), [140](#)
  - intrinsische, [168](#)
  - notwendige, [14](#), [46](#), [139–140](#)
- Erfüllbarkeit, [143](#), [202](#)
- Formular, [193](#)
- Graph, [27](#), [103](#), [186](#)
- Hintergrundwissen, [225](#)
- Homonym, [20](#)
  - Einfluss auf Relevanz, [242](#)
- Hyponym, [21](#), [58](#), [73](#), [89](#)

- Inferenz, 198
  - in Beschreibungslogiken, 143
  - linguistische, 21
  - logische, 41
- Integration
  - Daten-, 211
  - von Vokabularen, 68
- Kategorien, 29
- Kategoriensystem, 12, 84
- Keyword Density, 237
- Klassifikation, 144, 178, 201
- Konflikt, 204
  - semantischer, 124, 217
  - strukturell, 214
  - syntaktischer, 212
- Konsistenz, 171, 227
- Konzeptausdruck, 132, 150
- Konzeptbeschreibungen
  - Manipulation von, 192
- Konzeptdefinition, 191
- Konzepte
  - definierte, 177
- Konzepthierarchie, 164, 186
- Lexeme, 19
- Lloyd-Topor Transformation, 121
- Logik, 34
  - Beschreibungs-, 127
  - Prädikaten-, 36, 45, 81, 104, 121, 137
  - programme, 99, 105
- Mereonym, 60
- MILO, 78
- Modellierung, 156
- Modellierungsmuster, 194
- Morphologie, 19
- Ockhams Rasiermesser, 16
- OntoBroker, 229
- Ontologie
  - als Datenmodell, 95
  - Logik-basierte, 44
  - philosophische, 8
- Ontologie Editor, 184
- Ontologie-Bibliotheken, 161
- Ontologien
  - allgemeine, 54
  - domänen-spezifische, 54
- Ontologiesprache, 96
  - F-Logic, 115, 229
  - OWL, 146
  - RDF, 102, 129, 254
- RDF
  - Literale, 105
  - Vokabular, 105
- Relation, 73, 151, 188
  - is-a, 31, 33, 42
  - kind-of, 33
  - mathematische Eigenschaften, 86
  - n-stellige, 181
  - Subsumption, 138, 165
  - Teil-Ganzes, 61, 182
- Relationen
  - Definition von, 166
  - in SUO-KIF, 80
- Relevanz
  - maß, 236
  - von Dokumenten, 235
- Semantik
  - der Prädikatenlogik, 37
  - linguistische, 21, 56
  - mengentheoretische, 131
  - von RDF, 109
- semantisches Netz, 29, 41, 55
  - UMLS, 74
- Subsumption, 144

- SUMO, [77](#)
  - Verbindung zu WordNet, [88](#)
- Synonym, [57](#), [73](#), [88](#)
  - Einfluss auf Relevanz, [242](#)
- TF-IDF, [237](#)
- Thesaurus, [244](#)
  - EMTREE, [253](#)
  - MeSH, [66](#), [245](#)
  - Meta-, [67](#)
- UML, [197](#)
- UMLS, [65](#)
- Universalien, [9](#)
- Value Partition, [175](#)
- Vektorraum-Modell, [239](#)
- Visualisierung, [186](#), [256](#)
- Vorgehensmodell, [156](#)
- Web Ontology Language,  
[146](#)
  - Syntax, [147](#)
- Wertebereiche
  - Beschreibung von, [173](#)
- Wiederverwendung, [160](#)
- Wissensbasis, [142](#)
- WordNet, [55](#), [88](#)