
ÜBER DIE REPRÄSENTATION VON
RÄUMLICH-TERMINOLOGISCHEM WISSEN
DURCH BESCHREIBUNGSLOGISCHE
DEFAULT-THEORIEN
MIT SPEZIFITÄT

OMAR SOLTAN NURI

25. OKTOBER 1999

DIPLOMARBEIT
AM FACHBEREICH INFORMATIK
DER UNIVERSITÄT HAMBURG

ERSTBETREUER:
DR. RALF MÖLLER
ARBEITSBEREICH KOGNITIVE SYSTEME
ZWEITBETREUER:
PROF. DR. CHRISTOPHER HABEL
ARBEITSBEREICH WISSENS- UND
SPACHVERARBEITUNG

Inhaltsverzeichnis

1	Einleitung	4
2	Grundlagen	10
2.1	Terminologische Wissensrepräsentation	10
2.1.1	Inferenzdienste	13
2.1.2	Die Sprache \mathcal{ALC}	15
2.1.3	Die Sprache $\mathcal{ALC}(\mathcal{D})$	18
2.1.4	Die Sprache $\mathcal{ALCRP}(\mathcal{D})$	22
2.1.5	Räumlich-Terminologisches Schließen mit $\mathcal{ALCRP}(\mathcal{S}_2)$	26
2.1.6	Zusammenfassung	29
2.2	Default-Schließen	30
2.2.1	Motivation	31
2.2.2	Default-Logik und einige andere nichtmonotone Logiken	32
2.2.3	Reiters Default-Logik und der Begriff der Extension . .	33
3	Algorithmen zur Berechnung von Extensionen	37
3.1	Der definitionsbasierte Algorithmus	38
3.2	Das Theorem von Schwind und Risch	39
3.3	Algorithmus von Baader und Hollunder	41

<i>INHALTSVERZEICHNIS</i>	3
4 Default-Schließen mit Prioritäten	48
4.1 Priorisierte Default-Theorien	48
4.2 Die Berechnung von S-Extensionen	51
5 Räumliches Schließen mit Defaults	56
5.1 Räumlich-terminologische Default-Theorien	57
5.1.1 Erweiterung und Anpassung	61
5.2 Beispiele	65
6 Optimierungstechniken	69
6.1 Instantiieren der Defaults	69
6.2 Gegenüberstellung: Qualitativer Vergleich der Algorithmen . .	72
6.2.1 Maximal erfüllbare Teilmengen	74
7 Zusammenfassung	77

Kapitel 1

Einleitung

Die *Beschreibungslogiken (terminologischen Logiken)* sind eine Familie von Formalismen zur Repräsentation von konzeptuellem Wissen. Sie sind aus den früheren Ansätzen zur Wissensrepräsentation, wie den Semantischen Netzen und Frames entstanden, und besitzen im Gegensatz zu diesen eine formale Semantik, mit der das Auftreten von Mehrdeutigkeiten ausgeschlossen wird und Widersprüchlichkeiten entdeckt werden können. Die Semantik einer Beschreibungslogik kann mengentheoretisch spezifiziert werden, ähnlich der Tarskischen Semantik für die Prädikatenlogik. *Terminologische Systeme* implementieren Beschreibungslogiken und stellen den Benutzern Systemdienste zur Verfügung, die es erlauben, aus dem repräsentierten Wissen Schlüsse zu ziehen, d.h. es werden Fakten explizit gemacht, die vorher nur implizit vorhanden waren. Somit lassen sie sich als spezielle Theorembeweiser (für die jeweilige Logik) auffassen. Als erstes terminologisches System mit wohldefinierter Semantik gilt *KL-ONE*. Schon in *KL-ONE* gab es die Unterscheidung zwischen *konzeptuellem (terminologischen oder begrifflichem) Wissen* und *Objektwissen*. Das konzeptuelle Wissen wird in der *TBox* in Form von Begriffsdefinitionen (sog. *Konzepten*) ausgedrückt. Die für die Anwendung relevanten Objekte (sog. *Individuen*) werden in der *ABox* eingeführt und miteinander über *Rollen (Relationen)* in Beziehung gesetzt. Individuen sind i.d.R. Instanzen bestimmter Konzepte in der *TBox*. Aufgrund der bis heute bestehenden Zweiteilung in *TBox* und *ABox* werden terminologische Systeme auch als *hybride* Wissensrepräsentationssysteme bezeichnet. Auch in den durch die Systeme implementierten Beschreibungslogiken findet sich die Trennung zwischen *TBox* und *ABox*. Zum einen ist diese Trennung aus der Sicht der Benutzer sinnvoll; zum anderen erhofft man sich, bestimmte Inferenzdienste effizienter implementieren zu können.

Viele terminologische Logiken erlauben lediglich die Verwendung von sog. *primitiven Rollen* (z.B. die Logik \mathcal{ALC} , s.u.). Primitive Rollen sind (wie auch *atomare Konzepte*) nicht weiter definiert, sodaß die Eigenschaften der entsprechenden Relationen nicht repräsentiert werden (z.B. Transitivität). Offensichtlich ist adäquates Schlußfolgern über diese Relationen dann nicht mehr möglich, sodaß nicht-intendierte Modelle möglich werden. Das Schließen über raum-zeitliche Gegebenheiten erfordert *definierte Rollen* (auch komplexe Rollen genannt), sodaß die Eigenschaften der repräsentierten Domäne Raum bzw. Zeit adäquat beim Inferieren berücksichtigt werden. So ist z.B. die qualitative räumliche Relation *contains* („enthält“) transitiv: Eine Beschreibung wie $\text{contains}(a, b), \text{contains}(b, c), \neg \text{contains}(a, c)$ kann nur dann als inkonsistent erkannt werden, wenn die Transitivität der *contains*-Relation berücksichtigt wird. Als Ausgangspunkt dieser Arbeit dient die Beschreibungslogik $\mathcal{ALCRP}(\mathcal{D})$, die u.a. solche definierten Rollen bietet.

Das Modellieren von räumlichem oder zeitlichem Wissen ist eine der wichtigsten Aufgaben in der KI. Eine Beschreibungslogik, die das Schlußfolgern über räumliche Gegebenheiten unterstützt, ist die Sprache $\mathcal{ALCRP}(\mathcal{S}_2)$, wobei es sich um eine spezielle Instanz von $\mathcal{ALCRP}(\mathcal{D})$ handelt. \mathcal{S}_2 ist eine spezielle sog. *konkrete Domäne* zum qualitativen räumlichen Schließen. $\mathcal{ALCRP}(\mathcal{S}_2)$ unterstützt die sog. RCC-8-Relationen, mit welchen räumliche Beziehungen zwischen spez. topologischen Gebieten bzw. Region (sog. regulär abgeschlossenen Teilmengen von \mathbb{R}^2) beschrieben werden können. Die RCC-8-Basisrelationen sind erschöpfend (d.h. für jede mögliche räumliche Beziehung zwischen zwei Objekten existiert eine beschreibende Basisrelation) und paarweise disjunkt (d.h. keine zwei Basisrelationen beschreiben die gleiche räumliche Beziehung). Daher kann jegliche räumliche Beziehung zwischen zwei Regionen beschrieben werden. Vergleichbares findet man beim zeitlichen Schließen, die sog. Allensche Relationen, wobei es sich um einen „verwandten“ Formalismus handelt. Eine Instantiierung von $\mathcal{ALCRP}(\mathcal{D})$ mit einer konkreten Domäne zum temporalen Schließen mit Hilfe des Allenschen Kalküls ist ebenfalls denkbar. In [7] wird gezeigt, das $\mathcal{ALCRP}(\mathcal{D})$ eine geeignete Basis zur Integration raum-zeitlicher Inferenzmethoden in Beschreibungslogiken darstellt, wenn sie mit passenden konkreten Domänen instantiiert wird.

In dieser Arbeit wird untersucht, inwiefern die sog. Reiterschen Default-Regeln mit der Basislogik $\mathcal{ALCRP}(\mathcal{S}_2)$ in einem räumlichen Kontext verwendet werden können, um verschiedene Anwendungsprobleme zu lösen. Die Anwendungsprobleme werden durch Zuhilfenahme spez. nicht-monotoner Inferenztechniken behandelt, die eine echte Erweiterung terminologischer Stan-

dardinferenzen (wie Subsumptionsbestimmung und ABox-Realisation) darstellen. Insbesondere ist man daran interessiert, eine bereits in Form einer ABox gegebene Weltbeschreibung um weitere Assertionen automatisch zu ergänzen, wobei es sich um Konklusionen von sog. Default-Regeln handelt. Diese Ergänzungen sollen natürlich wiederum konsistente (also widerspruchsfreie) mögliche Weltbeschreibungen sein. Die Anwendbarkeit einer Default-Regel wird anhand ihrer Vorbedingung (diese muß ableitbar sein) und ihren sog. Konsistenzannahmen entschieden. Die Konsistenzannahmen sollen verhindern, daß durch die Anwendung einer Default-Regel eine offensichtlich inkonsistente Weltbeschreibung entsteht. Die Konklusionen von Default-Regeln könnten als Hypothesen oder Vermutungen angesehen werden, da sie auch dann zur Weltbeschreibung hinzugefügt werden können, wenn diese im Sinne der klassischen Logik nicht aus der Weltbeschreibung folgerbar bzw. ableitbar sind. Im Sinne der klassischen Logik ist letztlich alles ableitbare Wissen (die logische Theorie) bereits implizit vorhanden, d.h., durch die Hinzunahme von Theoremen als weitere Axiome für den Theorembeweiser werden keine wirklich „neuen“ Fakten oder Aussagen inferiert. Durch die Hinzunahme von Default-Regeln können jedoch neue Axiome bzw. Assertionen als Konklusionen dieser Default-Regeln hinzugefügt werden, die im klassischen Sinne nicht ableitbar (bzw. folgerbar) sind. Im klassisch-logischen Sinne handelt es sich daher um eine nicht-korrekte Inferenz.

Insbesondere ist diese Hinzunahme von nicht-folgerbaren Assertionen dann wünschenswert, wenn unvollständige Weltbeschreibungen vorliegen und mögliche Erweiterungen dieser Weltbeschreibung, sog. *Extensionen* (auch „mögliche Welten“ genannt) gesucht werden. Unterschiedliche Extensionen einer *Default-Theorie* entstehen immer dann, wenn mehrere einander widersprechende Vervollständigungen einer Weltbeschreibung anhand der gegebenen Default-Regeln existieren. Dann besteht die Möglichkeit, sich für eine von evtl. mehreren einander widersprechenden Hypothesen zu entscheiden. Offensichtlich ist es sinnvoll zu fordern, daß eine Extension immer widerspruchsfrei und in gewisser Weise maximal ist, so daß alles in dieser Welt unter der Default-Theorie folgerbare auch abgeleitet wurde. Da Default-Regeln interagieren können und durch die Konklusionen von bereits angewendeten Default-Regeln evtl. Vorbedingungen oder Konsistenzannahmen anderer Default-Regeln invalidiert werden, ist die Default-Logik im Gegensatz zu klassischen Logiken *nicht-monoton*.

In dieser Diplomarbeit wird gezeigt, daß Reitersche Default-Regeln zur Behandlung unvollständiger Informationen über räumliche Gegebenheiten gewinnbringend eingesetzt werden und interessante Anwendungsprobleme lösen

können. Im Rahmen dieser Arbeit wurden verschiedene Algorithmen zur Berechnung von Extensionen für räumlich-terminologische Default-Theorien implementiert und qualitativ miteinander verglichen, sowie Problempunkte herausgestellt. Die Synthese einer Beschreibungslogik mit Reiterschen Default-Regeln in einem raum-zeitlichen Kontext ist, sofern mir bekannt, noch nicht eingehend untersucht worden. Hingegen wurden Default-Regeln in terminologischen nicht-räumlichen Kontexten bereits eingehend untersucht. Die bereits geleisteten Arbeiten auf dem Gebiet des terminologischen Default-Schließens wurden angepaßt und erweitert, um eine adäquate Behandlung räumlicher Gegebenheiten zu ermöglichen. Unter anderem ist es nicht nur notwendig, Konzeptzugehörigkeiten von Individuen durch die Anwendung von Default-Regel-Konklusionen zu hypothetisieren, *sondern auch Relationszugehörigkeiten zwischen Objektpaaren, insbesondere räumliche Relationen zwischen Individuen (etwa RCC-8-Relationen) hinzuzufügen*. Dies ist die wesentliche im Rahmen dieser Arbeit vorgenommene Erweiterung, da in bisherigen Arbeiten zum Thema terminologische Default-Theorien lediglich Konzeptzugehörigkeiten von Individuen betrachtet und behandelt wurden.

Ein Beispiel soll die Anwendung Reiterscher Default-Regeln in räumlichen Kontext verdeutlichen, wie er mit $\mathcal{ALCRP}(\mathcal{S}_2)$ repräsentiert werden könnte:

Wir nehmen an, daß uns gewisse räumliche Objekte, wie Haus, Zimmer, Schlafzimmer, Wohnzimmer, Bad, Flur, Küche, Schrank, u.s.w., und ebenfalls räumliche Beziehungen zwischen diesen bekannt sind. Diese Weltbeschreibung würde in der sog. ABox stehen. Anhand der Fakten in der ABox könnten wir mit Hilfe von Default-Regeln Aussagen über die Objekte unseres Szenarios machen. Wenn ein Objekt sich sowohl mit dem Flur als auch mit dem Wohnzimmer überlappt, dann könnte per Default geschlossen werden, daß es sich um einen Teppich handelt. Befindet sich ein Möbelstück in der Küche, dann könnte per Default geschlossen werden, daß es sich um einen Küchenschrank handelt. Ebensogut könnte es sich jedoch auch um einen Küchentisch handeln. Sicherlich sollten die Konzepte Küchentisch und Küchenschrank disjunkt sein. Dies verdeutlicht die Idee der *disjunkten* konsistenten Extensionen: das Möbelstück kann nur ein Schrank oder ein Tisch sein, aber nicht beides. Natürlich könnte man auch eine hinreichende Bedingung für das Konzept „Küchenschrank“ in der sog. TBox haben, etwa: „Ein Objekt ist genau dann ein Küchenschrank, wenn es sich um einen Schrank handelt und dieser enthalten ist in einer Küche“. Dann könnte die Inferenz, daß es sich um einen Küchenschrank handelt, auch durch konventionelle logische Deduktion erzielt werden. Wir müßten jedoch bereits wissen, daß es sich um einen Schrank handelt, und nicht nur um ein Möbelstück. Für

viele Konzepte läßt sich jedoch kein Satz von hinreichenden und notwendigen Bedingungen finden, und oftmals ist die Weltbeschreibung in der ABox unvollständig. Betrachten wir jetzt den umgekehrten Fall. Zuvor haben wir anhand von bekannten Objektbeziehungen Konzeptzugehörigkeiten inferiert. Diesesmal wissen wir wenig über die Beziehungen der Objekte untereinander. Anhand der Kenntnis von Konzeptzugehörigkeiten von Objekten können wir jedoch sinnvolle Beziehungen (also Relationszugehörigkeiten) zwischen zwei Objekten per Default inferieren. Wissen wir z.B. von zwei Objekten daß es sich um einen Kühlschrank und eine Küche handelt, so ist es sinnvoll, durch die Anwendung einer Default-Regel zu schließen, daß der Kühlschrank in der Küche steht (enthalten ist). Somit können also zum einen Konzeptzugehörigkeiten von Objekten anhand gegebener Beziehungen zu anderen Objekten geschlossen werden, aber ebenso Relationszugehörigkeiten zwischen Objekten anhand bereits bekannter Konzeptzugehörigkeiten zwischen diesen.

Andere Anwendungsszenarien für Default-Regeln in Beschreibungslogiken sind auch denkbar. So werden in [14] Default-Regeln benutzt, um präzisere Antworten in einem Anfragesystem zu erhalten. Dabei werden die Anfragen durch Anwendung von Default-Regeln spezialisiert, sodaß die Antworten genauer ausfallen. Eine weitere Verwendungsmöglichkeit in diesem Anwendungskontext ist, unvollständige Datenbestände zu ergänzen um in diesen erweiterten Datenbeständen Dokumente zu finden, über die ansonsten wenig gesagt worden ist. Ist z.B. ein Benutzer an Fachbüchern interessiert, es existiere aber lediglich die Kategorie „Buch“ im Datenbestand, so könnten durch die Anwendung von Default-Regeln best. Bücher zu Fachbüchern erklärt werden (z.B. könnten die Attribute Autor, Verlag etc. ausschlaggebende Hinweise geben). Analog können diese Ideen auch für räumliche Informationssysteme eingesetzt werden, z.B. zur Anfragespezialisierung in einem Geographischen Informationssystem (GIS) oder zu dessen Datenergänzung.

Diese Arbeit ist wie folgt gegliedert. Im ersten Teil des Kapitels 2 wird näher auf terminologische Wissensrepräsentation, insbesondere auf die \mathcal{ALC} Familie eingegangen, und die unterschiedlichen Ausdrucksmächtigkeiten einiger Logiken dieser Familie werden anhand von einigen Beispielen demonstriert. Im zweiten Teil des Kapitels werden die grundlegenden Ideen einiger nicht-monotoner Logiken kurz erläutert, insbesondere die für diese Arbeit wesentliche Default-Logik von Reiter. Die bereits informell vorgestellte Idee der möglichen Extensionen einer Default-Theorie wird formalisiert.

In Kapitel 3 werden zwei Algorithmen zur Berechnung der möglichen Extensionen einer Default-Theorie nach [2] und [3] präsentiert.

Anschließend werden im Kapitel 4 die priorisierten Default-Regeln vorgestellt, und ein Algorithmus zur Berechnung der sog. *S-Extensionen* vorgestellt.

In Kapitel 5 werden dann die bereits angesprochenen Erweiterungen der terminologischen Default-Schließens diskutiert, die für das Default-Schließen mit $\mathcal{ALCRP}(\mathcal{S}_2)$ in unserem räumlichen Anwendungskontext notwendig wurden.

In Kapitel 6 werden die Algorithmen zur Berechnung der Extensionen gegenübergestellt, und auf einige Schwächen der behandelten Algorithmen, insbesondere solche, die durch die vorgenommene Erweiterung zwingendermaßen neu entstanden sind, eingegangen. Hier werden auch einige Optimierungsideen diskutiert, um einige dieser Probleme zu lindern.

Schließlich gibt es im letzten Kapitel eine Zusammenfassung dieser Arbeit.

Kapitel 2

Grundlagen

In diesem Kapitel werden zunächst die Grundlagen terminologischer Wissensrepräsentation eingeführt. Es werden die Beschreibungslogiken \mathcal{ALC} , $\mathcal{ALC}(\mathcal{D})$ und schließlich $\mathcal{ALCRP}(\mathcal{D})$ vorgestellt, wobei letztere das Schlußfolgern über räumliche Informationen erlaubt. Schließlich werden auch die grundlegenden Ideen einiger nicht-monotoner Logiken erwähnt und insbesondere die Reitersche Default-Logik eingeführt.

2.1 Terminologische Wissensrepräsentation

Wie schon in der Einleitung erwähnt, sind die Beschreibungslogiken eine Familie von Formalismen zur Repräsentation von konzeptuellem Wissen. Sie sind aus den früheren Ansätzen zur Wissensrepräsentation, wie den Semantischen Netzen und Frames entstanden, und besitzen im Gegensatz zu diesen eine formale Semantik, mit der das Auftreten von Mehrdeutigkeiten ausgeschlossen wird und Widersprüchlichkeiten entdeckt werden können. Die Semantik einer Beschreibungslogik kann mengentheoretisch spezifiziert werden, ähnlich der Tarskischen Semantik für die Prädikatenlogik. Terminologische Systeme implementieren Beschreibungslogiken und stellen den Benutzern Systemdienste zur Verfügung, die es erlauben, aus dem repräsentierten Wissen Schlüsse zu ziehen, d.h. es werden Fakten explizit gemacht, die vorher nur implizit vorhanden waren. Somit lassen sie sich als spezielle Theorembeweiser (für die jeweilige Logik) auffassen. Als erstes terminologisches System mit wohldefinierter Semantik gilt KL-ONE. Daraufhin entstanden viele andere KL-ONE-ähnliche Systeme, wie BACK, CLASSIC, KANDOR, KL-TWO, KRIS, K-REP, LOOM, MESON, NIKL, SB-ONE u.s.w.

Beschreibungslogiken können sich in ihrer Ausdrucksmächtigkeit stark unterscheiden. Die Ausdrucksmächtigkeit hängt wesentlich davon ab, welche *term-bildenden Operatoren* die Syntax der Sprache bereitstellt. Ist die Sprache zu ausdrucks mächtig, so kann die Entscheidbarkeit dabei verlorengehen. Selbst wenn die Sprache entscheidbar ist, müssen komplexitätstheoretische Resultate berücksichtigt werden. Daher ist man bemüht, gerade für praktische Anwendungen eine Kombination von Sprachkonstrukten zu wählen, sodaß die Inferenzprobleme in der resultierenden Logik entscheidbar und traktabel sind. Zudem sollten vollständige und korrekte Kalküle zur Lösung der Inferenzprobleme gefunden werden.

In Beschreibungslogiken wird eine Unterscheidung zwischen konzeptuellem Wissen und Objektwissen vorgenommen. Üblicherweise wird das konzeptuelle Wissen in der sog. TBox in Form von Begriffsdefinitionen ausgedrückt, welche somit den begrifflichen Rahmen der modellierten Domäne bereitstellen. Hier werden Begriffe durch Konzepte und Beziehungen (Relationen) durch Rollen beschrieben. Konzepte werden durch Konzeptterme beschrieben, die unter Verwendung von Konzeptnamen (atomaren Konzepttermen) und den Operatoren der jeweiligen Sprache gebildet werden. Analog hierzu bieten einige Beschreibungslogiken rollenbildende Terme (definierte oder komplexe Rollen). Ein terminologisches Konzept-Axiom (TBox-Axiom) hat üblicherweise die Form $CN \sqsubseteq C$ oder $CN \doteq C$, wobei CN ein (atomarer) Konzeptname und C ein Konzeptterm ist. Dabei ist es zugelassen, daß einige Konzeptnamen CN nicht auf der linken Seite eines terminologischen Axioms auftreten. Zusätzlich wird i.d.R. gefordert, daß ein Konzeptname nur einmal auf der linken Seite eines TBox-Axioms auftritt und die resultierende Terminologie zyklonfrei ist. Konzepte, die nicht auf der linken Seite eines terminologischen Axioms auftreten werden als *atomar* bezeichnet. Hingegen ist es bei Axiomen der Art $CN \sqsubseteq C$ und $CN \doteq C$ sinnvoll, vom *definierten* Konzept CN zu sprechen, wobei ersteres als unvollständig oder partiell definiertes, letzteres als vollständiges Konzept bezeichnet wird. Oftmals wird auch von *definierten* und *primitiven* Konzepten gesprochen. Für ein vollständig definiertes Konzept können sowohl die notwendigen als auch die hinreichenden Bedingungen angegeben werden, während bei einem partiell definierten Konzept nur die notwendigen Bedingungen bekannt sind.¹ Heutzutage läßt man in vielen Sprachen als TBox-Axiome auch Ausdrücke der Art $C \sqsubseteq D$ und $C \doteq D$ zu (wobei C und D beliebige Konzeptterme der Sprache sein können), sodaß die Unterscheidung zwischen atomaren, primitiven und definierten Konzepten nicht mehr sinnvoll ist, da es sich um beliebige Impli-

¹Eine Bedingung B heißt *notwendig* für A , falls $A \Rightarrow B$ ($A \sqsubseteq B$). Hingegen sagt man, A sei *hinreichend* für B .

kationen bzw. Biimplikationen zwischen Konzeptausdrücken handelt. Auch die Forderung der Zyklenfreiheit und einmaligen Verwendung von Konzeptnamen auf der linken Seite kann heutzutage in einigen Sprachen aufgegeben werden.

Beispiel: TBox mit primitiven und definierten Konzepten

$Mensch \sqsubseteq Lebewesen \sqcap intelligent$

$Mann \doteq Mensch \sqcap maennlich$

$Vater \doteq Mann \sqcap \exists hat_kind.Mensch$

In dieser TBox (auch Terminologie), welche einfach eine Menge von TBox-Axiomen ist, haben wir ein unvollständig definiertes bzw. primitives Konzept Mensch, das als Teilmenge der Klasse der intelligenten Lebewesen beschrieben wird. In diesem Fall (\sqsubseteq) haben wir nur die notwendigen Informationen (was es für ein Objekt bedeutet, ein Mensch zu sein), wobei wir ein TBox-Axiom der Form $CN \sqsubseteq C$ verwendet haben. Durch das dritte Axiom wird das Konzept Vater vollständig definiert, wobei wir ein TBox-Axiom der Form $CN \doteq C$ verwendet haben. Demnach ist ein Mann ein Vater, wenn er mindestens ein menschliches Kind hat. Hier sagt das Gleichheitszeichen (\doteq), daß wir sowohl die notwendigen als auch die hinreichenden Informationen über Väter haben. Folglich können wir über ein Objekt, das ein Mann ist und mindestens ein menschliches Kind hat, auch sagen, daß es sich um einen Vater handelt. Aus diesem Beispiel kann man leicht entnehmen, daß man sich so immer komplexere Definitionen ausdenken kann. Sofern die zugrundeliegende Beschreibungslogik terminologische Axiome der allgemeinsten Form ($C \sqsubseteq D, C \doteq D$) zuläßt, können Konzepte frei von jeglichen Restriktionen (wie etwa Zyklenfreiheit etc.) modelliert werden.

Die Konzepte in der TBox werden in einer sog. *Taxonomie* angeordnet, um das Wissen strukturiert in einem Vererbungsverband darzustellen. Zudem können best. Inferenzalgorithmen die verbandartige Struktur der Taxonomie (es handelt sich um einen gerichteten azyklischen Graphen) effizient ausnutzen, um best. Inferenzen zu beschleunigen.

Das Objektwissen wird in der ABox geführt. Hier finden sich sog. Individuen, für die best. Beschreibungen assertiert werden können. Eine ABox ist eine Menge assertorischer, sog. ABox-Axiome. Man unterscheidet im wesentlichen zwei Arten von ABox-Axiomen: Axiome der Form $i : C$, wobei C ein Konzeptterm ist, der Konzeptnamen aus der TBox enthalten kann, und Axiome der Form $(i, j) : R$, wobei R eine (in der TBox primitive oder definierte) Rolle sein kann. Informell bedeutet ein Axiom der Form $i : C$,

daß von dem Individuum i bekannt ist, daß es Instanz des Konzeptes C ist. Hingegen wird durch ein Axiom der Form $(i, j) : R$ ausgesagt, daß i und j in der Relation R stehen. Man sagt dann auch, daß j ein Füller der Rolle R von i ist. ABox-Axiome der Art $i : C$ werden als Konzeptzugehörigkeits-Axiome (Concept Membership Assertions/Axioms) bezeichnet, im Gegensatz zu den Rollenzugehörigkeitsaxiomen (Role Membership Assertions/Axioms). In ausdrucksstarken Beschreibungslogiken findet man noch andere Arten von ABox-Axiomen (s.u.). Eine Menge von ABox-Axiomen kann daher als Objektwissen oder Weltbeschreibung der erwähnten Individuen aufgefaßt werden. Während der terminologische Teil des Systems die Begriffe repräsentiert, durch die die Welt gesehen und dargestellt werden soll, kann man mit Hilfe des assertionalen Formalismus konkrete Tatsachen über die Welt ausdrücken.

Beispiel: ABox

$\{peter : Mensch, hans : Mensch \sqcap maennlich, (hans, peter) : hat_kind\}$

Offensichtlich folgt aus dieser ABox in Verbindung mit obiger TBox, daß es sich bei dem Individuum $hans$ um eine Instanz des Konzeptes *Vater* handelt. Dieser Schluß würde durch die sog. *ABox-Realisation* (Inferenzdienst) geführt werden (s.u.).

Ein Paar aus TBox und ABox wird als Wissensbasis bezeichnet.

Definition 2.1 (Wissensbasis) *Eine Wissensbasis $\Sigma = (T, A)$ ist ein Paar bestehend aus einer Menge von terminologischen Axiomen (TBox-Axiomen) T , genannt TBox, und einer Menge von assertorischen Behauptungen (ABox-Axiomen) A , genannt ABox.*

Die beschreibungslogischen Systeme stellen den Benutzern eine Reihe von Inferenzdiensten zur Verfügung, die im weiteren dargestellt werden.

2.1.1 Inferenzdienste

Zu den Basisinferenzen (Diensten) gehört der Erfüllbarkeits- bzw. Konsistenztest für Konzeptterme. Ein Konzeptterm C ist *erfüllbar* oder *konsistent* in Bezug auf eine TBox T , $C \not\sim \perp$, g.d.w. es ein Modell \mathcal{I} von T gibt, so daß $C^{\mathcal{I}} \neq \emptyset$. Wie üblich ist ein Modell eine Interpretation, die die entsprechenden Axiome wahr macht bzw. erfüllt. Die Interpretation wird für atomare Konzepte und Rollen festgelegt und ergibt sich dann anhand der Termstruktur

für zusammengesetzte Terme, s.u. Ein nicht-erfüllbarer Konzeptterm heißt auch *inkonsistent* oder *inkohärent* (die Extension des Konzeptes ist bzgl. der TBox T leer). Mit Hilfe des Konsistenztests kann man die Folgerbarkeit feststellen, denn $\alpha \models \beta$ gilt genau dann, wenn $\alpha \cup \neg\beta$ inkonsistent ist.

Es gibt auch einen ABox-Konsistenztest. Eine ABox A ist konsistent in Bezug auf eine TBox T g.d.w. es ein Modell \mathcal{I} für A in Bezug auf T gibt.

Eine der wichtigsten Inferenzen innerhalb einer terminologischen Logik ist die *Subsumptionsbestimmung* zwischen Konzepten. Die Subsumptionsbeziehung ermöglicht es uns, Konzepte in einer Vererbungshierarchie in einer Ober-Unterkonzeptbeziehung anzuordnen. Die Bestimmung der richtigen Position eines neuen Konzeptes in einer bereits bestehenden Taxonomie wird als *Klassifikation* des neuen Konzeptes bezeichnet. Die formale Definition der Subsumptionsbeziehung ist wie folgt gegeben:

D subsumiert C , $C \preceq D$ genau dann, wenn $\|C\|^{\mathcal{I}} \subseteq \|D\|^{\mathcal{I}}$ für alle Modelle \mathcal{I} von T gilt.

Beispiel: Subsumptionsbeziehung

$Vater \doteq Mann \sqcap \exists hat_kind.Mensch$
 $Großvater \doteq Mann \sqcap \exists hat_kind.Vater$

Hier wird das definierte Konzept *Großvater* durch das definierte Konzept *Vater* subsumiert, da für alle Domänenobjekte, für die das Konzept *Großvater* gilt, auch *Vater* in allen Modellen der Terminologie gilt, obwohl dies nicht explizit formuliert (angegeben) wurde. Ein vollständiger und korrekter Subsumptionsalgorithmus müßte diese Subsumptionsbeziehung ermitteln.

Ein weiterer Inferenz ist die *Äquivalenzbestimmung* zweier Konzepte in einer Terminologie. Zwei Konzepte C und D sind äquivalent, $C \sim D$, genau dann, wenn $\|C\|^{\mathcal{I}} = \|D\|^{\mathcal{I}}$ für alle Modelle \mathcal{I} von T gilt. In anderen Worten heißt dies, daß die Extensionen der beiden Konzepte gleich sind.

Konsistenz, Subsumption und Äquivalenz von Konzepten in einer TBox lassen sich wechselseitig aufeinander reduzieren, sodaß z.B. ein Erfüllbarkeitstester für Konzepte ausreichend ist (dieser entscheidet, ob in einer gegebenen TBox das Konzept C erfüllbar ist, also ob $C \not\perp$ gilt):

- $C \sim D$ genau dann, wenn $C \preceq D$ und $D \preceq C$ gilt.
- $C \sim \perp$ genau dann, wenn $C \preceq \perp$ gilt.

- $C \preceq D$ genau dann, wenn $C \sqcap \neg D \sim \perp$ gilt ²

Die *Instanzüberprüfung* (*Instance Checking*) nimmt ein Individuum i und ein Konzept C und überprüft, ob in allen Modellen der Wissensbasis Σ $i : C$ gilt. Das Problem Instanzüberprüfung kann auf ABox-Konsistenz reduziert werden, denn ein Individuum i ist eine Instanz eines Konzeptes C in einer ABox A bezüglich einer TBox T , g.d.w. die ABox $A \cup i : \neg C$ bezüglich T inkonsistent ist. Eine weiterer Dienst, der sich auf eine Reihe von Instanzüberprüfungsproblemen reduzieren läßt, ist die *ABox-Realisation*, die als Äquivalent zur Konzept-Klassifikation bezeichnet werden kann. Hier wird für jedes Individuum einer gegebenen ABox A die Menge der spezifischsten Konzepte bzgl. einer TBox T ermittelt, von denen das Individuum eine Instanz ist. Auch Konzeptkonsistenz (und somit Subsumption etc., s.o.) kann auf ABox-Konsistenz reduziert werden, denn C ist bzgl. einer TBox T genau dann erfüllbar, wenn die ABox $\{i : \neg C\}$ für ein neues Individuum i bzgl. der TBox T inkonsistent ist.

2.1.2 Die Sprache \mathcal{ALC}

In diesem Abschnitt wird die Sprache \mathcal{ALC} , welche von Schmidt-Schauß & Smolka [11] eingeführt wurde, formal vorgestellt. Diese Sprache kann als minimale propositional vollständige Beschreibungslogik angesehen werden und stellt daher die Basis für die Definition von zahlreichen mächtigeren Logiken dar. Die oben erläuterten Grundlagen terminologischer Logiken haben natürlich auch hier ihre Gültigkeit.

Ein kurzes Beispiel soll uns einen ersten Eindruck verschaffen. Das primitive Konzept „Auto“ könnte folgendermaßen definiert werden:

$$Auto \sqsubseteq Fahrzeug \sqcap \exists hat_motor.Motor$$

Dies bedeutet, daß ein *Auto* ein *Fahrzeug* (hier als atomares Konzept modelliert) mit einem Motor ist, mit dem es über die Rolle *hat_motor* verbunden ist.

Man könnte nun ein definiertes Konzept „Dieselauto“ folgendermaßen definieren:

$$Dieselauto \doteq Auto \sqcap \forall hat_motor.Dieselmotor$$

Demnach ist ein Auto genau dann ein Dieselauto, wenn es ein Auto ist,

²vorausgesetzt, die Sprache bietet die volle Negation

und wenn alle Dinge, die über die Rolle *hat_motor* mit ihm verbunden sind, Dieselmotoren sind.

Syntax von \mathcal{ALC}

Die atomaren syntaktischen Sprachelemente von \mathcal{ALC} sind Konzeptnamen und Rollennamen. Diese zwei disjunkten Mengen von Namen sind die Basisbausteine der Sprache. Mit Hilfe der in \mathcal{ALC} zugelassenen unten aufgelisteten Operatoren lassen sich komplexere Konzeptterme nach folgender induktiver Definition bilden:

1. Jeder Konzeptname CN aus der Menge der Konzeptnamen ist ein \mathcal{ALC} -Konzeptterm.
2. \top und \perp sind \mathcal{ALC} -Konzeptterm.
3. Wenn C und D \mathcal{ALC} -Konzepte sind, und R ein Rollenname, so sind auch die folgenden Ausdrücke \mathcal{ALC} -Konzeptterme:

$C \sqcap D$ Konjunktion
 $C \sqcup D$ Disjunktion
 $\neg C$ Negation
 $\exists R.C$ Existenz-Einschränkung
 $\forall R.C$ Werte-Restriktion

4. Keine andere Ausdrücke sind \mathcal{ALC} -Konzeptterme.

Konzepte entsprechen einstelligem und Rollen zweistelligen Prädikaten der Prädikatenlogik. Der Konjunktions-Operator (\sqcap) ist ein Operator, welcher ein Konzept bildet, das aus der Konjunktion von zwei oder mehreren Konzepten (wegen der Assoziativität) besteht. Ein Objekt ist Instanz des Konzeptes $C \sqcap D$ genau dann, wenn es sowohl Instanz von C als auch von D ist. Der Disjunktions-Operator (\sqcup) ist ein Operator, welcher ein Konzept bildet, das aus der Disjunktion von zwei oder mehreren Konzepten (s.o.) besteht. Ein Objekt ist Instanz des Konzeptes $C \sqcup D$ genau dann, wenn es Instanz von C oder Instanz von D ist. Der Negations-Operator (\neg) ist ein Operator, welcher ein Konzept definiert, das das Komplement von C ist. Ein Objekt ist Instanz des Konzeptes $\neg C$ genau dann, wenn es nicht Instanz von C ist. Der Werte-Restriktions-Operator (\forall) beschränkt die R -Füller auf ein bestimmtes Konzept C . Die R -Füller bzgl. eines Individuums x sind alle Objekte y , die

mit x in der Relation R stehen, $R(x, y)$. Ein Objekt ist Instanz des Konzeptes $\forall R.C$, wenn alle seine R -Füller Instanzen des Konzeptes C sind. Der Existenz-Einschränkungs-Operator (\exists) fordert die Existenz eines R -Füllers, der Instanz des Konzeptes C sein muß. Ein Objekt ist Instanz des Konzeptes $\exists R.C$, wenn dieses Objekt zumindest einen R -Füller hat, der Instanz des Konzeptes C ist. Außerdem gibt es zwei vordefinierte Konzepte, nämlich \top (top) für Alles und \perp (bottom) für Nichts. Im folgenden wird die Semantik der \mathcal{ALC} -Konzeptterme definiert.

Semantik von \mathcal{ALC}

Die formale Semantik von \mathcal{ALC} kann mengentheoretisch definiert werden. Dabei handelt es sich um eine extensionalen Semantik, die durch eine Interpretation gegeben wird. Eine Interpretation \mathcal{I} ist ein Paar $(\Delta, \|\cdot\|^{\mathcal{I}})$. Hierbei ist Δ die Grundmenge oder Domäne, eine willkürliche nichtleere Menge von Objekten, und $\|\cdot\|^{\mathcal{I}}$ ist eine Interpretationsfunktion, die Konzeptterme auf Teilmengen von Δ und Rollen auf Teilmengen von $\Delta \times \Delta$ abbildet. Anders gesagt ist eine Interpretations- oder Extensions-Funktion durch eine Abbildung der Menge der syntaktischen Ausdrücke (s. obige Definition) in eine semantische Domäne definiert. Die Semantik ergibt sich kompositional anhand der Termstruktur der syntaktischen Ausdrücke, sobald die Interpretationen der atomaren Konzept- und Rollennamen festgelegt worden sind. Zudem haben die beiden vordefinierten Konzepte \top und \perp die fixierten Interpretationen Δ und \emptyset . Die Semantik eines \mathcal{ALC} -Konzepttermes ist gegeben durch:

$$\|\top\|^{\mathcal{I}} = \Delta$$

$$\|\perp\|^{\mathcal{I}} = \emptyset$$

$$\|C \sqcap D\|^{\mathcal{I}} = \|C\|^{\mathcal{I}} \cap \|D\|^{\mathcal{I}}$$

$$\|C \sqcup D\|^{\mathcal{I}} = \|C\|^{\mathcal{I}} \cup \|D\|^{\mathcal{I}}$$

$$\|\neg A\|^{\mathcal{I}} = \Delta \setminus \|A\|^{\mathcal{I}}$$

$$\|\forall R.C\|^{\mathcal{I}} = \{a \in \Delta \mid \forall b \in \Delta : (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$$

$$\|\exists R.C\|^{\mathcal{I}} = \{a \in \Delta \mid \exists b \in \Delta : (a, b) \in R^{\mathcal{I}}, b \in C^{\mathcal{I}}\}$$

Terminologische Axiome von \mathcal{ALC}

Terminologische Axiome wurden bereits oben erläutert. Eine Menge solcher Axiome bildet die erwähnte TBox. Informell ist eine Interpretation \mathcal{I} genau dann ein Modell für ein TBox-Axiom der Form $\text{Konzeptname} \doteq \text{Konzeptterm}$, wenn $\text{Konzeptname}^{\mathcal{I}} = \text{Konzeptterm}^{\mathcal{I}}$ gilt. Analog ist eine Interpretation \mathcal{I} genau dann ein Modell für ein TBox-Axiom der Form $\text{Konzeptname} \sqsubseteq \text{Konzeptterm}$, wenn $\text{Konzeptname}^{\mathcal{I}} \subseteq \text{Konzeptterm}^{\mathcal{I}}$ gilt. Eine Interpretation \mathcal{I} ist genau dann ein Modell für eine TBox T , wenn sie Modell für jedes TBox-Axiom ist.

Während in früheren Arbeiten zum Thema \mathcal{ALC} stets nur TBox-Axiome dieser Art zugelassen wurden, so weiß man inzwischen, wie TBox-Axiome der allgemeinsten Art $\text{Konzeptterm1} \sqsubseteq (\doteq) \text{Konzeptterm2}$ zu behandeln sind.

Assertorische Axiome von \mathcal{ALC}

Wie bereits erläutert, gibt es in \mathcal{ALC} zwei Arten von ABox-Axiomen. Entweder wird assertiert, daß ein Individuum Instanz eines bestimmten Konzeptes ist, oder zwei Individuen werden in Relation gesetzt. Eine endliche Menge assertorischer Axiome bildet die bereits erwähnte ABox. Kurz gesagt werden hier Individuen zu Konzepten, und Individuen-Paare zu Rollen zugewiesen. Syntaktisch sieht diese Zuweisung wie folgt aus:

$$\begin{aligned} i &: C \\ (i, j) &: R \end{aligned}$$

Eine Interpretation \mathcal{I} erfüllt ein assertorisches Axiom $i : C$ genau dann, wenn $i^{\mathcal{I}} \in C^{\mathcal{I}}$ gilt, und $(i, j) : R$ genau dann, wenn $(i^{\mathcal{I}}, j^{\mathcal{I}}) \in R^{\mathcal{I}}$ ist. In \mathcal{ALC} gilt die *Unique name assumption*. Dies bedeutet, daß $\forall i, j \in ABox : i \neq j \rightarrow i^{\mathcal{I}} \neq j^{\mathcal{I}}$.

2.1.3 Die Sprache $\mathcal{ALC}(\mathcal{D})$

Die Sprache $\mathcal{ALC}(\mathcal{D})$ wurde von Baader und Hanschke [1] entwickelt. $\mathcal{ALC}(\mathcal{D})$ erweitert \mathcal{ALC} um einen zusätzlichen konzeptbildenden Operator, der auf sog. *konkreten Domänen* (s.u.) basiert. Dieser Operator stellt die Verbindung zwischen dem beschreibungslogischen Formalismus und einer konkreten Domäne her. Da $\mathcal{ALC}(\mathcal{D})$ eine echte Teilmenge von $\mathcal{ALCRP}(\mathcal{D})$ ist, wird

auf die formale Definition von $\mathcal{ALC}(\mathcal{D})$ hier verzichtet. Zunächst wird der Begriff der konkreten Domäne näher erläutert und anhand eines Beispiels verdeutlicht.

Konkrete Domänen

Ein Nachteil der bisher dargestellten terminologischen Logiken ist, daß alles Wissen auf einer abstrakten logischen Ebene dargestellt werden muß. Um Beschreibungslogiken z.B. in ingenieurwissenschaftlichen Kontexten sinnvoll anwenden zu können, wird es oftmals notwendig sein, z.B. mit Zahlen und arithmetischen Ausdrücken über diesen umgehen zu können. Für das Konzept „Zahnrad mit kleinem Durchmesser“ möchte man gerne Aussagen bzw. Einschränkungen bzgl. des erlaubten Durchmessers des Rades machen können, etwa daß der Durchmesser zwischen 3 und 4 cm liegen soll. Dies kann auf einer abstrakten logischen Ebene nicht befriedigend dargestellt werden. Die sogenannten konkreten Domänen erweitern daher den abstrakten logischen Teil konventioneller Beschreibungslogiken um z.B. reelle Zahlen und eine Reihe von (evtl. mehrstelligen) *Prädikaten* über diesen. Für obiges Bsp. könnte man das Prädikat $klein(x) := 3 \leq x \leq 4$ definieren und dann zur Definition des Konzeptes mit einbeziehen. Objekte wie reelle Zahlen werden in gewisser Weise als „konkretere Objekte“ der Welt angesehen. Weitere konkrete Domänen können z.B. Polygone oder Punktemengen sein, für die eine Reihe von Prädikaten definiert sind. Diese konkreten Objekte können nun über sog. *Attribute* (partielle Funktionen, welche als spez. Rollen angesehen werden können, engl. *Features*) aber auch über *Attributketten* (also komponierte Attribute, engl. *Feature Chains*) aus der abstrakten Domäne erreicht werden.³ Eine Instanz (in der abstrakten Domäne) des Konzeptes „Zahnrad mit kleinem Durchmesser“ hat z.B. ein Attribut „Durchmesser“, welches eine reelle Zahl als konkretes Objekt (in der konkreten Domäne) als Füller hat, für welches das Prädikate *klein* gilt.

Eine *zulässige konkrete Domäne* ist formal wie folgt definiert:

Definition 2.2 (Konkrete Domäne, Zulässigkeit) *Eine konkrete Domäne D ist ein Paar (Δ_D, Φ_D) , wobei Δ_D eine Menge (die Domäne) und Φ_D eine Menge von Prädikatsnamen ist. Jeder Prädikatsname P aus Φ_D ist mit einer Stelligkeit n und einem n -ären Prädikat $P^D \subseteq \Delta_D^n$ verknüpft. Seien P_1, \dots, P_k Prädikatsnamen aus Φ_D mit den Stelligkeiten n_1, \dots, n_k . Sei $x^{(i)}$*

³Ein einfaches Attribut gilt als Attributkette der Länge 1, und andersrum.

ein n_i -Tupel $(x_1^{(i)}, \dots, x_{n_i}^{(i)})$ von Variablen. Eine konkrete Domäne D heißt zulässig g.d.w.

1. die Menge der Prädikatsnamen unter Negation abgeschlossen ist sowie einen Namen für Δ_D enthält und
2. das Erfüllbarkeitsproblem für endliche Konjunktionen von Prädikaten $\bigwedge_{i=1}^k P_i(x^{(i)})$ entscheidbar ist.

Für einen beliebigen Prädikatsnamen $P \in \Phi_D$, \bar{P} steht für die Negation von P , für welche wir $\bar{P}^D := (\Delta_D)^n \setminus P^D$ haben, wobei n die Stelligkeit von P ist.

Die Vereinigung zweier zulässiger konkreter Domänen ergibt wieder eine zulässige konkrete Domäne (Beweis s. [7]). Zum Beispiel gibt es Anwendungen, in welchen sowohl eine konkrete räumliche als auch eine konkrete zeitliche Domäne gebraucht wird. Die Vereinigung zweier konkreter Domänen wird im folgenden nicht weiter betrachtet.

$\mathcal{ALC}(\mathcal{D})$ enthält \mathcal{ALC} als echte Teilmenge. Die Schnittstelle zwischen abstrakter (beschreibungslogischer) und konkreter Domäne ist durch den folgenden konzeptbildenden Operator gegeben:

$$\exists u_1, \dots, u_n.P,$$

wobei u_i eine Featurekette und P ein Prädikat aus der konkreten Domäne ($P \in \Phi_D$) der Stelligkeit n ist.

Im Unterschied zu \mathcal{ALC} können nun auch Attribute in den \exists - und \forall -Quantoren verwendet werden. Daher sind

$\forall f.C, \exists f.C$ (Werte- und Existenzrestriktion für Attribute)

ebenfalls $\mathcal{ALC}(\mathcal{D})$ -Konzeptterme, wenn f ein Attributname ist.

Informell besteht die Extension dieses Ausdrucks aus allen abstrakten Objekten $a \in \Delta_I$, für die es entspr. konkrete Objekte $x_1, \dots, x_n \in \Delta_D$ erreichbar über die Attributketten u_1, \dots, u_n gibt, sodaß $P(x_1, \dots, x_n)$ gilt, $(x_1, \dots, x_n) \in P^D$:

$$\| \exists u_1, \dots, u_n.P \|^\mathcal{I} = \{ a \in \Delta_I \mid \exists x_1, \dots, x_n \in \Delta_D : (a, x_1) \in u_1^\mathcal{I}, \dots, (a, x_n) \in u_n^\mathcal{I}, (x_1, \dots, x_n) \in P^D \}$$

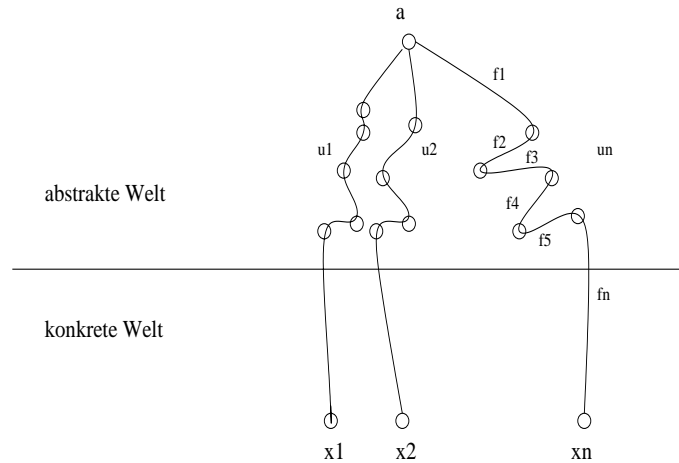


Abbildung 2.1: $\|\exists u_1, \dots, u_n.P\|^{\mathcal{I}}$

In Abbildung 2.1 sind Objekte aus der konkreten Domäne (x_1, \dots, x_n) durch Featureketten u_1, \dots, u_n (für $u_n = f_1 \circ \dots \circ f_n$) von dem abstrakten Objekt a aus der abstrakten Domäne zu erreichen. Zwischen diesen Objekten (x_1, \dots, x_n) gilt das n -äre Prädikat P .

Die Interpretation der anderen Operatoren ist mit den entsp. \mathcal{ALC} -Interpretationen identisch. Natürlich müssen Attribute nun aber als partielle Funktionen interpretiert werden.

Zudem gibt es aufgrund des neuen Operators ein zusätzliches ABox-Axiom (s.u.).

Beispiel für $\mathcal{ALC}(\mathcal{D})$

Es soll an dieser Stelle an einem Beispiel verdeutlicht werden, was $\mathcal{ALC}(\mathcal{D})$ von \mathcal{ALC} unterscheidet. In $\mathcal{ALC}(\mathcal{D})$ können wir nun das Konzept „schnelles Auto“ als die Menge aller Autos, die schneller als 200 km/h sind, definieren. Die Definition dieses Konzeptes war in \mathcal{ALC} nicht möglich.

$$\text{Schnelles_auto} \doteq \text{Auto} \sqcap \exists \text{geschwindigkeit. } >_{200\text{km/h}}$$

Hier werden abstrakte Objekte von Typ Auto über das Attribut *geschwindigkeit* mit einem konkreten Objekt (einer Zahl) verbunden. Das einstellige Prädikat $>_{200\text{km/h}}$ stellt sicher, daß das konkrete Objekt das gewünschte Kriterium erfüllt.

2.1.4 Die Sprache $\mathcal{ALCRP}(\mathcal{D})$

Die Sprache $\mathcal{ALCRP}(\mathcal{D})$, eine Entwicklung von V. Haarslev, C. Lutz und R. Möller, ist eine Erweiterung der Sprache $\mathcal{ALC}(\mathcal{D})$ um einen zusätzlichen *rol-lenbildenden Operator*. Dieser Operator basiert auf konkreten Domänen und ist das rollenbildende Pendant zum konzeptbildenden Prädikats-Operator aus $\mathcal{ALC}(\mathcal{D})$. Ebenso wie $\mathcal{ALC}(\mathcal{D})$ kann auch $\mathcal{ALCRP}(\mathcal{D})$ mit einer zulässigen konkreten Domäne parametrisiert werden.

Die Idee bei $\mathcal{ALCRP}(\mathcal{D})$ ist, konkrete Beziehungen zwischen zwei Objekten der konkreten Domäne durch Rollen in der abstrakten Domäne darstellen zu können. Die Objekte der abstrakten Domäne werden durch Rollen in Beziehung gesetzt, wenn die über Featureketten erreichbaren konkreten Objekte ein beliebiges Prädikat P der konkreten Domäne erfüllen (wobei P die richtige Stelligkeit haben muß, also mindestens zweistellig ist). Hier werden also Rollen in der abstrakten Domäne anhand von Prädikaten in der konkreten Domäne definiert. Die Eigenschaften dieser definierten oder komplexen Rollen ergeben sich anhand der Eigenschaften der korrespondierenden Prädikate der konkreten Domäne (z.B. Transitivität).

In [6] wird gezeigt, daß das Erfüllbarkeitsproblem in $\mathcal{ALCRP}(\mathcal{D})$ unentscheidbar ist, und daß daher gewisse Restriktionen vorgenommen werden müssen, um das Terminieren des Verfahrens zu sichern. Daher wird im Weiteren die eingeschränkte Version von $\mathcal{ALCRP}(\mathcal{D})$ betrachtet. Der Terminierungsbeweis für die eingeschränkte Version von $\mathcal{ALCRP}(\mathcal{D})$ ist ebenfalls in [6] zu finden.

Bevor wir $\mathcal{ALCRP}(\mathcal{D})$ formal kennenlernen, betrachten wir zunächst ein Beispiel aus der Auto-Domäne.

Zunächst definieren wir ein Prädikat *gefahrlische_situation*:

$$\textit{gefahrlische_situation}(a, b, c, d) := (|a - c| < 10 \wedge |b - d| > 50).$$

Unter Verwendung des Prädikates definieren wir die komplexe Rolle *zu_nah*:

$$\textit{zu_nah} \doteq \exists(\textit{abstand}, \textit{geschw})(\textit{abstand}, \textit{geschw}).\textit{gefahrlische_situation}$$

Nun kann unser Beispiel-Konzept wie folgt aussehen:

$$\textit{gefahrlisches_Auto} \doteq \exists \textit{zu_nah}. \textit{Auto} \sqcap \textit{Auto}$$

Ein Auto ist erst dann ein *gefahrlisches_Auto*, wenn die Objekte, die über die Attribute *abstand* und *geschw* (für Geschwindigkeit) zu erreichen sind das Prädikat *gefahrlische_situation* erfüllen. Diese Definition wäre in $\mathcal{ALC}(\mathcal{D})$

nicht möglich gewesen, da in der Definition des Konzeptes *gefaehrliches_Auto* über die definierte (komplexe) Rolle *zu_nah* existentiell quantifiziert wird.

Im folgenden werden die $\mathcal{ALCRP}(\mathcal{D})$ Rollen- und Konzeptterme sowie assertorischen Axiome vorgestellt.

Syntax von $\mathcal{ALCRP}(\mathcal{D})$

$\mathcal{ALC}(\mathcal{D})$ ist eine echte Teilmenge von $\mathcal{ALCRP}(\mathcal{D})$. Bezüglich der konzeptbildenden Operatoren gibt es keinen Unterschied zu $\mathcal{ALC}(\mathcal{D})$.

Seien R und F zwei disjunkte Mengen von Rollen- bzw. Attributnamen. Die Elemente von F werden Attribute genannt. Eine Attributkette ist eine Verkettung von Attributen $f_1 \circ \dots \circ f_n$, wobei $n = 1$ möglich ist. Alle Elemente von R und F sind Rollenterme. Sei $P \in \Phi_{\mathcal{D}}$ ein Prädikatsname der Stelligkeit $n + m$ und seien u_1, \dots, u_n sowie v_1, \dots, v_m Attributketten, dann ist der Ausdruck $\exists(u_1, \dots, u_n)(v_1, \dots, v_m).P$ mit $n \geq 1$ und $m \geq 1$ (*rollenbildender Prädikats-Operator*) ebenfalls ein Rollenterm. Ausdrücke dieser Form heißen auch *komplexe* Rollenterme. Die Elemente von R und F werden im Gegensatz dazu *atomare* Rollenterme genannt. Sei S ein Rollename und T ein Rollenterm. Dann ist $S \doteq T$ ein terminologisches Axiom.

Im folgendem wird für $\mathcal{ALCRP}(\mathcal{D})$ eine mengentheoretische Semantik angegeben.

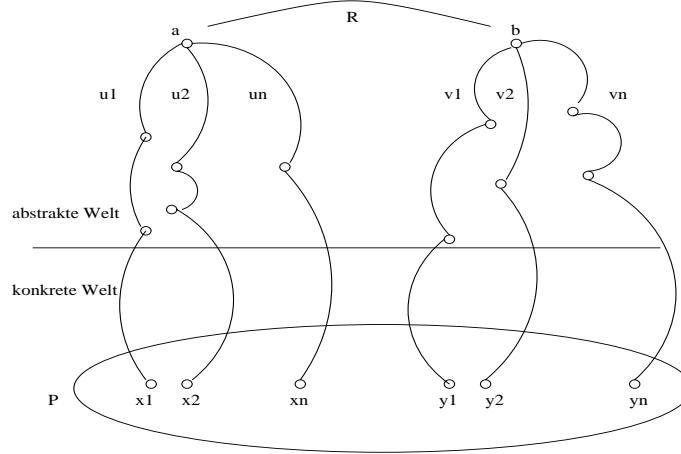
Semantik von $\mathcal{ALCRP}(\mathcal{D})$

Eine Interpretation $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ besteht aus einer Menge $\Delta_{\mathcal{I}}$ (der abstrakten Domäne) und einer Interpretationsfunktion $\cdot^{\mathcal{I}}$. Die Mengen $\Delta_{\mathcal{D}}$ (konkrete Domäne) und $\Delta_{\mathcal{I}}$ sind disjunkt. Die Interpretationsfunktion bildet jeden Konzeptterm C auf eine Teilmenge $C^{\mathcal{I}}$ von $\Delta_{\mathcal{I}}$, jeden Rollenterm R auf eine Teilmenge $R^{\mathcal{I}}$ von $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$ und jeden Attributnamen f auf eine partielle Funktion $f^{\mathcal{I}}$ von $\Delta_{\mathcal{I}}$ nach $\Delta_{\mathcal{D}} \cup \Delta_{\mathcal{I}}$ ab. $f^{\mathcal{I}}(a) = x$ wird als $(a, x) \in f^{\mathcal{I}}$ geschrieben. Für eine Attributkette $u = f_1 \circ \dots \circ f_n$ ist $u^{\mathcal{I}}$ die Komposition $f_1^{\mathcal{I}} \circ \dots \circ f_n^{\mathcal{I}}$ der partiellen Funktionen $f_1^{\mathcal{I}}, \dots, f_n^{\mathcal{I}}$.

Die Denotation einer Konzeptbeschreibung ist gegeben durch :

$$\| \top \|^{\mathcal{I}} = \Delta_{\mathcal{I}}$$

$$\| \perp \|^{\mathcal{I}} = \emptyset$$


 Abbildung 2.2: $\|R\|^{\mathcal{I}} = \|\exists(u_1, \dots, u_n)(v_1, \dots, v_m).P\|^{\mathcal{I}}$

$$\|C \sqcap D\|^{\mathcal{I}} = \|C\|^{\mathcal{I}} \cap \|D\|^{\mathcal{I}}$$

$$\|C \sqcup D\|^{\mathcal{I}} = \|C\|^{\mathcal{I}} \cup \|D\|^{\mathcal{I}}$$

$$\|\neg A\|^{\mathcal{I}} = \Delta_I \setminus \|A\|^{\mathcal{I}}$$

$$\|\forall R.C\|^{\mathcal{I}} = \{a \in \Delta_I \mid \forall b : (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$$

$$\|\exists R.C\|^{\mathcal{I}} = \{a \in \Delta_I \mid \exists b \in \Delta_I : (a, b) \in R^{\mathcal{I}}, b \in C^{\mathcal{I}}\}$$

$$\|\exists u_1, \dots, u_n.P\|^{\mathcal{I}} = \{a \in \Delta_I \mid \exists x_1, \dots, x_n \in \Delta_D : \\ (a, x_1) \in u_1^{\mathcal{I}}, \dots, (a, x_n) \in u_n^{\mathcal{I}}, (x_1, \dots, x_n) \in P^D\}$$

$$\|\exists(u_1, \dots, u_n)(v_1, \dots, v_m).P\|^{\mathcal{I}} = \{(a, b) \in \Delta_I \times \Delta_I \mid \\ \exists x_1, \dots, x_n, y_1, \dots, y_m \in \Delta_D : \\ (a, x_1) \in u_1^{\mathcal{I}}, \dots, (a, x_n) \in u_n^{\mathcal{I}}, \\ (b, y_1) \in v_1^{\mathcal{I}}, \dots, (b, y_m) \in v_m^{\mathcal{I}}, \\ (x_1, \dots, x_n, y_1, \dots, y_m) \in P^D\}$$

Eine Interpretation \mathcal{I} heißt *Modell* einer TBox T g.d.w. für alle terminologischen Axiome $A \doteq D$ in T $A^{\mathcal{I}} = D^{\mathcal{I}}$ und für alle terminologischen Axiome $A \sqsubseteq D$ in T $A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ gilt.

Die Abbildung 2.2 zeigt, daß zwei Objekte aus der abstrakten Domäne (a und b) über einer Rolle R in Beziehung gesetzt werden können, wenn die über die Attributketten u_1, \dots, u_n und v_1, \dots, v_m erreichbaren konkreten Objekte der konkreten Domäne $(x_1, \dots, x_n, y_1, \dots, y_m)$ das $n+m$ -äre Prädikat P erfüllen, d.h. wenn diese Eigenschaft vorhanden ist, kann zwischen a und b die Rolle R etabliert werden.

Wie oben schon erwähnt, ist $\mathcal{ALCRP}(\mathcal{D})$ bezüglich des Erfüllbarkeitsproblems unentscheidbar. Es gibt aber mindestens zwei Möglichkeiten, um diese Unzulänglichkeit zu umgehen. Die erste Möglichkeit wäre, weitere Forderungen an die Struktur der erlaubten konkreten Domänen z.B. bzgl. der Eigenschaften der verwendeten Prädikate zu stellen. Unglücklicherweise sind viele interessante konkrete Domänen bereits in ihrer Struktur vorgegeben (z.B. die später noch zu behandelnde konkrete Domäne S_2 , die die RCC-8-Prädikate und somit auch deren Eigenschaften verwendet). Eine weitere Möglichkeit ist, die Kombinationsmöglichkeiten einiger kritischer Operatoren einzuschränken. Dieser Möglichkeit wird im folgenden dargestellt, wozu vorerst einige technische Definitionen (aus [8]) gemacht werden müssen:

Ein Konzeptterm C heißt *expandiert* (engl. *unfolded*) bezüglich einer TBox T , g.d.w. keiner der in C verwendeten Konzept- und Rollennamen auf der linken Seite eines terminologischen Axiomes in T erscheint. Ein Konzeptterm kann expandiert werden, indem die Konzept- und Rollennamen iterativ durch ihre Definitionen ersetzt werden, solange bis alle Konzept- und Rollennamen undefiniert bzgl. T sind. Dieser Algorithmus terminiert, sofern die Terminologie azyklisch ist.

Jeder expandierte Konzeptterm kann in die äquivalente *Negationsnormalform* (engl. *Negation Normal Form, NNF*) transformiert werden. Von einem expandierten Konzeptterm sagt man, er sei in NNF, g.d.w. der Negationsoperator ausschließlich vor den Konzeptnamen steht (z.B. wird $\neg(C \sqcap D)$ zu $\neg C \sqcup \neg D$ transformiert).

Definition 2.3 *Ein Konzeptterm X heißt eingeschränkt bezüglich einer TBox T , g.d.w. sein bezüglich T expandiertes und in NNF gewandeltes Äquivalent X' die folgenden Bedingungen erfüllt:*

1. *Für jeden Teilkonzeptterm C von X' , welcher die Form $\forall R_1.D$ hat, wobei R_1 eine komplexe Rolle ist, enthält D keinen Teilkonzeptterm der Form $\exists R_2.E$, wobei R_2 auch eine komplexe Rolle ist.*
2. *Für jeden Teilkonzeptterm C von X' , der die Form $\exists R_1.D$ hat, wobei R_1 eine komplexe Rolle ist, enthält D keinen Teilkonzeptterm der Form $\forall R_2.E$, wobei R_2 auch eine komplexe Rolle ist.*
3. *Für jeden Teilkonzeptterm C von X' , der die Form $\exists R.D$ oder $\forall R.D$ hat, wobei R eine komplexe Rolle ist, enthält der Konzeptterm D nur solche Terme der Form $\exists u_1, \dots, u_n.P$, die ausschließlich über Attributketten der Länge eins quantifizieren und zudem nicht ihrerseits in einer*

Werte- oder Anzahlrestriktion enthalten sind, die ebenfalls Teilkonzeptterm von D ist.

Eine Terminologie T heißt *eingeschränkt*, g.d.w. auf der rechten Seite aller Axiome in T ausschließlich bezüglich T eingeschränkte Konzeptterme stehen. Eine ABox A heißt *eingeschränkt* bezüglich einer TBox T , g.d.w. T eingeschränkt ist und alle in den assertorischen Axiomen in A verwendeten Konzeptterme bezüglich T eingeschränkt sind. Diese „syntaktische“ Einschränkung verhindert nun die Unentscheidbarkeit der Sprache.

Assertorische Axiome von $\mathcal{ALCRP}(D)$

Seien O_D und O_A zwei disjunkte Mengen von Objektnamen. Die Elemente von O_D heißen konkrete Objekte und die Elemente von O_A abstrakte Objekte. Sei C ein Konzeptterm, R ein Rollenterm, f ein Attributname, P ein Prädikatsname mit der Stelligkeit n , a und b Elemente aus O_A und x_1, \dots, x_n Elemente aus O_D , dann sind die folgenden Ausdrücke assertorische Axiome:

$$a : C, \quad (a, b) : R, \quad (a, x) : f, \quad (x_1, \dots, x_n) : P$$

Eine Interpretation erfüllt ein assertorisches Axiom

$$a : C \quad \text{g.d.w.} \quad a^{\mathcal{I}} \in C^{\mathcal{I}}$$

$$(a, b) : R \quad \text{g.d.w.} \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$$

$$(a, x) : f \quad \text{g.d.w.} \quad f^{\mathcal{I}}(a^{\mathcal{I}}) = x^{\mathcal{I}}$$

$$(x_1, \dots, x_n) : P \quad \text{g.d.w.} \quad (x_1^{\mathcal{I}}, \dots, x_n^{\mathcal{I}}) \in P^{\mathcal{I}}$$

Eine Interpretation \mathcal{I} ist ein Modell für eine ABox A , wenn \mathcal{I} alle assertorischen Axiome in A erfüllt. Eine Interpretation \mathcal{I} ist ein Modell für eine ABox in Bezug auf eine TBox T , wenn \mathcal{I} ein Modell von T ist und zusätzlich alle assertorischen Axiome in A erfüllt.

2.1.5 Räumlich-Terminologisches Schließen mit $\mathcal{ALCRP}(S_2)$

Zum Schlußfolgern über räumliche Gegebenheiten wird die Domäne S_2 verwendet. Wir betrachten in folgenden $\mathcal{ALCRP}(S_2)$, eine Instanz der Sprache $\mathcal{ALCRP}(D)$. Die konkrete Domäne S_2 enthält alle nichtleeren, *regulär abge-*

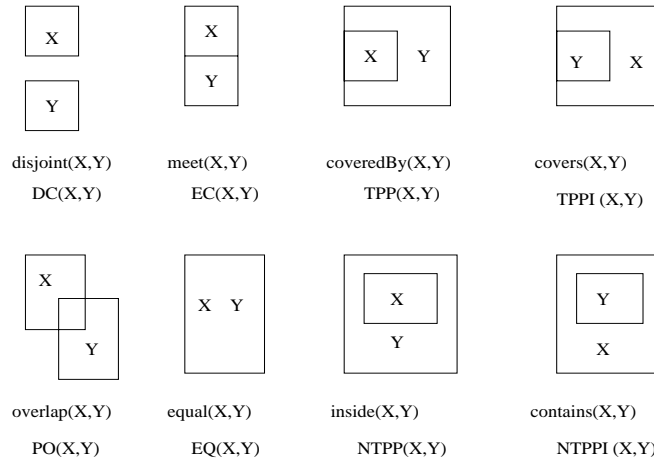


Abbildung 2.3: Beispiele für topologische Relationen in der Ebene.

schlossenen Teilmengen von \mathbb{R}^2 , auch Regionen genannt.⁴

Die Menge der Prädikate in S_2 ist gegenüber der Negation abgeschlossen. Der Beweis für die Zulässigkeit von S_2 ist in [7] zu finden. In S_2 gibt es neben den unären Prädikaten *is-region* und seiner Negation *is-no-region* auch Prädikate, mit denen man qualitative räumliche RCC-8 Beziehungen als Rollen zwischen Objekten beschreiben kann. Die RCC-8 Relationen, eine Entwicklung von Randel et al.1992, sind exhaustiv (d.h. für jede mögliche räumliche Beziehung zwischen zwei Objekten existiert eine beschreibende Basisrelation) und paarweise disjunkt. Das bedeutet, daß zwischen jeweils zwei Regionen im topologischen Raum genau eine der acht Relationen gilt, und daß jegliche räumliche Beziehung zwischen zwei topologischen Gebieten beschrieben werden kann.

Die RCC-8-Relationen sind den acht Relationen von Egenhofer und Franzosa [1991] ([4]) ähnlich. Die in Abbildung 2.3 oben angegebenen Namen werden in [Egenhofer und Franzosa 1991] ([4]) verwendet. Die unten angegebenen Namen bzw. Abkürzungen stammen aus [Randel et al. 1992] ([10]) und sind wie folgt zu lesen.

<i>TPP</i>	<i>TangentialProperPart</i>	<i>PO</i>	<i>PartiallyOverlap</i>
<i>DC</i>	<i>DisConnected</i>	<i>EC</i>	<i>ExternallyConnected</i>
<i>EQ</i>	<i>EQual</i>	<i>NTPP</i>	<i>NonTangentialProperPart</i>

TPPI und NTPPI bezeichnen die Inversen von TPP bzw. NTPP. Zwar kann

⁴Regulär abgeschlossen bedeutet Abgeschlossenheit bzgl. der topologischen regulären Operatoren \cap , \cup und \star .

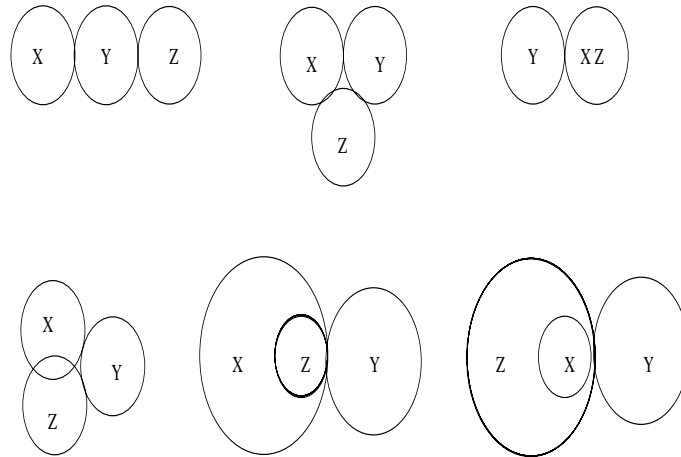


Abbildung 2.4: Schließen über räumliche Relationen.

die Bedeutung dieser Relationen aus der Graphik entnommen werden, doch für eine formale Definition dieser Relationen verweise ich auf [7].

Um nun aber Basisrelationen negieren zu können, werden auch Disjunktionen von Basisrelationen benötigt, um unvollständiges Wissen bzgl. der vorliegenden räumlichen Relation ausdrücken zu können. So ist z.B. das Komplement des Prädikates DC die Disjunktion aller Basisrelation ohne DC .

Sind nun bzgl. einer Menge von Regionen nur einige der Relationen bekannt, so kann anhand der Eigenschaften der bekannten Relationen etwas über die möglichen Relationen zwischen den anderen Regionen geschlossen werden. Das Schließen über räumliche Relationen soll an Hand eines Beispiels erläutert werden. Wir stellen uns drei Regionen X , Y und Z vor, von denen wir die folgenden Beziehungen untereinander kennen. Die Regionen X und Y stehen in der Relation EC . Die Regionen Y und Z stehen in der Relation EC . Die Beziehung zwischen X und Z sei unbekannt (bzw. die Disjunktion aller Basisrelationen). Gesucht sind alle mögliche Beziehungen der Regionen X und Z . Die möglichen Basisrelationen zwischen X und Z findet man in der Kompositionstabelle. Die Kompositionstabelle (auch in [7] zu finden) repräsentiert einschrittiges Schließen. Offensichtlich handelt es sich in den meisten Fällen um eine Disjunktion von Basisrelationen. Z.B. findet man in der RCC-8-Tabelle unter $EC \circ EC = DC \vee EC \vee PO \vee TPP \vee TPPI \vee EQ = DC-EC-PO-TPP-TPPI-EQ$. Diese möglichen Beziehungen zwischen X und Z sind in Abbildung 2.4 dargestellt.

Anhand dieser Basisrelationen (Prädikate) können wir komplexe Rollen definieren. Hierdurch ergibt sich die Möglichkeit, daß aufgrund der Wechsel-

wirkungen zwischen taxonomischen und räumlichen Aspekten von Konzeptbeschreibungen nun interessante Subsumptionsbeziehungen gefunden werden können, die nicht offensichtlich sind, und auch Inkonsistenzen bzgl. räumlicher Beschreibungen werden entdeckt, da die Eigenschaften der Relationen ja durch die konkrete Domäne repräsentiert werden. Angenommen haben wir folgende TBox:

$$\begin{aligned}
\textit{covers} &\doteq \exists(\textit{has_area})(\textit{has_area}).\textit{tpp} \\
\textit{contains} &\doteq (\textit{has_area})(\textit{has_area}).\neg\textit{ntpp} - \textit{tpp} \\
\textit{figure} &\doteq (\textit{has_area}).\textit{is_region} \\
\textit{circle} &\sqsubseteq \textit{figure} \\
\textit{rectangle} &\sqsubseteq \textit{figure} \\
\textit{circle_contains_only_figures} &\doteq \textit{circle} \sqcap \forall\textit{contains}. \textit{figure} \\
\textit{circle_contains_only_circles} &\doteq \textit{circle} \sqcap \forall\textit{covers}. \textit{circle}
\end{aligned}$$

Hier sind zwei komplexe Rollen *covers* und *contains* definiert, wobei *contains* als Oberrolle von *covers* angesehen werden kann. Dann wird ein Konzept *figure* als ein räumliches Gebiet definiert. Die beiden Konzepte *circle* und *rectangle* werden unvollständig definiert. Schließlich werden zwei weitere Konzepte *circle_contains_only_figures* und *circle_contains_only_circles* definiert. Mit $\mathcal{ALCRP}(\mathcal{S}_2)$ kann nun die Subsumptionsbeziehung zwischen den beiden letzt genannten Konzepten berechnet werden. Das Konzept *circle_contains_only_figures* ist nun ein Oberkonzept des Konzeptes *circle_contains_only_circles*.

2.1.6 Zusammenfassung

In diesem ersten Teil dieses Kapitels wurden die Sprachen \mathcal{ALC} , $\mathcal{ALC}(\mathcal{D})$ und $\mathcal{ALCRP}(\mathcal{D})$ eingeführt, und anhand von Beispielen die einzelnen Ausdrucksmöglichkeiten gezeigt. In \mathcal{ALC} ist ein Konzept wie *kleiner_kreis* nicht „richtig“ (tiefgehend) definierbar, wogegen in $\mathcal{ALC}(\mathcal{D})$ ein *kleiner_kreis* als ein *kreis* definiert werden kann, der z.B. einen Radius von < 5 cm besitzt. Da aber in $\mathcal{ALC}(\mathcal{D})$ Rollen nicht definiert werden können, können die Eigenschaften räumlicher Relationen, z.B. die Transitivität der Rolle *enthalten_in* nicht korrekt repräsentiert werden, sodaß Inkonsistenzen nicht erkannt werden. Die ABox $\{a : \textit{kreis}, b : \textit{kreis}, c : \textit{kreis}, (a, b) : \textit{enthalten_in}, (b, c) : \textit{enthalten_in}, (a, c) : \textit{disjunkt}\}$, die in unserem Sinne inkonsistent sein muß, wird z.B. in $\mathcal{ALC}(\mathcal{D})$ nicht als inkonsistent erkannt. Daher benötigen wir schon eine ausdrucksstarke Sprache wie $\mathcal{ALCRP}(\mathcal{D})$, die uns die Definition und den Umgang mit komplexen Rollen (mit Hilfe des rollenbildenden Operators) gestattet, und somit die Repräsentation räumlicher Information

adäquat ermöglicht, zumindest bzgl. topologischer Beziehungen. Natürlich hat so eine Sprache den Nachteil, daß sie eine größere Komplexität mit sich bringt, aber dieser Preis ist meistens für solche ausdrucksstarke Sprachen zu zahlen.

Wir sind nun in der Lage, $\mathcal{ALCRP}(\mathcal{S}_2)$ in Anwendungsdomänen wie geographischen Informationssystemen (GIS) erfolgreich und gewinnbringend einzusetzen. So können wir z.B. ein GIS mit komplexeren Anfragemöglichkeiten ausstatten. Im folgenden sind einige denkbare Anfragen beispielhaft in natürlicher Sprache gegeben:

- Welche Wohngebiete kommen in Frage, wenn eine Schwimmhalle in der Nähe gewünscht wird?
- In welchem Gebiet kann eine Schule, ohne Gefährdung der Kinder durch Überlandleitungen, gebaut werden?

Offensichtlich sind hier komplexe Wechselwirkungen bzgl. räumlich-thematischer bzw. räumlich-terminologischer Zusammenhänge zu berücksichtigen, wofür $\mathcal{ALCRP}(\mathcal{S}_2)$ eine saubere Fundierung bereitstellen könnte.

2.2 Default-Schließen

Default-Schließen kann als die Fähigkeit, rationale Ergebnisse in Anwesenheit von widersprüchlichen (auch unvollständigen) Informationen zu erhalten, beschrieben werden. Ein Problem dabei ist, daß rationale Ergebnisse ihre Gültigkeit verlieren können, wenn neue Informationen hinzugefügt werden, die nicht als logische Konsequenzen aus dem bisherigen Wissen folgen. Klassische Logiken können dies nicht behandeln, solange alles Wissen als universell angenommen wird, da es dort keine Ausnahmen gibt. Folglich werden zur Behandlung des Problems andere Typen von Formalismen benötigt.

In der Literatur finden sich viele Formalismen und Theorien, um einige zu nennen, Inheritance Networks, Default-Logik und Circumscription, sowie Formalismen, die mit einer Präferenz-Semantik ausgestattet sind. Die wesentliche Eigenschaft dieser Formalismen ist die *Nichtmonotonie*, während klassische Prädikatenlogik *monoton* ist.

In diesem Abschnitt folgt eine kurze Erläuterung einiger Ansätze zur Formalisierung nichtmonotonen Schließens, insbesondere von Reiters Default-Logik.

2.2.1 Motivation

Klassische Logiken (z.B. Prädikatenlogik) können nur striktes (allgemeingültiges) Wissen ausdrücken. Menschen aber sind geneigt, Schlüsse zu ziehen und somit Ergebnisse zu erreichen, auch wenn diese nicht als strikte Konsequenzen aus dem Gewußten folgen. Als ein Beispiel betrachten wir folgende Aussagen:

Tweety ist ein Vogel, und Vögel können normalerweise fliegen. Auf die Frage, ob Tweety fliegen kann, würden viele antworten: „Ja“. Obwohl wir hier unvollständige Informationen vorliegen haben (woher wissen wir, daß Tweety ein „normaler“ Vogel ist?), können wir eine plausible Folgerung schließen. Fügen wir nun zu unseren Aussagen die Aussage hinzu, daß Tweety ein Pinguin ist, so schließen wir, daß Tweety nicht fliegt und auch, daß Tweety kein „normaler“ Vogel ist. Wir sind hier in der Lage, Schlüsse zu revidieren („Tweety fliegt“ wird zurückgenommen). Man möchte also gerne auch aus partiellen Informationen etwas schließen können. Im obigen Beispiel wäre eine Möglichkeit, Default-Regeln einzuführen. Solche Regeln können ausdrücken, was typischerweise der Fall ist. Mit einer Default-Regel könnten wir ableiten, daß Tweety fliegt, sofern es konsistent ist anzunehmen, daß Tweety ein „normaler Vogel“ ist.

Die Methoden und Formalismen, die Computer zum Schließen mit unvollständigen Informationen brauchen, entstanden aus dem „commonsense reasoning“ (Repräsentation von bzw. Schlußfolgern mit Alltagswissen) und sind ein breites Feld in der KI. Ein wesentliches Charakteristikum des „commonsense reasoning“ ist die Nichtmonotonie. Es gibt klassische Logiken wie propositionale Logik, die monotone Formalismen sind, d.h. bei der Addition eines neuen Faktums kann die Gültigkeit von alten Konklusionen nicht verletzt werden (evtl. wird jedoch durch die Addition eines Faktums die Weltbeschreibung inkonsistent). Diese Logiken heißen monoton, weil die Menge der ableitbaren Theoreme monoton mit der Menge der Assertionen zunimmt [13]. Nichtmonotone Formalismen weisen diese Eigenschaft nicht auf. Die Nichtmonotonie entsteht u.a., wenn wir Regeln mit Ausnahmen zulassen.

Es hat auch schon verschiedene Ansätze gegeben (Brachman et al. 1991; Mac Gregor & Bates 1987, Quantz und Royer (1992) und Padgham und Nebel (1993), u.s.w.), mit denen versucht wurde, beschreibungslogische Systeme um Defaults zu erweitern (aus [5]).

2.2.2 Default-Logik und einige andere nichtmonotone Logiken

Zu den wichtigsten Ansätzen zur Formalisierung des nichtmonotonen Schließens gehören:

- Autoepistemische Logik (McDermott, Doyle und Moore),
- Circumscription (McCarthy, Lifschitz),
- Default-Logik (Reiter)

Wie wir weiter unten sehen werden, können all diese Logiken verwendet werden, um einige der angesprochenen Problempunkte klassischer Logiken zu adressieren. Im folgenden werde ich die autoepistemische Logik sowie die Circumscription auf die wichtigsten Grundideen beschränken (s. [5][Kap. 1.2]) und anschließend Reiters Default-Logik ausführlicher behandeln.

Autoepistemische Logik

Moore, McDermott und Doyle führen einen modalen Operator L ein, mit dem Zweck auszudrücken, was geglaubt wird und was nicht. Das oben angesprochene Vogel- bzw. Tweety-Beispiel könnte durch folgende Implikation repräsentiert werden:

$$Vogel(x) \wedge \neg L\neg Fliegt(x) \rightarrow Fliegt(x)$$

Intuitiv bedeutet diese Regel: Vögel, von denen wir nicht wissen, daß sie nicht fliegen können, fliegen (m.a.W., wir können glauben, daß ein solcher Vogel fliegt). Diese Regeln haben große Ähnlichkeit mit den Default-Regeln, welche wir unten kennenlernen werden. Tatsächlich kann man diese Regeln in Default-Regeln übersetzen, aber auch umgekehrt:

$$(\alpha : \beta_1, \dots, \beta_n / \gamma \implies L\alpha \sqcap \neg L\neg\beta_1 \sqcap \dots \sqcap \neg L\neg\beta_n \rightarrow \gamma)$$

Circumscription

McCarthy schlägt als allgemeine Methode zum Default-Schließen mit Circumscription die Verwendung sogenannter Abnormalitätsprädikate vor. Default-Regeln, wie unser Vogelbeispiel, werden dabei folgendermaßen repräsentiert:

$$\forall x.Vogel(x) \sqcap \neg Ab(x) \rightarrow Fliegt(x)$$

Die intuitive Bedeutung der Regel ist: Vögel, die nicht abnormal sind, fliegen. Circumscription ist eine Technik, die es uns erlaubt, die Extension bestimmter ausgewählter Prädikate (hier Ab-Prädikate) zu minimieren. Leider hat sich herausgestellt ([5]), daß komplexere Versionen der Circumscription als die vorgestellte nötig sind, um adäquate Resultate zu erzielen.

2.2.3 Reiters Default-Logik und der Begriff der Extension

Reiter entwickelte eine Theorie zum Default-Schließen. Er verwendete in seiner Theorie sog. Default-Regeln, welche folgende Form haben:

$$D = \alpha : \beta_1, \dots, \beta_n / \gamma$$

α ist die Prämisse oder die Vorbedingung (engl. Prerequisite), auch als $Pre(D)$ notiert,

β_1, \dots, β_n sind die Konsistenzannahmen (engl. Justifications), auch als $Jus(D)$ notiert,

γ ist die Konsequenz oder Konklusion (engl. Consequent), auch als $Con(D)$ notiert.

Dies heißt, wenn die Prämisse α geglaubt ($\alpha \in W$ oder $W \models \alpha$) wird und die Annahmen β_1, \dots, β_n mit dem Geglaubten konsistent sind, dann glaube γ . α , β_i und γ sind üblicherweise prädikatenlogische Formeln erster Stufe. Das beliebte Vogel-Beispiel sieht nun wie folgt aus:

$$Vogel(x) : Fliegt(x) / Fliegt(x)$$

Intuitiv bedeutet diese Regel: Vögel, von denen man annehmen kann, daß sie fliegen, fliegen auch.

Definition 2.4 (Reiters Default-Theorie) Eine Default-Theorie ist ein Paar (W, D) , wobei W unsere Weltbeschreibung ist und D eine Menge von Default-Regeln ist.

Eine Default-Regel ist *offen*, wenn sie freie Variable x enthält, sonst ist sie *geschlossen*. Eine Default-Theorie ist geschlossen, wenn alle Defaults in der Theorie geschlossen sind. Eine Default-Regel ist *semi-normal* genau dann, wenn ihre Annahme (justification) die Konsequenz impliziert (aus der einzi-

gen Annahme folgt die Konsequenz), und sie ist *normal* genau dann, wenn ihre einzige Annahme mit der Konsequenz identisch ist ($\alpha : \beta/\beta$). Reiter hat gezeigt, daß normale Default-Theorien immer eine Extension besitzen [2]. Im beschreibungslogischen Fall (*terminologische Default-Theorie*) ist die Weltbeschreibung W eine beschreibungslogische Wissensbasis, also ein Paar aus TBox T und ABox A , $W = (T, A)$, und die Formeln α , β_i und γ sind Konzeptterme einer Basisbeschreibungslogik.

Reiter bezeichnet die Menge von Konsequenzen einer *geschlossenen* Default-Theorie als *Extensionen*. Eine Extension repräsentiert eine Menge von Annahmen (zusammen mit dem sicheren Wissen), welche aus einer Default-Theorie folgt. Eine Default-Theorie könnte verschiedene Extensionen haben, die für verschiedene Glaubenssätze stehen. Eine Formel ist eine *skeptische* Konsequenz einer Default-Theorie, wenn sie zu der Schnittmenge aller Extensionen gehört (sceptical reasoning). Wenn die Formel zu wenigstens einer Extension gehört, dann ist sie eine *leichtgläubige* Konsequenz (credulous reasoning).

Extensionen sind jedoch ausschließlich für geschlossene Default-Theorien definiert. Eine Möglichkeit, eine geschlossene Default-Theorie zu erhalten, ist die von Reiter vorgeschlagene *Skolemisierung*. In [2] wird aber gezeigt, daß diese Behandlung offener Defaults in der Beschreibungslogik \mathcal{ALCF} zur Unentscheidbarkeit des Konsequenzproblems bzw. der Ableitbarkeitsrelation führt.⁵ Baader & Hollunder [2] verwenden daher eine *eingeschränkte Semantik* ohne Skolemisierung für *terminologische Default-Theorien* (s.u.), in der eine geschlossene Default-Theorie durch Instantiierung der in den Defaults vorkommenden freien Variablen mit allen in einer ABox *explizit* vorhandenen Individuen erhalten wird. Diese eingeschränkte Semantik stellt die Entscheidbarkeit sicher⁶ (s.u.).

Sobald wir eine geschlossene Default-Theorie vorliegen haben, ist eine Extension dieser Theorie wie folgt definiert:

Definition 2.5 (Reiters Extension) Sei (W, D) eine geschlossene Default-Theorie, und sei E eine Menge von geschlossenen Formeln. Dann definieren wir $E_0 := W$ und für alle $i \geq 0$

$$E_{i+1} := E_i \cup \{ \gamma \mid \alpha : \beta_1, \dots, \beta_n / \gamma \in D, \alpha \in Th(E_i), \text{ und } \neg\beta_1, \dots, \neg\beta_n \notin Th(E) \}.$$

⁵Es wäre zu überprüfen, ob das gleiche Problem mit $\mathcal{ALCRP}(S_2)$ gibt.

⁶Wir sind mit dieser Idee auch mit $\mathcal{ALCRP}(S_2)$ auf der sicheren Seite

Dann ist $Th(E)$ eine Extension von (W, D) genau dann, wenn

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i).$$

Hier steht $Th(\Gamma)$ für die deduktive Hülle einer Menge von Formeln Γ . Da wir bei obiger Iteration nur die Konsequenzen der Defaults dazuaddieren, hat eine Extension $Th(E)$ von (W, D) die Form $Th(W \cup Con(\hat{D}))$ für eine Teilmenge \hat{D} von D . \hat{D} heißt auch die Menge der *generierenden Defaults* (engl. „Generating Defaults“) einer Extension E .

Es gibt auch Default-Theorien, die keine Extensionen besitzen.:

Sei $W = \{ \}$ und $D = \{ : Vogel(Tweety) / \neg Vogel(Tweety) \}$.

Diese Default-Theorie besitzt keine Extension, da die Konsequenz des Defaults im Widerspruch zu seiner Annahme steht. Da hier die Prämisse α immer geglaubt wird, darf sie weggelassen werden. Extensionen können als mögliche Welten gesehen werden, die nach der Anwendung von Defaults auf die Wissensbasis entstehen.

Das Ermitteln der Extensionen kann auch, wie im nächsten Kapitel gezeigt wird, durch andere Algorithmen geschehen. Manchmal ist man nicht an allen Extensionen interessiert, sondern nur an Teilmengen davon. Die Grundidee besteht darin, Defaults zu priorisieren. Wir werden später darauf zurückkommen.

Beispiel: Default-Theorie

$W = \{ Sportler(Michael), Raucher(Michael) \}$ und
 $D = \{ Raucher(x) : \neg Sprinter(x) / \neg Sprinter(x),$
 $Sportler(x) : Sprinter(x) / Sprinter(x) \}$

Hieraus entstehen zwei geschlossene Defaults

$D = \{ Raucher(Michael) : \neg Sprinter(Michael) / \neg Sprinter(Michael),$
 $Sportler(Michael) : Sprinter(Michael) / Sprinter(Michael) \}$

Wir erhalten die geschlossenen Defaults, indem wir alle freien Variablen der Defaults (in D) durch alle Individuen, die in A (der ABox unserer Wissensbasis $W = (T, A)$) vorkommen (in unserem Fall „Michael“), instantiieren.

Unsere Wissensbasis kann andere Konzepte beinhalten, wie z.B. „Forscher“, doch dies ist irrelevant für unsere Defaults, da wir keine Default-Regeln haben, welche überhaupt „Forscher“ als Vorbedingung haben. Doch wenn es hier andere Individuen gäbe (genauer in der ABox), wie z.B. Peter, und er ist auch Raucher (also $Raucher(Peter)$), dann würde der Default $Raucher(x) : \neg Sprinter(x) / \neg Sprinter(x)$ auch für Peter feuern, da die Variable x durch alle Individuen (auch durch $Peter$) instantiiert würde. Da die ABox und die Menge aller Defaults endlich sind, bekommen wir durch diese Methode eine endliche Menge von geschlossenen Defaults. Diese Theorie hat zwei Extensionen:

$$E_1 = \{Raucher(Michael), Sportler(Michael), \neg Sprinter(Michael)\}$$

$$E_2 = \{Raucher(Michael), Sportler(Michael), Sprinter(Michael)\}$$

Obwohl wir uns leicht vorstellen können, was diese einzelnen Extensionen bedeuten und wie sie entstehen können (s. Definition der Extensionen), so ist es dennoch nicht trivial, einen Algorithmus anzugeben, der diese auch berechnet. Vor allem, wenn es eine Vielzahl von Defaults gibt, und wenn unsere Domäne (Wissensbasis) umfangreicher wird. Dennoch werden wir später einige Algorithmen kennenlernen.

Kapitel 3

Algorithmen zur Berechnung von Extensionen

Im vorherigen Kapitel haben wir Reiters Extensionen kennengelernt. Anhand der Definition der Reiterschen Extensionen kann man sich einen direkten Algorithmus überlegen, bzw. implementieren. Dieser sog. *definitionsbasierte Algorithmus* zur Berechnung der Extensionen wird zunächst erläutert. Danach wird ein Ansatz von Schwind und Risch geschildert. Schließlich wird ein Algorithmus von Baader und Hollunder dargestellt, der auf dem Ansatz von Schwind und Risch aufbaut und dadurch das Ermitteln der Extensionen effizienter macht.

Im folgenden werden ausschließlich geschlossene terminologische Default-Theorien mit der eingeschränkten Semantik (keine Skolemisierung) nach Baader & Hollunder behandelt. Als Basissprache wird eine entscheidbare Beschreibungslogik angenommen, z.B. \mathcal{ALC} . Die Weltbeschreibung W ist nun eine ABox A . Die Formeln α, β_i und γ seien beschreibungslogische Konzeptterme der Basislogik, welche durch Instantiieren mit allen in der ABox A explizit genannten Individuen geschlossen werden. Dann handelt es sich bei den geschlossenen Formeln nicht mehr um Konzeptterme, sondern um ABox-Konzeptzugehörigkeitsassertionen (ABox Concept Membership Assertions). Das entsp. Inferenzproblem zur Entscheidung, ob aus einer ABox A (und einer TBox T) die Assertion $i : C$ für ein Individuum i gilt heißt das Instanzerkennungsproblem (Instance Checking Problem), welches entscheidbar sein muß. Sofern nur einfache terminologische Axiome in der TBox T vorkommen, kann durch Expandieren der Terme in der ABox A die TBox T verworfen werden. Somit ist ein ABox-Konsistenzprüfer hier ausreichend.

3.1 Der definitionsbasierte Algorithmus

Wir führen uns an dieser Stelle nochmal die Definition der Reiterschen Extensionen vor Augen:

Definition 3.1 (Reiters Extension) Sei (W, D) eine geschlossene Default-Theorie, und sei E eine Menge von geschlossenen Formeln. Dann definieren wir $E_0 := W$ und für alle $i \geq 0$

$$E_{i+1} := E_i \cup \{ \gamma \mid \alpha : \beta_1, \dots, \beta_n / \gamma \in D, \alpha \in Th(E_i), \text{ und } \neg\beta_1, \dots, \neg\beta_n \notin Th(E) \}.$$

Dann ist $Th(E)$ eine Extension von (W, D) genau dann, wenn

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i).$$

Bei dieser Definition handelt es sich um eine Art iterative Konstruktionsvorschrift: beginnend mit $E_0 = W$ werden in jedem Iterationsschritt die Konklusionen γ_i einiger Defaults $d \in \mathcal{D}$ hinzugenommen, sofern diese Defaults d anwendbar sind. Da wir eine endliche Menge von Defaults \mathcal{D} betrachten, wird irgendwann für irgendein i $E_{i+1} = E_i$ sein, sodaß der „Fixpunkt“ erreicht wird und die Iteration endet. Am Ende des Iterationsprozesses wird geprüft, ob durch die iterative Konstruktion tatsächlich eine Extension konstruiert wurde, d.h., ob die Extension sich selbst durch Nachvollziehen der Konstruktion rekonstruiert hat. Dies ist der Fall, wenn $Th(E) = \bigcup_{i=0}^{\infty} Th(E_i)$.

Zur Bestimmung der Anwendbarkeit eines Defaults in der Konstruktion muß zum einen die Vorbedingung in der bis zu diesem Iterationsschritt konstruierten Theorie $Th(E_i)$ enthalten sein, $\alpha \in Th(E_i)$ bzw. $E_i \models \alpha$. Offensichtlich muß die zugrundeliegende Basislogik daher entscheidbar sein. Der Operator $Th(E_i)$ bildet den deduktiven Abschluß der Formelmenge E_i . Existiert für die zugrundeliegende Basislogik ein korrekter und vollständiger Kalkül, so gilt $E_i \models \alpha$ gdw. $\alpha \in Th(E_i)$ (bzw. $E_i \vdash \alpha$). Zum anderen muß zur Bestimmung der Anwendbarkeit eines Defaults jedoch auch geprüft werden, ob $\neg\beta_1, \dots, \neg\beta_n \notin Th(E)$ gilt. Da hier $Th(E)$ verwendet wird (und nicht $Th(E_i)$), ist diese Iterationsvorschrift leider nicht konstruktiv, denn die zu konstruierende Extension $Th(E)$ wird an dieser Stelle bereits benötigt. Daher kann diese Definition nicht direkt in einen iterativen Algorithmus

überführt werden. Glücklicherweise hat jede Extension jedoch immer die Form $Th(W \cup Con(\mathcal{D}'))$ (mit $Con(\mathcal{D}')$ ist γ von D gemeint) für eine Menge $\mathcal{D}' \subseteq \mathcal{D}$ sog. *generierender Defaults*. Daher können wir nichtdeterministisch eine Menge generierender Defaults \mathcal{D}' raten, deren Konklusionen zu W addiert konsistent sind. Ein solcher geratener Kandidat E für eine Extension $Th(E)$ kann dann mit Hilfe der Definition daraufhin überprüft werden, ob es sich bei dem Kandidaten tatsächlich um eine Extension handelt.

Eine der wesentlichen Schwäche dieses Algorithmus ist, daß in jedem Iterationsschritt auch die β 's betrachtet werden, und daß hier alle Teilmengen der Defaults betrachtet werden. Dadurch benötigen wir für das Berechnen der Extensionen viel Zeit. Es gibt aber auch andere Ansätze, die das Problem etwas anders handhaben.

3.2 Das Theorem von Schwind und Risch

In der Literatur kann man einige Methoden finden, die alle Extensionen berechnen. Hier werde ich die Methode von Schwind und Risch [12] kurz vorstellen, die nicht alle Teilmengen der Defaults betrachtet.

Im Prinzip basiert diese Methode auf der Tatsache, daß eine Extension einer geschlossenen Default-Theorie (W, D) ¹ die Form $Th(W \cup Con(\hat{D}))$ für eine Teilmenge \hat{D} von D besitzt. Wenn D endlich ist, dann gibt es auch nur endlich viele Teilmengen, und das einzige Problem ist, herauszufinden, welche dieser eine Extension generiert. Sicher könnte man alle Teilmengen von D durch obigen Iterationsprozeß überprüfen lassen, und der Prozeß würde auch terminieren, da D endlich ist, wie beim definitionsbasierten Algorithmus. Doch im Gegensatz zum definitionsbasierten Algorithmus werden hier nur die relevanten Teilmengen betrachtet, und dadurch die Suche nach einer Menge von *generierenden* Defaults einer Extension effizienter gemacht.

Schwind und Risch geben ein Theorem an, welches die Teilmengen \hat{D} von D charakterisiert, die die Menge der generierenden Defaults einer Extension einer geschlossenen Default-Theorie (W, D) ist. Bevor wir das Theorem formulieren, müssen wir uns noch notieren:

Definition 3.2 (Verankerung (engl. „Groundedness”)) Sei W eine Menge von geschlossenen Formeln und D eine Menge von geschlossenen Defaults.

¹Statt W kann jetzt schon A für ABox stehen, da wir geschlossene Default-Theorien betrachten

Wir definieren $D_0 = \emptyset$ und für $i \geq 0$

$$D_{i+1} = D_i \cup \{d = \alpha : \beta_1, \dots, \beta_n / \gamma \mid d \in D \text{ und } W \cup \text{Con}(D_i) \models \alpha\}$$

Dann heißt D verankert (prinzipiell anwendbar) in W genau dann, wenn

$$D = \bigcup_{i=0}^{\infty} D_i.$$

In anderen Worten ausgedrückt heißt dies, daß man nur an solchen Defaults interessiert ist, deren Vorbedingung ableitbar ist (aus $W \cup \text{Con}(D_i) \models \alpha$).

Wenn D nicht in W verankert ist, dann ist $\bigcup_{i=0}^{\infty} D_i$ die größte Teilmenge von

D , welche in W verankert ist. Dieser Iterationsprozeß ($\bigcup_{i=0}^{\infty} D_i$) korrespondiert

zu der Iteration in der Definition der Extensionen (Def. 5.2), aber mit dem Unterschied, daß hier die Annahmen (Justifications) nicht betrachtet werden.

Um den Begriff verankert besser zu verstehen, ist es besser ein Beispiel zu inspizieren.

Beispiel: Verankert

Sei $W = \{\text{Junge}(\text{Klaus})\}$ und

die Defaults D_i (hier zur Vereinfachung numeriert)

$$\begin{aligned} D_1 &= \{\text{Junge}(x) : \text{Student}(x) / \text{Student}(x)\}, \\ D_2 &= \{\text{Junge}(x) : \text{Sportler}(x) / \text{Sportler}(x)\}, \\ D_3 &= \{\text{Mann}(x) : \text{Trinker}(x) / \text{Trinker}(x)\}, \\ D_4 &= \{\text{Student}(x) : \text{Arbeiter}(x) / \text{Arbeiter}(x)\} \text{ und} \\ D_5 &= \{\text{Sportler}(x) : \neg \text{Raucher}(x) / \text{Sprinter}(x)\} \end{aligned}$$

Welche dieser Defaults sind nun verankert? Die Vorbedingung von D_1 und D_2 ist schon direkt in W vorhanden, also sind diese schon prinzipiell anwendbar. Durch ihre Anwendung werden die Vorbedingungen von D_4 und D_5 ableitbar, sodaß wir diese zu unserer gesuchten Menge der verankerten Defaults hinzunehmen können. Bis auf D_3 sind alle Defaults verankert.

Mit Hilfe der Mengen der verankerten Defaults lassen sich nun die möglichen Extensionen einer Default-Theorie charakterisieren:

Theorem 3.1 (Schwind und Risch) Sei (W, D) eine geschlossene Default-Theorie. Eine Teilmenge \hat{D} von D ist eine Menge von generierenden Defaults (engl. „generating Defaults“) von einer Extension von (W, D) genau dann, wenn die folgenden Bedingungen erfüllt sind:

1. \hat{D} ist verankert in W
2. Für alle $d \in D$ mit $d = \alpha : \beta_1, \dots, \beta_n / \gamma$ haben wir $d \in \hat{D}$ genau dann,
 - wenn $W \cup \text{Con}(\hat{D}) \models \alpha$
(bzw. $W \cup \text{Con}(\hat{D}) \cup \{-\alpha\}$ inkonsistent ist)
 - und für alle i , $1 \leq i \leq n$, $W \cup \text{Con}(\hat{D}) \not\models \neg\beta_i$
(bzw. $W \cup \text{Con}(\hat{D}) \cup \{\beta_i\}$ konsistent ist).

Nur wenn D endlich und das Schlußfolgerungsproblem (\models) in der Basissprache entscheidbar ist, dann liefert uns dieses Theorem einen effektiven Test, nämlich ob eine Teilmenge \hat{D} von D eine Menge von generierenden Defaults einer Extension von (W, D) ist.

Wenn W (Weltbeschreibung) inkonsistent ist, dann haben wir nur eine Extension (die Menge aller Formeln). Im folgenden gehen wir von der Annahme aus, daß W konsistent ist. Nun seien D_1, \dots, D_n die maximalen Teilmengen von D , die mit $W \cup \text{Con}(D_i)$ konsistent sind. Wenn wir davon ausgehen, daß W konsistent ist, dann sind die Extensionen auch konsistent. Dies bedeutet, daß eine generierende Menge von Defaults einer Extension eine Teilmenge von einer der D_i 's ist. Die Idee dieser Methode ist, mit solch einer maximalen Teilmenge D_i anzufangen, und sukzessive alle Defaults zu eliminieren, die die erste Bedingung des Theorems verletzen (Verankerung). Und wenn keine zu eliminierenden Defaults mehr übrigbleiben, dann wird der Wenn-Teil der zweiten Bedingung getestet.

Aufbauend auf diesem Theorem wird von Baader und Hollunder in [2] ein Verfahren geschildert, das es uns erlaubt, alle Mengen von generierenden Defaults und anhand dieser die möglichen Extensionen zu berechnen, ohne alle Teilmengen von D zu betrachten.

3.3 Algorithmus von Baader und Hollunder

In der unten dargestellten Prozedur werden die folgenden Subprozeduren verwendet.

- Ist W konsistent?
- Berechnung der maximalen Teilmengen D' von D , die mit $W \cup \text{Con}(D')$ konsistent sind.

- Berechnung der größten in W verankerten Teilmenge D_0 von D .
- Berechnung aller maximaler Teilmengen D'' von D_0 , für die $W \cup Con(D'') \not\models \neg\beta_i$ gilt.

Diese Subprozeduren werden im weiteren Verlauf noch genauer erläutert.

```

Berechne-alle-Extensionen( $W, D$ )
beginne
  Wenn  $W$  inkonsistent ist
    dann drucke „inkonsistente Weltbeschreibung“
    ansonsten für alle maximalen Teilmengen  $D'$  von  $D$ ,
      die mit  $W \cup Con(D')$  konsistent sind
        Berechne-generierende-Defaults( $W, D, D'$ );
ende

Berechne-generierende-Defaults( $W, D, D'$ )
beginne
  Sei  $D_0$  die größte Teilmenge von  $D'$ , die in  $W$  verankert ist;
  wenn  $W \cup Con(D_0) \models \neg\beta_i$  für einige Annahmen  $\beta_i \in Jus(D_0)$ 
    dann soll  $d = \alpha : \beta_1, \dots, \beta_n / \gamma$  der betreffende Default sein;
    Berechne-generierende-Defaults( $W, D, D_0 \setminus \{d\}$ );
    für alle maximalen Teilmengen  $D''$  von  $D_0$ , für die
       $d \in D''$  und  $W \cup Con(D'') \not\models \neg\beta_i$  ist,
        Berechne-generierende-Defaults( $W, D, D''$ );
  sonst wenn für alle  $\alpha : \beta_1, \dots, \beta_n / \gamma \in D \setminus D_0$ 
    entweder  $W \cup Con(D_0) \not\models \alpha$ 
    oder  $W \cup Con(D_0) \models \neg\beta_i$  für einige  $i$  gilt
    dann add.  $D_0$  zur Liste der generierenden Defaults
ende
    
```

Prozedur zum Berechnen der Menge der Mengen generierender Defaults von allen Extensionen der geschlossenen Default-Theorie (W, D)

In der obigen Prozedur wird zuerst getestet, ob W inkonsistent ist. Dieser Test ist schon in der Basissprache gegeben. Wenn W konsistent ist, dann nimmt man sich die maximalen Teilmengen D' von D vor, die mit $W \cup Con(D')$ konsistent sind. Man will also solche maximalen Mengen von Defaults betrachten, deren Konklusionen mit W konsistent sind. Dies kann man auch erreichen, indem man die minimal inkonsistenten Mengen bildet und deren Komplement betrachtet. Die zweite Prozedur wird mit jeder dieser Mengen aufgerufen und entfernt widersprüchliche und überflüssige Defaults aus dieser Menge. Wird kein solcher Default gefunden, dann gehört die Menge D_0 zur Menge der Mengen generierender Defaults. Zunächst werden

die Defaults gestrichen, die nicht in W verankert (also prinzipiell nicht anwendbar) sind. So erhalten wir die Menge D_0 , $W \cup \text{Con}(D_0) \models \alpha$. Jetzt wird getestet, ob für irgendeine Annahme β_i eines beliebigen Defaults d gilt: $W \cup \text{Con}(D_0) \models \neg\beta_i$ (was bisher noch nicht geschehen ist).

- Fällt der Test positiv aus, so gibt es zwei Möglichkeiten:
 1. Der dafür verantwortliche Default d wird aus der Menge D_0 entfernt und wir rufen die Prozedur rekursiv auf.
 2. Der Default d wird nicht entfernt. Allerdings müssen wir jetzt dafür sorgen, daß seine widersprüchliche Annahme $\neg\beta_i$ nicht mehr hergeleitet werden kann. Wiederum sollen diese Mengen maximal sein, und für jede dieser maximalen Teilmengen von D_0 wird die Prozedur erneut rekursiv aufgerufen.
- Fällt der Test jedoch negativ aus, so wird in einem zweiten Test geprüft, ob die momentan betrachtete Menge D_0 auch wirklich maximal ist. Dies bedeutet, daß im Komplement von D_0 ($D_0 = D \setminus D_0$) keine Defaults mehr vorhanden sein dürfen, die sowohl verankert und deren Annahmen konsistent mit D_0 und der Weltbeschreibung W sind (denn ein solcher Default sollte zu D_0 gehören!). Wird kein solcher Default gefunden, dann gehört die Menge D_0 zur Menge der Mengen generierender Defaults.

Anhand eines Beispielen soll verdeutlicht werden, wie dieser Algorithmus arbeitet, wobei das folgende Nixon-Beispiel verwendet wird:

Beispiel: Nixon-Diamond

geg: $W = \{nixon(nixon)\}$
 $D_1 = \{nixon(x) : \top(x)/quaker(x)\}$
 $D_2 = \{nixon(x) : \top(x)/republican(x)\}$
 $D_3 = \{quaker(x) : \neg hawk(x)/dove(x)\}$
 $D_4 = \{republican(x) : \neg dove(x)/hawk(x)\}$
 $D_5 = \{hawk(x) : \top(x)/politically_motivated(x)\}$
 $D_6 = \{dove(x) : \top(x)/politically_motivated(x)\}$

Ersetzt man die freie Variable x in den Defaults durch das Individuum $nixon$ (welches hier auch Instanz des Konzeptes $nixon$ ist), dann erhält man die

geschlossenen Defaults, für die wir die Extensionen berechnen können. Da sich D_3 und D_4 gegenseitig ausschließen, können ihre Konklusionen nicht gemeinsam in einer Extension vorkommen. Deshalb ergeben sich erwartungsgemäß zwei Extensionen. In jedem Fall ist Nixon politisch motiviert ($politically_motivated(nixon)$), sodaß $politically_motivated(nixon)$ eine skeptische Konsequenz dieser Default-Theorie ist.

$$E_1 = \{ nixon(nixon), quaker(nixon), republican(nixon), \\ dove(nixon), politically_motivated(nixon) \}$$

$$E_2 = \{ nixon(nixon), quaker(nixon), republican(nixon), \\ hawk(nixon), politically_motivated(nixon) \}$$

Hingegen sind sowohl $dove(nixon)$ als auch $hawk(nixon)$ leichtgläubige Konsequenzen, da sie nur in jeweils einer der zwei Extensionen vorkommen.

Es folgt nun das Protokoll für das Nixon-Beispiel. Im Protokoll wird eine LISP-Präfix-Syntax verwendet, da der Algorithmus in dieser Sprache implementiert wurde. So steht (DOVE NIXON) für das ABox-Axiom $nixon : dove$, wobei $nixon$ das Individuum und $dove$ der entsp. Konzeptterm ist. NIL steht für die leere Liste bzw. die leere Menge, z.B. für keine Annahmen β bzw. $\beta = \top$. Einrückungen des Protokolls deuten die Aufruftiefe der rekursiven Prozedur an.

Der Algorithmus wird zunächst mit der Menge aller Defaults D' und dem sicheren Wissen W aufgerufen.

Momentan betrachtete Menge von Defaults D' :

```
((<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
  <(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
  <(REPUBLICAN NIXON) : (NOT (DOVE NIXON)) / (HAWK NIXON)>
  <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
  <(NIXON NIXON) : NIL / (QUAKER NIXON)>
  <(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>))
```

Sicheres Wissen W bzw. A :

```
((NIXON NIXON))
```

Prinzipiell anwendbare Defaults DO bzgl. W :

```
((<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>
  <(NIXON NIXON) : NIL / (QUAKER NIXON)>
  <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
  <(REPUBLICAN NIXON) : (NOT (DOVE NIXON)) / (HAWK NIXON)>))
```

KAPITEL 3. ALGORITHMEN ZUR BERECHNUNG VON EXTENSIONEN⁴⁵

```
<(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>  
<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
```

Konsequenzen von W und D₀:

```
((NIXON NIXON) (REPUBLICAN NIXON) (QUAKER NIXON) (DOVE NIXON)  
(HAWK NIXON) (POLITICALLY-MOTIVATED NIXON))
```

Widerspruechlicher Default:

```
<(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>  
Beta (NOT (HAWK NIXON))
```

Konflikt mit:

```
(HAWK NIXON)
```

Entferne <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)> aus D'

Da in der Menge D_0 ein Default gefunden wurde, dessen β_i in Konflikt mit D_0 steht, wird dieser zunächst entfernt. Die Prozedur wird mit der Menge D_0 ohne den kritischen *Quaker(Nixon)*-Default rekursiv aufgerufen:

Momentan betrachtete Menge von Defaults D':

```
(<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>  
<(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>  
<(REPUBLICAN NIXON) : (NOT (DOVE NIXON)) / (HAWK NIXON)>  
<(NIXON NIXON) : NIL / (QUAKER NIXON)>  
<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>)
```

Sicheres Wissen W bzw. A:

```
((NIXON NIXON))
```

Prinzipiell anwendbare Defaults D₀ bzgl. W:

```
(<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>  
<(NIXON NIXON) : NIL / (QUAKER NIXON)>  
<(REPUBLICAN NIXON) : (NOT (DOVE NIXON)) / (HAWK NIXON)>  
<(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>)
```

Konsequenzen von W und D₀:

```
((NIXON NIXON) (REPUBLICAN NIXON) (QUAKER NIXON) (HAWK NIXON)  
(POLITICALLY-MOTIVATED NIXON))
```

Keine Konflikte gefunden!

Eine Menge generierender Defaults wurde gefunden:

```
(<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>  
<(NIXON NIXON) : NIL / (QUAKER NIXON)>)
```

<(REPUBLICAN NIXON) : (NOT (DOVE NIXON)) / (HAWK NIXON)>
 <(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>

Da nun keine widersprüchlichen Defaults mehr gefunden wurden und die Menge D_0 auch maximal ist (in Bezug auf das Komplement, s.o.), wird diese Menge als Menge generierender Defaults gespeichert.

Die Alternative zur Entfernung des *Quaker(Nixon)*-Defaults ist, wie bereits erwähnt, den Default beizubehalten, aber dafür zu sorgen, daß $\neg(\neg\text{Hawk}(\text{Nixon})) = \text{Hawk}(\text{Nixon})$ nicht mehr abgeleitet werden kann. Daher wird der Default *Republican(Nixon)* entfernt, da seine Konklusion *Hawk(Nixon)* ist.

Alternative:

Behalte <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>

Betrachte Teilmenge:

<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
 <(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
 <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
 <(NIXON NIXON) : NIL / (QUAKER NIXON)>
 <(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>

von allen Teilmengen:

((<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
 <(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
 <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
 <(NIXON NIXON) : NIL / (QUAKER NIXON)>
 <(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>))

 Momentan betrachtete Menge von Defaults D':

<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
 <(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>
 <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
 <(NIXON NIXON) : NIL / (QUAKER NIXON)>
 <(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>

Sicheres Wissen W bzw. A:

((NIXON NIXON))

Prinzipiell anwendbare Defaults D_0 bzgl. W:

<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>
 <(NIXON NIXON) : NIL / (QUAKER NIXON)>
 <(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
 <(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>

KAPITEL 3. ALGORITHMEN ZUR BERECHNUNG VON EXTENSIONEN⁴⁷

Konsequenzen von W und D0:

```
((NIXON NIXON) (REPUBLICAN NIXON) (QUAKER NIXON) (DOVE NIXON)
(POLITICALLY-MOTIVATED NIXON))
```

Keine Konflikte gefunden!

Eine Menge generierender Defaults wurde gefunden:

```
(<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>
<(NIXON NIXON) : NIL / (QUAKER NIXON)>
<(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>)
```

Es wurde nun eine zweite Menge generierender Defaults gefunden. Anhand der beiden Mengen generierender Defaults können nun die Extensionen als deren Konsequenzen mit dem sicheren Wissen berechnet werden:

Menge generierender Defaults D1':

```
(<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>
<(NIXON NIXON) : NIL / (QUAKER NIXON)>
<(REPUBLICAN NIXON) : (NOT (DOVE NIXON)) / (HAWK NIXON)>
<(HAWK NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>)
```

Generierte Extension1:

```
((NIXON NIXON) (REPUBLICAN NIXON) (QUAKER NIXON) (HAWK NIXON)
(POLITICALLY-MOTIVATED NIXON))
```

Menge generierender Defaults D2':

```
(<(NIXON NIXON) : NIL / (REPUBLICAN NIXON)>
<(NIXON NIXON) : NIL / (QUAKER NIXON)>
<(QUAKER NIXON) : (NOT (HAWK NIXON)) / (DOVE NIXON)>
<(DOVE NIXON) : NIL / (POLITICALLY-MOTIVATED NIXON)>)
```

Generierte Extension2:

```
((NIXON NIXON) (REPUBLICAN NIXON) (QUAKER NIXON) (DOVE NIXON)
(POLITICALLY-MOTIVATED NIXON))
```

Wie wir sehen, entsprechen diese Ergebnisse unseren Erwartungen. Dieses Verfahren, wie wir später anhand einiger Tests sehen werden, ist wesentlich schneller als der definitionsbasierte Algorithmus.

Kapitel 4

Default-Schließen mit Prioritäten

Wir haben schon den Umgang mit Defaults in den vorherigen Kapiteln kennengelernt. Oftmals ist mehr als eine Default-Regel anwendbar. Manchmal gibt es Default-Regeln, die zusammen nicht anwendbar sind (weil sie sich widersprechen), sodaß mehrere Extensionen entstehen. Befaßt man sich etwas eingehender mit diesen Extensionen, so sind oftmals einige Extensionen „plausibler“ als andere, weil diese z.B. genauere Informationen beinhalten. Eine Möglichkeit plausiblere Extensionen zu erhalten wäre, die Defaults in einer bestimmten Reihenfolge anzuwenden, z.B. könnte man jeweils den „speziellsten“ Default nehmen. Der speziellste Default könnte z.B. derjenige mit der speziellsten Vorbedingung sein. Dies kann interessante Auswirkungen auf die Menge der Extensionen haben. Dieser Sachverhalt wird nun in diesem Kapitel näher erläutert.

4.1 Priorisierte Default-Theorien

Betrachten wir zunächst einmal ein Beispiel. Wir nehmen an, daß wir einen Default haben, welcher sagt, daß Priester nicht lügen können, und einen anderen, der sagt, daß Menschen lügen können. Die Klassifikation zeige uns, daß das Konzept Priester ein Unterkonzept des Konzeptes Mensch sei. Intuitiv wissen wir, daß für irgendeinen Priester der erste Default feuern müßte, womit wir dann nur eine Extension hätten, in der Priester nicht lügen können. Doch da wir bisher keine Prioritäten in Reiters Default-Logik haben,

bekommen wir auch eine zweite Extension, in der Priester lügen können.¹

Wenn Konflikte mit Defaults auftauchen, so liegt es nahe, daß die spezifischste Information bevorzugt werden sollte. Im Kontext terminologischer Default-Theorien bedeutet dies, daß für eine Instanz des Konzepte D , welches vom Konzept C subsumiert wird, ein Default mit der Vorbedingung D gegenüber einem Default mit der Vorbedingung C bevorzugt werden sollte. Bisher war dies nicht der Fall. Wenn wir annehmen, daß P , M und S Konzeptbeschreibungen sind, welche Priester, Menschen und schwache Menschen definieren, und P durch M subsumiert wird, dann hat unsere terminologische Default-Theorie (W, D) , die aus der Weltbeschreibung $\{M(Peter), P(Peter)\}$ und den Defaults $P(x) : \neg S(x)/\neg S(x)$ und $M(x) : S(x)/S(x)$ besteht, zwei *R-Extensionen* (R für Reiters Default-Logik), also die Extensionen $\{M(Peter), P(Peter), S(Peter)\}$, $\{M(Peter), P(Peter), \neg S(Peter)\}$. Die Semantik von Reiter gibt uns jedoch keinen Grund, die zweite Extension der ersten vorzuziehen, in welcher der spezifischste Default angewendet worden ist.

Um solche Probleme zu überwinden, gibt es in der Literatur zahlreiche Vorschläge; u.a. den Vorschlag von Baader und Hollunder (s. [3]). Dieser Vorschlag, den wir im folgenden kennenlernen, beachtet die Subsumptionsrelation zwischen den Vorbedingungen von Defaults. Demnach ist eine *Default-Theorie mit Spezifität* ein Tripel $(W, D, <)$, welches aus einer geschlossenen Default-Theorie (W, D) und einer strikten partiellen Ordnung $<$ auf D besteht.

Im terminologischen Fall ist W eine ABox und D wird durch die Instantiierung der terminologischen Default-Regeln mit allen Konstanten bzw. Individuen, die in der ABox auftauchen, erhalten. Für zwei instantiierte terminologische Default-Regeln d_1, d_2 mit den Vorbedingungen $C_1(a_1)$ und $C_2(a_2)$ haben wir $d_1 < d_2$ genau dann, wenn sie das gleiche Individuum betreffen ($a_1 = a_2$) und C_1 spezifischer als C_2 ist (C_2 subsumiert C_1).

Die Definition der Extensionen einer Default-Theorie mit Spezifität ist der iterativen Charakterisierung von Reiters Extensionen nachempfunden. Die Hauptidee für die Behandlung der Prioritäten ist, daß die Konsequenz eines Defaults nur dann in einem neuen Iterationsschritt hinzugefügt wird, wenn dieser nicht durch einen bevorzugten Default *blockiert* wird; d.h., es existieren keine kleineren (also präferierten) Defaults, die im Moment ebenfalls *aktiv* sind.

¹Dies ist auch ein gutes Beispiel dafür, daß wir alleine durch zwei Defaults Extensionen erhalten, die zusammengenommen bereits inkonsistent sind.

Für eine Menge E geschlossener Formeln und einen Default $d = \alpha : \beta/\gamma$ sagen wir, daß d in E *aktiv* ist, genau dann, wenn seine Vorbedingung eine Folgerung von E ist (d.h. $\alpha \in Th(E)$), seine Annahmen konsistent mit E sind (d.h. $\neg\beta \notin Th(E)$) und seine Konsequenz nicht aus E folgt (d.h. $\gamma \notin Th(E)$).

Definition 4.1 (S-Extensionen) Sei $(W, D, <)$ eine Default-Theorie mit Spezifität, und \mathcal{E} sei eine Menge von geschlossenen Formeln. Wir definieren $E_0 = W$ und für alle $i \geq 0$

$$E_{i+1} := E_i \cup \{ \gamma \mid \exists d \in D : d = \alpha : \beta/\gamma, \alpha \in Th(E_i), \\ \neg\beta \notin \mathcal{E}, \text{ und alle } d' < d \text{ sind nicht aktiv in } E_i \}$$

dann ist \mathcal{E} eine S-Extension genau dann, wenn $\mathcal{E} = \bigcup_{i \geq 0} Th(E_i)$

Mit dieser Definition der Extensionen bekommen wir das intuitiv korrekte Resultat in dem folgenden Beispiel.

Als Beispiel nehmen wir an, daß wir Konzeptbeschreibungen für Priester und Menschen sowie Subjekte, die schwach sind (und daher lügen) haben, und daß die einzige Subsumptionsbeziehung zwischen Priestern und Menschen besteht. Außerdem soll unsere terminologische Default-Theorie aus der Weltbeschreibung $P(Peter)$ und den Defaults

$$\begin{aligned} \text{Priester}(x) &: \neg\text{lügt}(x)/\neg\text{lügt}(x) \\ \text{Mensch}(x) &: \text{schwach}(x)/\text{schwach}(x) \\ \text{schwach}(x) &: \text{lügt}(x)/\text{lügt}(x) \end{aligned}$$

bestehen. Die bevorzugte Extension sollte diejenige sein, in der Peter schwach ist, aber nicht lügt. Tatsächlich kann der zweite Default für irgendeinen Priester nur dann feuern, wenn der spezifischere Default (die Behauptung, daß Priester normalerweise nicht lügen) schon angewendet worden ist. Dies bedeutet, daß der dritte Default (die Behauptung, daß Objekte mit Schwäche normalerweise lügen) nie auf einen Priester angewendet werden kann. Somit ist von den zwei existierenden R-Extensionen nur eine dieser auch S-Extension, nämlich die, die die Prioritäten respektiert.

Theorem 4.1 Sei \mathcal{E} eine S-Extension einer Default-Theorie mit Spezifität $(W, D, <)$. Dann ist \mathcal{E} eine R-Extension von (W, D) .

Dieses Theorem besagt, daß die Menge der S-Extensionen immer eine Teilmenge der Menge der R-Extensionen ist. Der Beweis ist in [3] zu finden.

Wenn eine Default-Theorie mit Spezifität keine R-Extensionen hat, dann existieren natürlich auch keine S-Extensionen. Aber auch wenn wir R-Extensionen haben, muß dies nicht heißen, daß S-Extensionen für eine Default-Theorie mit Spezifität existieren.

Angenommen die Wissensbasis W sei leer, und wir führen drei Defaults ein $\{ : \beta / \beta, : \neg \beta / \neg \beta, \beta : \alpha / \neg \alpha \}$.

Wir nehmen an, daß der erste Default kleiner ist als der zweite, und daß eine weitere Priorisierung nicht besteht. Diese Default-Theorie hat die R-Extension $Th(\{\neg \beta\})$, aber keine S-Extension. Tatsächlich würde eine S-Extension den ersten Default bevorzugen, welcher zu β führen würde, und damit würde der dritte Default anwendbar werden. Der dritte Default kann jedoch niemals angewendet werden, da er in sich widersprüchlich ist $(\alpha, \neg \alpha)$.

Wie auch im nicht-priorisierten Fall, so weisen *normale* Default-Theorien mit Spezifität schönere Eigenschaften als willkürliche Default-Theorien auf. Analog zu den normalen Default-Theorien bzgl. R-Extensionen gilt, daß alle geschlossenen normalen Default-Theorien mit Spezifität eine S-Extension haben (s. [3]).

4.2 Die Berechnung von S-Extensionen

Da alle S-Extensionen auch R-Extensionen sind, könnte man zuerst die R-Extensionen der Default-Theorie berechnen und dann mit Hilfe der Definition der S-Extensionen alle S-Extensionen herausfiltern. Wie auch immer, es existieren ggf. mehr R-Extensionen als S-Extensionen, was bedeutet, daß dieser Weg nicht kostengünstig ist. Analog zum definitionsbasierten Algorithmus für die R-Extensionen kann auch ein definitionsbasierter Algorithmus für die S-Extensionen implementiert werden, ebenfalls nach dem „generate & test“-Verfahren, wofür aber wieder alle Teilmengen von Defaults betrachtet werden müssen.

Daher wird in [3] ein anderer Algorithmus vorgeschlagen, der die S-Extensionen ebenfalls direkt berechnet. Die Idee ist, gemäß der Definition der S-Extensionen zu iterieren, ohne daß bereits - sowohl wie in der Definition der R-Extensionen als auch in der Definition der S-Extensionen - die ja noch zu findende Extension bereits vorliegen muß. Das Problem ist, herauszufinden,

welche möglichen Mengen von Konsequenzen die „Kandidaten“ sind, die im nächsten Schritt der Iteration addiert werden können. Natürlich kann es dort mehr als eine korrekte Wahl geben, da es möglich ist, daß mehr als eine S-Extension existiert.

Im folgenden (nichtdeterministischen) Algorithmus wird E_i immer eine Teilmenge von $W \cup \text{Con}(D)$ und J_i eine Teilmenge von $\neg \text{Jus}(D)$ sein (für eine Menge F von Formeln sei $\neg F := \{ \neg \beta \mid \beta \in F \}$).

Die Idee hinter den Mengen J_i ist wie folgt: Wenn die Konsequenz eines minimalen aktiven Defaults (minimal heißt, es gibt keinen kleineren ebenfalls aktiven Default) nicht in E_{i+1} vorhanden ist, dann ist der Grund dafür, daß seine Annahmen nicht konsistent mit der endgültigen Extension sind. Folglich, wenn wir solch einen Default aus \hat{D}_{i+1} herausnehmen, wissen wir, daß seine negierte Annahme zu der Extension gehören muß. Die Bedingung für \hat{D}_{i+1} korrespondiert zu dem Faktum, daß Defaults, deren Konsequenzen zu S-Extensionen addiert werden, Annahmen besitzen müssen, die konsistent mit der Extension sind. Diese Bedingung kann lediglich lokale Korrektheit bzgl. des aktuellen Iterationsschrittes erzwingen, jedoch keine globale Korrektheit. Haben wir einen Kandidaten für eine S-Extension, so müssen für diesen am Ende noch zwei weitere Bedingungen, die weiter unten stehen, getestet werden, um sicher zu sein, daß es sich wirklich um eine S-Extension handelt.

Algorithmus: Sei $(W, D, <)$ eine geschlossene Default-Theorie mit Spezifität. Wenn W inkonsistent ist, dann ist $\text{Th}(W)$ die einzige S-Extension. Sonst definieren wir $E_0 := W$ und $J_0 = \emptyset$. Nun nehmen wir an, daß E_i ($i \geq 0$) schon definiert ist.

Wir führen

$$D_{i+1} := \{d \in D \mid d \text{ ist aktiv in } E_i \text{ und kein } d' < d \text{ ist aktiv in } E_i\}$$

ein und wählen eine nichtleere Teilmenge \hat{D}_{i+1} von D_{i+1} , welche

$$\neg \beta \notin \text{Th}(E_i \cup \text{Con}(\hat{D}_{i+1}) \cup J_i \cup \neg \text{Jus}(D_{i+1} \setminus \hat{D}_{i+1}))$$

für alle $\beta \in \text{Jus}(\hat{D}_{i+1})$ erfüllt.

Wenn es keine solche Menge gibt, dann ist $E_{i+1} := E_i$, $J_{i+1} := J_i$.

Sonst wird jede Wahl zu neuen Mengen

$$E_{i+1} := E_i \cup \text{Con}(\hat{D}_{i+1}) \text{ und } J_{i+1} := J_i \cup \neg \text{Jus}(D_{i+1} \setminus \hat{D}_{i+1})$$

führen.

Die Menge $\mathcal{E} := \bigcup_{i \geq 0} Th(E_i)$ ist eine S-Extension genau dann, wenn:

1. für alle $d = \alpha : \beta/\gamma \in \bigcup_{i \geq 1} \hat{D}_i$ wir $\neg\beta \notin \mathcal{E}$ haben, und
2. für alle $\neg\beta \in \bigcup_{i \geq 1} J_i$ wir $\neg\beta \in \mathcal{E}$ haben.

Nun wollen wir anhand eines Protokolles einer Implementierung dieses Algorithmus' die Vorgänge in diesem Algorithmus anhand unseres Beispiels etwas verdeutlichen.

Alle Defaults:

```
<<(PRIESTER PETER) : (NOT (LUEGT PETER)) / (NOT (LUEGT PETER))>>
<(MENSCH PETER) : (SCHWACH PETER) / (SCHWACH PETER)>
<(SCHWACH PETER) : (LUEGT PETER) / (LUEGT PETER)>
```

Der Default mit der Vorbedingung (PRIESTER PETER) hat eine höhere Priorität als der Default mit der Vorbedingung (MENSCH PETER).

Sicheres Wissen W:

```
((PRIESTER PETER) (MENSCH PETER))
```

R-Extensionen:

1. ((LUEGT PETER) (SCHWACH PETER) (PRIESTER PETER) (MENSCH PETER))
2. ((SCHWACH PETER) (NOT (LUEGT PETER)) (PRIESTER PETER) (MENSCH PETER))

E0:

```
((PRIESTER PETER) (MENSCH PETER))
```

J0: NIL

aktive Defaults:

```
<<(PRIESTER PETER) : (NOT (LUEGT PETER)) / (NOT (LUEGT PETER))>>
<(MENSCH PETER) : (SCHWACH PETER) / (SCHWACH PETER)>
```

D1 (spezifischste aktive Defaults):

$\langle\langle \text{PRIESTER PETER} \rangle : (\text{NOT} (\text{LUEGT PETER})) / (\text{NOT} (\text{LUEGT PETER})) \rangle\rangle$

D^1 (Teilmenge von D_1 , die obige Bedingung erfuehlt):

$\langle\langle \text{PRIESTER PETER} \rangle : (\text{NOT} (\text{LUEGT PETER})) / (\text{NOT} (\text{LUEGT PETER})) \rangle\rangle$

Hier finden wir eine Teilmenge \hat{D}_1 , die die Bedingung $\neg\beta \notin Th(E_0 \cup Con(\hat{D}_1) \cup J_0 \cup \neg Jus(D_1 \setminus \hat{D}_1))$ für alle $\beta \in Jus(\hat{D}_1)$ erfuehlt.

Daher wird der Algorithmus nun rekursiv mit $E_1 := E_0 \cup Con(\hat{D}_1)$ und $J_1 := J_0 \cup \neg Jus(D_1 \setminus \hat{D}_1)$ aufgerufen.

E1:

$((\text{NOT} (\text{LUEGT PETER})) (\text{PRIESTER PETER}) (\text{MENSCH PETER}))$

J1: NIL

aktive Defaults:

$\langle\langle \text{MENSCH PETER} \rangle : (\text{SCHWACH PETER}) / (\text{SCHWACH PETER}) \rangle\rangle$

D2 (spezifischste aktive Defaults):

$\langle\langle \text{MENSCH PETER} \rangle : (\text{SCHWACH PETER}) / (\text{SCHWACH PETER}) \rangle\rangle$

D^2 (Teilmenge von D2, die obige Bedingung erfuehlt):

$\langle\langle \text{MENSCH PETER} \rangle : (\text{SCHWACH PETER}) / (\text{SCHWACH PETER}) \rangle\rangle$

Hier wird der Algorithmus erneut aufgerufen.

E2:

$((\text{SCHWACH PETER}) (\text{NOT} (\text{LUEGT PETER})) (\text{PRIESTER PETER}) (\text{MENSCH PETER}))$

J2: NIL

aktive Defaults:

NIL

D3: NIL

S-Extension gefunden!

$((\text{SCHWACH PETER}) (\text{NOT} (\text{LUEGT PETER})) (\text{PRIESTER PETER}) (\text{MENSCH PETER}))$

Zum Schluß werden die beiden Bedingungen auf der gefundenen Menge getestet. Da dieser Test positiv ausfällt, ist diese Menge eine S-Extension.

Wir haben im letzten Kapitel einige Algorithmen betrachtet und ihre Arbeitsweise verstanden. Es stellte sich heraus, daß das Verfahren von Baader und Hollunder eine effizientere Möglichkeit zur Berechnung von R- bzw. S-Extensionen darstellt, als das definitionsbasierte Verfahren. Die Notwendigkeit, Defaults zu priorisieren, wurde uns in diesem Kapitel auch klar.

Es bleibt nun die Frage, in wieweit wir diese Algorithmen zum räumlichen Schließen verwenden können, und welche Punkte in diesem Zusammenhang anders behandelt werden müssen.

Kapitel 5

Räumlich-terminologisches Schließen mit Defaults

In diesem Kapitel will ich zunächst anhand einiger Beispiele die Signifikanz der Defaults gerade beim räumlichen Schließen verdeutlichen. Wir haben bereits erwähnt, daß i.d.R. immer unvollständige Weltbeschreibungen vorliegen. Handelt es sich um eine Weltbeschreibung räumlicher Gegebenheiten, so kann eine $\mathcal{ALCRP}(\mathcal{S}_2)$ ABox zusammen mit einer entspr. TBox, die das räumlich-konzeptuelle Hintergrundwissen darstellt, u.U. eine adäquate Repräsentation dieser Gegebenheiten sein. Wollen wir z.B. im Rahmen einer geographischen Wissensbasis (s.u.) Wissen über geographische Entitäten repräsentieren, so könnte eine sinnvolle Forderung für das Konzept „Land“ sein, daß je zwei Länder sich ausschließlich berühren dürfen oder disjunkt sein müssen.

Mit Hilfe der ABox-Realisation können wir für jedes ABox-Individuum alle seine Konzeptzugehörigkeiten berechnen, sofern alle hinreichenden Informationen (bzw. Bedingungen) für das entspr. Individuum und definierte Konzept vorhanden sind. Viele Konzepte sind jedoch nur primitiv definiert, d.h., nur die notwendigen Bedingungen liegen vor, sodaß ABox-Realisation nicht weiter hilft. In solchen Fällen kann die Verwendung von Default-Regeln hilfreich sein.

Doch welche Art von Kriterien können wir für unsere räumlich-terminologischen Default-Regeln ausnutzen? Zum einen können bereits bekannte Relationen zwischen Objekten verwendet werden, um Konzeptzugehörigkeiten (oder auch andere Relationen) zu hypothetisieren. Ist ein Auto in einer Werkstatt, so muß es sich um ein Auto handeln, das entweder eine Repa-

ratur benötigt, oder inspiziert werden muß. Zum anderen können bereits bekannte Konzeptzugehörigkeiten verwendet werden, um plausible Relationen zu anderen Objekten hinzuzufügen. Ist das Wetter heute regnerisch, so ist es plausible anzunehmen, daß das Auto in der Garage ist.

Wenn wir räumliche Beziehungen mit Defaults hinzufügen wollen, müssen wir in unseren Defaults ABox-Ausdrücke verwenden, sodaß wir nicht nur wie bisher Konzepte, sondern auch Relationen zwischen zwei Objekten in α , β und γ behandeln können. Ein geschlossener Default könnte nun so aussehen:

$$D = \{a : \text{stadt}, b : \text{Land}\} : \{(b, a) : \text{enthaelt}\} / \{(b, a) : \text{enthaelt}\}$$

Dieser könnte aus dem offenen Default $D = \{X : \text{stadt}, Y : \text{Land}\} : \{(Y, X) : \text{enthaelt}\} / \{(Y, X) : \text{enthaelt}\}$ durch das Instantiieren der freien Variablen X und Y mit den Individuen a und b entstanden sein. Dies bedeutet, daß wir nicht wie bisher mit einer freien Variable auskommen können, und mindestens zwei benötigen werden.

In diesem Kapitel wird nun auf die nötigen Veränderungen eingegangen. Dabei handelt es sich sowohl um syntaktische als auch um theoretische Veränderungen.

5.1 Räumlich-terminologische Default-Theorien

An dieser Stelle betrachten wir ein Beispiel, das uns die oben erwähnte Problematik der Syntax sowie der Semantik näher bringt.

Hier wird nun eine Wissensbasis über eine Landkarte aufgebaut. Anhand dieser sollen bestimmte Inferenzen gezogen werden. Die Wissensbasis soll Konzepte wie Land, Stadt, See u.s.w. beinhalten. Diese Wissensbasis wird auch in späteren Abschnitten verwendet.

Die Rollen und Konzepte unserer Wissensbasis sollen wie folgt (aus [9]) definiert sein:

$$\begin{aligned} \text{disjunkt} &\doteq \exists(\text{hat_gebiet})(\text{hat_gebiet}).dc \\ \text{ueberlappt} &\doteq \exists(\text{hat_gebiet})(\text{hat_gebiet}).po \\ \text{in} &\doteq \exists(\text{hat_gebiet})(\text{hat_gebiet}).tpp - ntp \\ \text{enthaelt} &\doteq \exists(\text{hat_gebiet})(\text{hat_gebiet}).tpi - ntpi \\ \text{beruehrt} &\doteq \exists(\text{hat_gebiet})(\text{hat_gebiet}).ec \end{aligned}$$

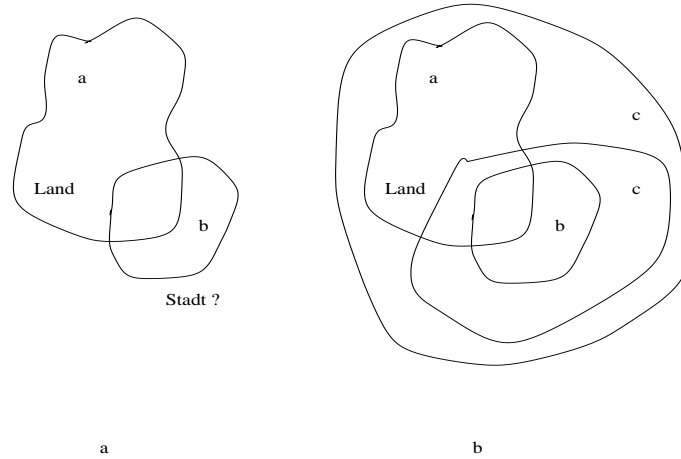


Abbildung 5.1: Was ist b? Kann b eine Stadt sein?

$$\begin{aligned}
 \text{Gebiet} &\doteq \exists \text{hat_gebiet.ist_Region} \\
 \text{Land_Gebiet} &\sqsubseteq \text{administratives_Gebiet} \sqcap \text{Gebiet} \sqcap \text{Gross} \\
 \text{Stadt_Gebiet} &\sqsubseteq \text{administratives_Gebiet} \sqcap \text{Gebiet} \sqcap \neg \text{Gross} \\
 \text{natuerliches_Gebiet} &\doteq \neg \text{administratives_Gebiet} \\
 \text{Land} &\doteq \text{Gebiet} \sqcap \forall \text{enthaelt.} \neg \text{Land_Gebiet} \sqcap \\
 &\quad \forall \text{in.} \neg \text{Land_Gebiet} \sqcap \forall \text{ueberlappt.} \neg \text{Land_Gebiet} \\
 \text{Stadt} &\doteq \text{Stadt_Gebiet} \sqcap \exists \text{in.Land_Gebiet} \\
 \text{See} &\sqsubseteq \text{natuerliches_Gebiet} \sqcap \text{Gebiet}
 \end{aligned}$$

Zunächst werden die komplexen Rollen *disjunkt*, *ueberlappt*, *in*, *enthaelt* und *beruehrt* definiert, welche von ihren Bedeutungen her klar sein dürften. Dann wird ein *Land* als ein Gebiet, das selbst kein *Land_Gebiet* enthält, sich nicht mit einem *Land_Gebiet* überlappt und sich nicht in einem *Land_Gebiet* befindet, definiert. Diese bestimmte Beziehungen können also nicht zwischen *Land* und *Land_Gebiet* existieren. Mit *Land_Gebiet* ist eigentlich *Land* gemeint. Damit wird sicher gestellt, daß diese Beziehungen nicht zwischen zwei Ländern gelten dürfen. Der Grund, warum *Land_Gebiet* verwendet wird, ist daß wir keine zyklische Definitionen behandeln können. Eine *Stadt* wird als ein *Stadt_Gebiet*, die sich in einem *Land_Gebiet* befindet, definiert. Hier wird das primitive Konzept *Stadt_Gebiet* auch wegen der Zyklenfreiheit verwendet. Wieder ist hier mit *Land_Gebiet* *Land* gemeint. Bei der Definition von *Stadt* dürfen wir kein *Ein.Land* verwenden, da sonst das Kriterium der Zulässigkeit verletzt wird. Die Definition von *Land* beinhaltet Ausdrücke wie $\forall \dots$, sodaß nach dem expandieren $\exists \text{in.} \forall \dots$ entstehen würde. Ein *See* wird als ein *natuerliches_Gebiet* (kein administratives Gebiet) definiert.

Die Abbildung 5.1(a) soll einen Ausschnitt einer Karte darstellen. Über die

zwei Objekte soll folgendes in der ABox assertiert sein:

$$\{a : Land, b : Gebiet, (a, b) : ueberlappt\}$$

Diese ABox kann als Datenbestand eines GIS angesehen werden. Für die Objekte könnte zuerst angenommen werden, daß sie *Gebiete* sind. Weiter kann angenommen werden, daß das Objekt *a*, wegen der schon vorhandenen Information, oder durch Realisation, als ein *Land* erkannt wird. Doch was kann nun *b* sein?

Sicher kann die Frage, ob *b* eine Stadt sei, durch ABox-Konsistenz beantwortet werden. Dabei fügt man $\{b : Stadt\}$ zu der ABox hinzu, und wenn der ABox weiterhin konsistent bleibt, dann könnte *b* eine Stadt sein. Als Ergebnis erhalten wir, daß *b* keine Stadt ist. Denn eine Stadt muß in einem Land enthalten sein, und wenn *b* in einem Land enthalten ist, dann gibt es einen „Clash“ (eine Inkonsistenz taucht auf), da sich zwei Länder nicht überlappen oder enthalten dürfen (siehe 5.1 b). Durch ABox-Konsistenz kann also die Konsistenz der Datenbestände sichergestellt werden.

Um zu bestimmen, was *b* nun wirklich ist, müssten wir für *b* alle möglichen Konzepte unserer TBox austesten. Dieser Weg scheint ineffizient zu sein. Ausserdem würde die Begründung fehlen. Gerade bei solchen Systemen wie GIS kann es von Vorteil sein, eine Möglichkeit in der Hand zu haben, mit dem man den Datenbestand ergänzen kann. Das System sollte Hypothesen bilden, was *b* sein könnte, und im Dialog mit dem Benutzer den Datenbestand ergänzen. Dabei sollen die Hypothesen „durchdacht“ und nicht willkürlich sein. Dabei bietet sich Defaults miteinzubeziehen, und die Frage, was *b* sein kann, zu beantworten.

Nehmen wir an, daß wir folgende Defaults haben, wobei wir in α, β und γ Konzeptterme (wie aus den terminologischen Default-Theorien bekannt) verwenden.

1. $D_1 = Gebiet : Land / Land$
2. $D_2 = Gebiet : Stadt / Stadt$
3. $D_3 = Gebiet : See / See$

Nach dem Instantiieren der freien Variable *x* (das Konzept *Stadt* ist in Prädikatenlogischer Form $Stadt(x)$) mit den Individuen *a* und *b* („Closing“) erhalten wir 6 verschiedene geschlossene Defaults

$(D_1(a), D_2(a), D_3(a), D_1(b), D_2(b), D_3(b))$.

Die Defaults $D_2(a)$ und $D_3(a)$ können nicht feuern, da die Annahmen β_2 und β_3 von D_2 und D_3 mit der ABox nicht konsistent sind, b.z.w. ein *Gebiet* kann nicht zugleich eine *Stadt* oder ein *See* sein, da diese Konzepte disjunkt sind. $D_1(b)$ kann nicht feuern, da sich a und b in der ABox überlappen, und diese Beziehung ist zwischen zwei Ländern nicht erlaubt. $D_2(b)$ kann auch nicht feuern, da eine *Stadt* vollkommen in einem *Land* enthalten sein muß. Sicher kann b eine Stadt eines anderen Landes (z.B. c) sein, doch das würde heißen, daß c sich mit a entweder überlappt, oder daß zwischen a und c eine andere Beziehung, wie *enthalten* gilt, die es wieder nicht zwischen zwei Ländern geben kann (siehe 5.1 b). Folglich können nur die Defaults $D_1(a)$ und $D_3(b)$ feuern. Wir erhalten eine Extension:

- $E = \{a : Land, b : Gebiet, (a, b) : ueberlappt, b : See\}$

Anhand dieser Extension kann angenommen werden, daß b ein *See* ist.

In den obigen Defaults haben wir nur Konzeptterme in α , β und γ geführt. Manchmal wollen wir vielleicht räumliche Beziehungen zwischen zwei Objekten (durch Defaults) assertieren, bzw. annehmen. Solche Konklusionen können aber mit den terminologischen Default-Theorien, wie in [2] besprochen, nicht ausgedrückt werden, da wir dort auf Konzepte in den α , β_i und γ beschränkt sind. Daher erweitern wir unsere Default-Regeln, und lassen in α , β_i und γ sog. ABox-Muster zu. Diese ABox-Muster können ABoxen sein, die allerdings auch freie Variablen enthalten können. Beim Instantiieren dieser Variablen durch alle Individuen, die in ABox vorkommen, erhalten wir geschlossene Defaults, deren α , β_i und γ $\mathcal{ALCRP}(\mathcal{S}_2)$ ABoxen sind.

Ein Default mit ABox-Muster und zwei freien Variablen kann nun so aussehen:

- $\{(x, y) : inside, x : Stadt\} : \{y : Land\} / \{y : Land\}$

Angenommen es gibt in unserer ABox 2 Individuen, dann erhalten wir nach dem Instantiieren (Closing) vier Defaults. Hier sehen wir schon, daß Kombinatorik ins Spiel kommt, und daher die Anzahl der Defaults und die Zeit zum Schließen exponentiell wächst.

Es können auch beliebig viele Variablen vorkommen. Natürlich ist es einem selbst überlassen, wieviele Variablen man verwendet, oder verwenden muß,

um den erwünschten Aussagekraft ¹ zu erreichen. So kann ein Default mit 3 freien Variablen wie folgt aussehen:

- $\{(x, y) : touches, (z, x) : touches\} : \{(z, y) : touches\} / \{(z, y) : touches\}$

Sicher bedarf das Instantiieren hier mehr Zeit, als mit 2 freie Variablen, wenn wir alle möglichen Kombinationen der Individuen statt der freien Variablen einsetzen wollen. Dies bedeutet, daß durch das Instantiieren, die Anzahl der Defaults zunimmt, doch wie wir im nächsten Kapitel sehen werden, kann durch Filterung die Anzahl der Defaults auf einige, für uns relevanten reduziert werden.

Desweiteren, wie schon oben in einigen Stellen (in einigen Defaults schon verwendet), ist es von Vorteil nicht nur von einem Axiom in α , β und γ auszugehen, sondern von ABoxen. Ein Default mit mehreren Axiomen in α , β oder γ kann so aussehen.:

- $\{x : hamburg, y : Land, (x, y) : enthealt\} : \{y : deutschland\} / \{y : deutschland\}$

So ist die Vorbedingung der obigen Defaults erst dann erfüllt, wenn alle Axiome in α (konjunktiv verknüpft) erfüllt sind. Für die Handhabung der Extensionen mit ABoxen in α , β und γ müssen einige Erweiterungen (aus [9]) in der Theorie vorgenommen werden.

5.1.1 Erweiterung und Anpassung

Für die obigen „syntaktischen“ Veränderungen muß einiges in der Theorie gezeigt bzw. angepasst werden. In diesem Abschnitt erweitern wir die terminologische Default-Theorien zu sog. *räumlich-terminologischer Default-Theorie* (aus [9]).

Definition 5.1 *Eine räumlich-terminologische Default-Regel hat die Form $d = \alpha : \beta_1, \dots, \beta_n / \gamma$ wobei α, β_i und γ konsistente und eingeschränkte $\mathcal{ALCRP}(\mathcal{S}_2)$ -ABoxen sind, die unter anderem auch Prädikats-basierte Rollen-Axiome der Form $(x, y) : \exists(u)(v).P$ enthalten dürfen, wobei P ein S_2 -Prädikat der Stelligkeit 2 ist. Eine räumlich-terminologische Default-Theorie ist ein Tupel (A, D) , wobei D eine Menge von räumlich-terminologischen Default-Regeln und A eine eingeschränkte $\mathcal{ALCRP}(\mathcal{S}_2)$ -ABox ist.*

¹gemeint ist das, was wir mit Defaults assertieren wollen.

Hier werden nur solche Konzeptterme zugelassen, die die Einschränkungskriterien (s.o.) von $\mathcal{ALCRP}(\mathcal{S}_2)$ erfüllen. Weiter wird von den Axiomen in den ABoxen verlangt, daß sie expandiert sind. Unsere Default-Theorie ist weiterhin (wie in [2]) ein Paar (A, D) , wobei A eine ABox und D ein endlicher Satz von terminologischen Default-Regeln ist. Diese Regeln werden durch das Instantiieren der freien Variablen durch alle Individuen der ABox geschlossen. Hier werden wir die Defaults nicht auf solche Individuen anwenden, die implizit durch einen Existenz-Restriktion (\exists) entstehen. Das bedeutet, daß das Skolemisieren der Defaults, die für die Behandlung der offene Defaults nötig war (Vorschlag von Reiter in [2]) wegfällt.

Nun führen wir uns nochmal die Definition 5.2 vor Augen.

Definition 5.2 (Reiters Extension) *Sei (A, D) eine geschlossene Default-Theorie, und sei A eine Menge von geschlossenen Formeln. Dann definieren wir $E_0 := A$ und für alle $i \geq 0$*

$$E_{i+1} := E_i \cup \{ \gamma \mid \alpha : \beta_1, \dots, \beta_n / \gamma \in D, \alpha \in Th(E_i), \text{ und } \neg\beta_1, \dots, \neg\beta_n \notin Th(E) \}.$$

Dann ist $Th(E)$ eine Extension von (A, D) genau dann, wenn

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i).$$

Extensionen entstehen ja dadurch, daß die Konsequenzen der Defaults, zu dem bisherigen Wissen dazuaddiert werden ($A \cup \gamma \mid \alpha : \beta_1 \dots \beta_n / \gamma \in D$) sofern sie mit dem sicheren Wissen A nicht im Konflikt stehen. Die iterative Konstruktionsvorschrift endet, wenn der „Fixpunkt“ erreicht ist. In der Absicht räumliche Beziehungen zwischen Domänenobjekten schließen zu können, übernehmen wir diese Definition der Extension, und lassen in den α, β_i und γ ABoxen zu. Dabei muß nun sichergestellt werden, daß das Folgerbarkeitsproblem weiterhin entscheidbar bleibt.

Aus der Definition 2.5 können wir entnehmen, daß $\gamma \in D$ erst dann dazu addiert wird, wenn zunächst $\alpha \in Th(E_i)$ ($E_i \models \alpha$) ist, und wenn $\neg\beta_1, \dots, \neg\beta_n \notin Th(E)$ ($E_i \not\models \neg\beta$) ist. Um zu überprüfen, ob $\alpha \in Th(E_i)$ ist, muß nur gezeigt werden, daß aus einer ABox E_i α folgt ($E_i \models \alpha$). Da α eine ABox ist, die aus einem oder mehreren Axiomen bestehen kann, müssen einige Fälle berücksichtigt werden.

Ein $\mathcal{ALCCRP}(\mathcal{S}_2)$ ABox-Axiom δ folgt aus einem $\mathcal{ALCCRP}(\mathcal{S}_2)$ ABox A , genau dann wenn folgendes gilt:

$$A \models \delta, \quad \text{iff} \quad \left\{ \begin{array}{l} \delta = a : C \longrightarrow \neg SAT(A \cup \{a : \neg C\}) \\ \delta = (a, b) : R \longrightarrow \neg SAT(A \cup \{a : \forall R.X_{new}, b : \neg X_{new}\}) \\ \delta = (a, b) : f \longrightarrow \\ \quad \neg SAT(A \cup \{a : \forall f.X_{new} \sqcup \exists(f).is_region, b : \neg X_{new}\}) \\ \delta = (a, x) : f \longrightarrow \\ \quad \neg SAT(A \cup \{a : \exists(f).\Psi \sqcup \exists f.\top \sqcup \forall f.\perp, x : \overline{\Psi}\}) \\ \delta = (x_1, x_2) : P \longrightarrow \\ \quad \neg SAT(A \cup \{(x_1, x_2) : \overline{P}\}) \\ \delta = (a, b) : \exists(u)(v).P \longrightarrow \\ \quad \neg SAT(A \cup \{(a, b) : \exists(u)(v).\overline{P}\}) \wedge \\ \quad \neg SAT(A \cup \{a : \forall u.\top\}) \wedge \\ \quad \neg SAT(A \cup \{b : \forall v.\top\}), \\ \quad \text{wobei } u = v = has_area. \end{array} \right.$$

Hier steht X_{new} für einen atomaren Konzeptterm, der nicht in der ABox auftaucht, und nirgends sonst verwendet wird, der aber als ein „Marker-Konzept“ benutzt wird. Analog dazu ist Ψ (wie auch $\overline{\Psi}$) ein neues, sonstwo nicht verwendetes „Marker-Prädikat“ der konkreten Domäne. Diese zwei Prädikate haben die Eigenschaft, daß sie nicht mit den anderen Prädikaten P_i der konkreten Domäne interagieren. Daher sind die beiden Konjunktionen der Prädikaten aus der konkreten Domänen $\bigwedge_{i=1}^k P_i \wedge \Psi$ bzw. $\bigwedge_{i=1}^k P_i \wedge \overline{\Psi}$ auch erfüllbar, wenn $\bigwedge_{i=1}^k P_i$ erfüllbar ist. Unabhängig von der Erfüllbarkeit von $\bigwedge_{i=1}^k P_i$ ist $\bigwedge_{i=1}^k P_i \wedge \Psi \wedge \overline{\Psi}$ niemals erfüllbar. Desweiteren ist R eine primitive Rolle, und f ist ein Attribut. Mit $SAT(A)$ testen wir, ob eine ABox A erfüllbar ist. Weiter sollen a und b als Objekte der abstrakten Domäne interpretiert werden, wogegen x, x_1, x_2 als Objekte der konkreten Domäne interpretiert werden. Die konkrete Domäne \mathcal{S}_2 und die abstrakte Domäne sind disjunkt.

Im ersten Fall ($\delta = a : C$) haben wir ein Instanzerkennungs-Problem, welches entscheidbar ist, da C ein eingeschränkter Konzeptterm ist. Beim zweiten Fall ($\delta = (a, b) : R$) handelt es sich um eine primitive Rollen-Assertion. Hier ist b der R-Füller von a . Aus der Assertion $a : \forall R.X_{new}$ würde $b : X_{new}$ folgen, wobei X_{new} ein neues atomares „Marker-Konzept“ ist. Dies würde sich offensichtlich mit der Assertion $b : \neg X_{new}$ widersprechen. Der gleiche Trick kann verwendet werden, um zu zeigen, daß $(a, b) : f$ gilt. Hier könnte der Füller von a auch ein Objekt der konkreten Domäne sein, welches sich auch

mit der Assertion $a : \forall R.X_{new}$ widersprechen würde. Das Vorhandensein eines konkreten Domänen-Füllers von a über f kann jedoch durch die Assertion von $\exists(f).is_region$ ² festgestellt werden. Um zu zeigen, ob $(a, x) : f$ gilt, können wir leider nicht einfach einen neuen X_{new} -„Marker“ propagieren, da dies zu einem sofortigen Widerspruch führt (da die abstrakte Domäne und die konkrete Domäne disjunkt sind). Daher müssen wir ein neues Prädikat Ψ der konkreten Domäne als Marker propagieren. Wie oben schon angedeutet, hat Ψ bzw. $\overline{\Psi}$ keine Wirkung auf die Erfüllbarkeit der Prädikate der konkreten Domäne P_i , sodaß für das Zeigen eines Widerspruchs, in Bezug auf Ψ bzw. $\overline{\Psi}$ wir beides ($\Psi(x)$ und $\overline{\Psi(x)}$) für ein konkretes Domänen-Objekt x assertieren müssen. Dabei wollen wir nicht $(a, x) : f$ inferieren, wenn a über f einen Füller in der abstrakten Domäne hat, ebenso auch nicht, wenn es keinen Füller (sowohl in der abstrakten, als auch in der konkreten Domäne) gibt. Daher testen wir die Existenz eines abstrakten Domänen-Füllers durch die Assertion $\exists f.\top$ und analog, wenn es keinen geben sollte, mit $\forall f.\perp$. Im fünften Fall ($\delta = (x_1, x_2) : P$) muß gezeigt werden, daß das binäre Prädikat P der konkreten Domäne für die konkreten Objekt x_1 und x_2 gilt. Da wir zulässige konkrete Domänen betrachten, gibt es auch ein konkretes Domänen-Prädikat \overline{P} . Der letzte Fall scheint etwas problematischer zu sein, da $\mathcal{ALCRP}(\mathcal{S}_2)$ keinen Operator für die Negation von prädikats-basierten Rollen-Axiomen bereitstellt. Trotzdem können wir testen, ob $(a, b) : \exists(has_area)(has_area).\overline{P} \vee a : \neg\exists(has_area).is_region \vee b : \neg\exists(has_area).is_region$ gilt. Die NNF (Negations-Normalform) von $\neg\exists(has_area).is_region$ ist $\neg\exists(has_area).is_no_region \wedge \forall(has_area).\top$.³ Wenn $\neg\exists(has_area).is_no_region$ inkonsistent ist, ist der Ergebnisterm $(a, b) : \exists(has_area)(has_area).\overline{P} \vee a : \forall has_area.\top \vee b : \forall has_area.\top$. Anscheinend ist dies keine $\mathcal{ALCRP}(\mathcal{S}_2)$ -ABox mehr, da die Terme in einer ABox als Konjunkte behandelt werden. Auch wenn wir hier disjunkte Terme vorliegen haben, so können wir zeigen, ob sie zusammengenommen konsistent sind oder nicht. $A \cup \{a_1 \vee a_2 \vee \dots \vee a_n\}$ ist genau dann inkonsistent, wenn $\forall a_i : A \cup \{a_i\}$ inkonsistent ist.

In [9] wird auch das folgende Theorem aufgestellt:

Theorem 5.1 *Das Konsequenzproblem für eine räumlich-terminologische Default-Theorie ist entscheidbar.*

Für eine eingeschränkte $\mathcal{ALCRP}(\mathcal{S}_2)$ ABox kann mit der Hilfe des Tableaux-Kalküls [7] die Konsistenz entschieden werden. Statt $Th(E)$ können wir die

² is_region ist das unäre Prädikat.

³ is_no_region ist das Komplement von is_region .

ABox E als Repräsentanten für eine Extension nehmen. Die Fixpunkt-konstruktion aus 5.2 kann als Tester benutzt werden, um zu entscheiden, ob die ABox E (aus der Definition) wirklich eine Extension der betrachteten Default-Theorie ist. Da jede Extension eine ABox der Form $A \cup \{\gamma | \alpha : \beta_1 \dots \beta_n / \gamma \in D'\}$ für eine Menge generierender Defaults $D' \subseteq D$ ist, können wir für jedes Element E von $\{A \cup X | X \in 2^{\{\gamma | \alpha : \beta_1 \dots \beta_n / \gamma \in D'\}}\}$ testen, ob es eine Extension ist oder nicht. Dazu müssen folgende Inferenzen entschieden werden:

1. $\alpha \in Th(E_i)$: Dies kann getestet werden, indem gezeigt wird, ob $E_i \models \alpha$, wobei $\alpha = \{a_1, a_2, \dots, a_n\}$. Wir können dieses ABox-Folgerbarkeitsproblem genau dann entscheiden, wenn für jedes assertionales Axiom a_i entschieden werden kann (s.o.), ob es eine logische Folgerung von E_i ist ($\forall a_i \in \alpha : E_i \models a_i$).
2. $\neg\beta_i \notin Th(E)$: Dies kann getestet werden, indem $E \not\models \neg\beta_i$ gezeigt wird. $E \not\models \neg\beta_i$ genau dann, wenn $A \cup \beta_i$ konsistent ist. Das ABox-Konsistenzproblem für $\mathcal{ALCRP}(\mathcal{S}_2)$ ABoxen ist entscheidbar.
3. $Th(E) = \bigcup_{i=0}^{\infty} Th(E_i)$: Dieser Fixpunkt kann in endlich vielen Schritten erreicht werden, da wir nur endlich viele Defaults betrachten. Im Prinzip müssen wir das ABox-Äquivalenzproblem entscheiden. Eine ABox A_1 ist äquivalent zu einer ABox A_2 ($A_1 \equiv A_2$) genau dann, wenn $A_1 \models A_2$ und $A_2 \models A_1$. Daher kann das ABox-Äquivalenzproblem auf zwei ABox-Folgerbarkeitsprobleme reduziert werden.

Wenn wir in einigen Fällen die S-Extensionen berechnen möchten, so muß ein weiteres Problem noch gelöst werden. Wie sollen Spezifitäten zwischen Defaults mit ABoxen in den Prämissen bestimmt werden? Ein Default D_1 heißt spezifischer als ein Default D_2 , $D_1 \prec D_2$ g.d.w. $(\alpha(D_1) \models \alpha(D_2) \wedge \alpha(D_2) \not\models \alpha(D_1))$, wobei $\alpha(D)$ für die Prämisse des Defaults steht.

Durch diese Erweiterungen können wir nun Default-Regeln auch im Kontext von $\mathcal{ALCRP}(\mathcal{S}_2)$ benutzen. Diese Theorie für $\mathcal{ALCRP}(\mathcal{S}_2)$ kann auch für $\mathcal{ALCRP}(\mathcal{D})$ erweitert werden.

5.2 Beispiele

In diesem Abschnitt werden einige Beispiele für die Verwendung der räumlich-terminologischen Default-Theorien präsentiert. In den folgenden Beispielen

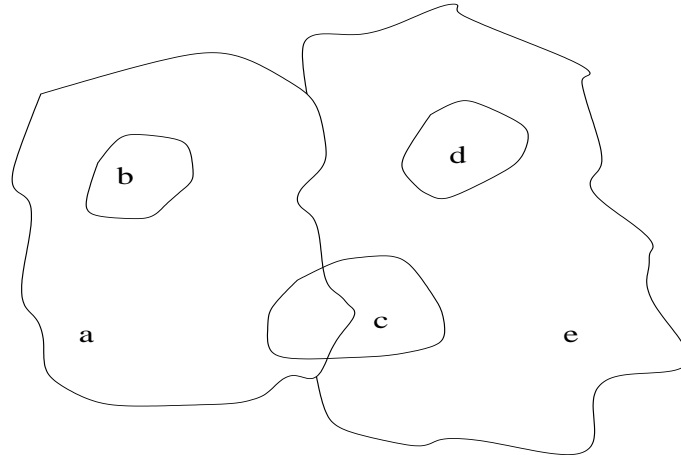


Abbildung 5.2: Eine Karte

wird weiterhin die obige TBox vorausgesetzt.

Die Abbildung 5.2 soll eine Karte sein. Über die Objekte haben wir in der ABox folgende Informationen:

$$\{a : Land, (b, a) : in, (a, e) : beruehrt, (a, c) : ueberlappt, (d, e) : in, (e, c) : ueberlappt\}$$

Bis auf die Information, daß a ein Land ist, haben wir sonst nur Relationen zwischen Objekten in unserer ABox. Betrachten wir nun folgende Defaults:

1. $\{(x, y) : in, y : Land\} : \{x : Stadt\} / \{x : Stadt\}$
2. $\{(x, y) : in, y : Land\} : \{x : See\} / \{x : See\}$
3. $\{(x, y) : beruehrt\} : \{x : Land\} / \{y : Land\}$
4. $\{(x, y) : ueberlappt\} : \{x : Land\} / \{y : See\}$

Hier erwarten wir die folgenden vier Extensionen, die allerdings mit unseren Algorithmen nicht eher als einen Tag ⁴berechnet werden können. Tatsächlich ist es auch nicht anders zu erwarten, da nach dem „Closing“ (Instantiieren der freien Variablen) 100 Defaults ($5^2 * 4$, wegen 5 Objekten, 2 Variablen und 4 Defaults) entstehen.

⁴Dieser Versuch wurde nach einem Tag abgebrochen.

- $E_1 = \{a : Land, b : Stadt, c : See, d : Stadt, e : Land\} \cup ABox$
- $E_2 = \{a : Land, b : Stadt, c : See, d : See, e : Land\} \cup ABox$
- $E_3 = \{a : Land, b : See, c : See, d : Stadt, e : Land\} \cup ABox$
- $E_4 = \{a : Land, b : See, c : See, d : See, e : Land\} \cup ABox$

Über das Objekt c kann eindeutig gesagt werden, daß es ein *See* sein muß (sceptical reasoning). Eindeutig kann auch über das Objekt e gesagt werden, daß es ein *Land* ist. Die Objekte b und d können Städte oder Seen sein (credulous reasoning). Hier wurde anhand der bekannten Objektbeziehungen Konzepte (Konzeptzugehörigkeiten) inferiert.

Natürlich ist die umgekehrte Richtung auch denkbar. Im folgenden Beispiel werden anhand der Konzeptzugehörigkeiten (Assertionen in unserer ABox) die Objektbeziehungen (Relationen) inferiert, bzw. hinzugefügt.

Nehmen wir an, wir haben keine Karte, bzw. keine Möglichkeit, anhand der wir die Relationen zwischen Objekten erkennen könnten. Unser ABox liefert uns folgende Informationen:

$\{a : Deutschland, a : Land, b : Frankreich, b : Land, c : Italien, c : Land\}$

Wir können die Beziehungen dieser Objekte anhand der folgenden, von uns definierten Defaults bestimmen.

1. $\{x : Deutschland, y : Frankreich\} : \{(y, x) : beruehrt\} / \{(y, x) : beruehrt\}$
2. $\{x : Deutschland, y : Italien\} : \{(y, x) : disjunkt\} / \{(y, x) : disjunkt\}$

Durch diese Defaults können nun die Beziehungen zwischen den Ländern (hier Konzepten) *Deutschland* und *Frankreich*, sowie *Deutschland* und *Italien* assertiert (geschlossen) werden. Zwischen *Deutschland* und *Frankreich* wird geschlossen, daß sie sich berühren. Für *Deutschland* und *Italien* wird geschlossen, daß sie sich nicht berühren. Weltliche Zusammenhänge, in unserem Fall räumliche Beziehungen, die in der Welt ihre Richtigkeit besitzen (*Deutschland* und *Frankreich* sind Nachbarländer), könnten auch, wie oben der Fall war, in Form von Defaults ausgedrückt werden.

An dieser Stelle will ich ein Beispiel für die S-Extensionen einführen. In unserer ABox sollen nur folgende Information vorhanden sein:

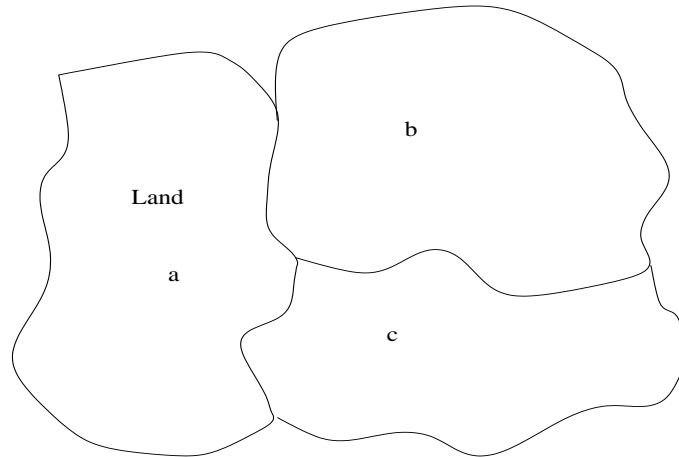


Abbildung 5.3: Eine Karte

$\{a : \text{Land}, (a, b) : \text{beruehrt}, (a, c) : \text{beruehrt},$
 $(b, c) : \text{beruehrt}, b : \text{Gebiet}, c : \text{Gebiet}\}$

Die Abbildung 5.3 soll die ABox besser verdeutlichen. Betrachten wir jetzt die folgenden Defaults:

- $D_1 = \{x : \text{Gebiet}, y : \text{Gebiet}, (x, y) : \text{beruehrt}\} : \{x : \text{Land}\} / \{y : \text{Land}\}$
- $D_2 = \{x : \text{Gebiet}, y : \text{Gebiet}, (x, y) : \text{beruehrt}\} : \{x : \text{Stadt}\} / \{y : \text{Stadt}\}$
- $D_3 = \{x : \text{Land}, y : \text{Gebiet}, (x, y) : \text{beruehrt}\} : \{x : \text{Land}\} / \{y : \text{Land}\}$

Hier ist D_3 spezifischer als D_1 und D_2 ($D_3 \prec D_1, D_3 \prec D_2$), und damit auch aktiv. D.h. wir bekommen nur einen Extension, daß a , b und c Länder sind.

Wie wir aus den bisherigen Beispielen entnehmen konnten, eignen sich Defaults gut für die Vervollständigung unvollständiges Wissens (auch im räumlichen Kontext). Die Kombination von Defaults mit der Sprache $\mathcal{ALCRP}(\mathcal{S}_2)$ kann in einigen Systemen, wie geographischen Informationssystemen, genutzt werden. Es sind aber auch andere Anwendungen denkbar, in denen die Einbeziehung von Defaults Vorteile bringen kann.

Aus den früheren Kapiteln ist schon bekannt, daß die angegebenen Algorithmen unterschiedliche Zeiten zum Berechnen der Extensionen benötigen. Durch unsere Erweiterungen werden diese Unterschiede noch deutlicher. Auf dieses Phänomen will ich im nächsten Kapitel näher eingehen.

Kapitel 6

Optimierungstechniken für Inferenzalgorithmen

Zum Berechnen der Extensionen werden einige Prozeduren benötigt, die exponentielle Eigenschaft besitzen, und daher teuer sind, wie der Erfüllbarkeitstest und das Berechnen der Teilmengen. Für die Optimierung ist es also von großer Bedeutung die genannten Punkte stets im Auge zu behalten. Zum einen sollen möglichst wenig teure Operationen verwendet werden, und zum anderen sollen möglichst nur die Mengen von „relevanten“ Defaults, und somit auch ihre Teilmengen, betrachtet werden.

Aus dem letzten Kapitel wurde deutlich, daß wenn wir ABoxen in α , β und γ zulassen wollen, wir das Problem des Instantiierens anders behandeln müssen, denn sonst würden unnötige Defaults entstehen, die wiederum das Berechnen der Extensionen „erschweren“ würden. Dieser Punkt dient in diesem Kapitel als einer der Diskussionspunkte.

Desweiteren soll anhand einiger Test-Ergebnisse gezeigt werden, daß der Algorithmus von Baader und Hollunder (s. Kapitel 3.3) schneller ist, als der definitionsbasierte Algorithmus (s. Kapitel 3.1). In diesem Zusammenhang werden einige Gründe für die unterschiedliche Zeiten gegeben.

6.1 Instantiieren der Defaults

Wie schon erwähnt müssen wir zum Instantiieren der Defaults uns alle in der ABox vorkommenden Individuen nehmen und kombinatorisch mit allen

freien Variablen in den Defaults substituieren. Danach soll auf die neu entstandenen geschlossenen Defaults eine Filterung angewendet werden, sodaß solche Defaults entfernt werden, die nie eine Extension erzeugen könnten.

Algorithm 1 instantiiere-Default-Regel(*default*, *ABox*)

```

res :=  $\emptyset$ 
Var := finde-alle-freien-Variablen(default)
Ind := finde-alle-Individuen(ABox)
Anz := |Var|
Komb := IndAnz
for X ∈ Komb do
    res := res ∪ substituiere(default, X)
end for
return res

```

Eine Prozedur zum Instantiiieren (Substituieren) der Defaults mit allen Individuen.

In der obigen Prozedur wird eine Default-Regel zusammen mit der ABox betrachtet. Zuerst werden alle freie Variablen, die in dieser Default-Regel verwendet werden mit der Funktion *finde-alle-freien-Variablen* ermittelt. Es empfiehlt sich hierbei eine Baumsuche, da die Axiome verschachtelt sein können.¹ Nachdem die Anzahl der freien Variablen in dieser betrachteten Default-Regel ermittelt ist, können nun alle möglichen Kombinationen der Individuen aus der ABox hierauf abgebildet werden. Damit ist das Kreuzprodukt der Individuen auf der Anzahl der betrachteten Variablen in den Default-Regeln gemeint. Diese möglichen Kombinationen werden nach und nach mit den freien Variablen substituiert. Am Ende gibt es nun eine Menge von geschlossenen Default-Regeln. Danach kann diese Prozedur zum Schließen weiterer offener Default-Regeln genutzt werden.

Es soll an dieser Stelle notiert werden, daß zumindest meine Überlegungen, wie etwa das Finden von nur „relevanten“ Individuen, d.h. Individuen die wirklich Instanzen der Axiome in α , β und γ sein können, zu keinen brauchbaren Ergebnissen geführt haben; die Sache unnötigt erschwert und verlangsamt haben.

Die Filterung soll wie folgt vorgenommen werden:

- Zunächst verlangen wir, daß die Vorbedingung α von unserem Default mit A konsistent ist. Ist dies nicht der Fall, so kann sie nicht aus A

¹Natürlich hängt dies von der Sprache ab, in dem die Axiome präsentiert werden. In unserem Fall ist es Lisp, und daher listen.

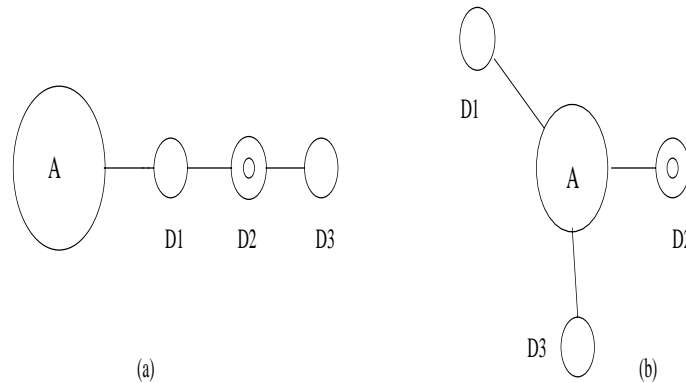


Abbildung 6.1: Zwei mögliche Verankerungen.

folgen, d.h. die Vorbedingung wäre nie erfüllt, und daher ist dieser Default unbrauchbar.

- Dann verlangen wir, daß $A \cup \beta_i$ konsistent ist. Wenn dies nicht der Fall ist, so kann γ nicht geglaubt werden, und d.h. daß dieser Default niemals in der Menge der generierenden Defaults einer Extension sein kann.
- Zuletzt wollen wir $A \cup \gamma$ auf Konsistenz prüfen. Dies aber machen wir erst, wenn der Default weder *normal* noch *semi-normal* ist.²

Die Vorteile des zweiten Tests könnten sich insbesondere dann bemerkbar machen, wenn wir die Extensionen mit dem Algorithmus von Baader und Hollunder berechnen wollen. In der Definition von Verankerung 3.2 wird β nicht betrachtet, d.h. es können zu große verankerte Mengen von Defaults geben. Die Abbildung 6.1 soll uns 2 mögliche Verankerungen ohne diesen Test zeigen. β von $D2$ (im Bild mit einem zusätzlichen Kreis markiert) soll mit A (unsere ABox) inkonsistent sein, und deswegen rausgenommen werden. Das bedeutet im Falle von Abbildung 6.1 (a), daß $D3$ auch nicht mehr verankert sein kann. Wenn allerdings die Defaults wie in der Abbildung 6.1 (b) verankert sind, so hat unsere Einschränkung jedoch keine größere Wirkung. Dieser Test bekommt gerade dann einen Stellenwert, wenn in den β 's eine andere freie Variable als die in α verwendet worden ist. Dies kann auch der Fall sein, da wir beim Formulieren unserer Defaults keinerlei Einschränkungen unterlegen sind.

²Zur Erinnerung: Eine Default-Regel ist *semi-normal* genau dann, wenn ihre Annahme (justification) die Konsequenz impliziert (aus der Annahme folgt die Konsequenz), und sie ist *normal* genau dann, wenn ihre Annahme mit der Konsequenz identisch ist ($\alpha : \beta/\beta$).

Z.B:

- $\{x : C, y : D\} : \{z : E\} / \{x : F\}$

Nach dem Instantiieren könnten für z auch Individuen substituiert sein, die zusammengenommen mit der ABox eine Inkonsistenz verursachen können.

Der letzte Punkt wird nur dann für einen Default getestet, wenn er weder normal noch semi-normal ist, denn wenn er dies ist, dann haben wir mit dem zweiten Punkt dies schon getestet.

Diese Filterung bewirkt, daß tatsächlich nur noch einige geschlossene Defaults übrig bleiben.

6.2 Gegenüberstellung: Qualitativer Vergleich der Algorithmen

In diesem Abschnitt will ich den aus Reiter's Definition der Extensionen gewonnenen definitionsbasierten Algorithmus aus Kapitel 3.1 (hier abgek. als (1)) mit dem Algorithmus von Baader und Hollunder aus Kapitel 3.3 (hier abgek. als (2)) vergleichen und gegenüberstellen.

Die Tabelle soll uns zeigen, welche unterschiedlichen Zeiten zum Berechnen der Extensionen, jeweils von den Algorithmen, für unterschiedliche Tests (mit unterschiedlichen „Schwierigkeitsgraden“³) benötigt werden.

	Defaults mit nur 2 Variablen	Anzahl der Individuen	Zeit
(1)	1	2	1,1msec
(2)	1	2	1,0msec
(1)	1	3	39,8msec
(2)	1	3	7,5msec
(1)	2	2	4,5msec
(2)	2	2	1,4msec
(1)	2	3	248msec
(2)	2	3	12,4msec

In diesen Beispielen werden Defaults mit nur zwei freien Variablen verwendet. Weiter werden in diesen Beispielen nur eine geringe Anzahl von Defaults und

³Damit ist die Anzahl der Defaults und ihre Konflikte untereinander gemeint.

relativ kleine ABoxen mit bis zu drei Individuen verwendet, um die Zahl der geschlossenen Defaults möglichst klein zu halten. Bei einer größeren Anzahl von Defaults oder Individuen brauchen die Algorithmen Zeiten, für die wir uns keine Anwendungsszenarien dafür vorstellen können. Was man aber nicht aus der Tabelle entnehmen kann, ist daß beide Verfahren im „worst case“ nicht zu unterscheiden sind, da beide Verfahren von dem Berechnen von Teilmengen, welches exponentiell in der Zeit ist, abhängen. Trotzdem kann anhand dieser Test gesehen werden, daß die Extensionen mit (2), zumindest in diesen Fällen, schneller berechnet werden als mit (1).

Es gibt offensichtlich einige Gründe dafür, warum wir hier unterschiedliche Zeiten bekommen. Als wichtigster Grund ist zu erwähnen, daß in (1) alle erfüllbaren Teilmengen von Defaults als mögliche Extensionen (generierenden Defaults) in Betracht gezogen werden, während in (2) nur die maximal erfüllbaren Teilmengen D' von Defaults D , die mit $W \cup Con(D')$ konsistent sind, ins Betracht gezogen werden. Die Anzahl der Elemente dieser Mengen können sich, insbesondere wenn viele Defaults vorliegen, enorm unterscheiden.

Als weiterer Grund, zumindest für diese Beispiele, kann die unterschiedliche Vorgehensweise zur Berechnung von Extensionen in den beiden Algorithmen gezählt werden. Während in (1) in jedem Testdurchlauf und in jedem Iterationsschritt die Konklusion γ erst dann hinzugenommen wird, wenn sowohl $\alpha \in Th(E_i)$ und $\neg\beta_1, \dots, \neg\beta_n \notin Th(E)$ gilt, werden in (2) durch die Definition der Verankerung zunächst die β' s außer acht gelassen, und später, sofern es tatsächlich einige β' s gibt, die mit γ' s der anderen Defaults im Konflikt stehen,⁴ auf sie eingegangen und somit das Problem gelöst. Grob gesagt wird in (2) das Ziel zunächst grob skizziert, und später feiner gezeichnet. Offenbar ist (2) für solche Default-Theorien besser geeignet, deren β' s keine oder wenig Konflikte mit sich bringen (z.B. normale Defaults). Wenn nur normale Defaults vorliegen, dann werden mit (2) die Extensionen noch schneller berechnet, da $W \cup Con(D_0) \models \neg\beta$ (wobei D_0 die verankerte Menge ist) nie eintreffen wird, und somit die zusätzliche Arbeit mit β ausfällt. Wir sind über die Ergebnisse unserer Tests (s.o.) nicht weiter erstaunt, da aufgrund der geringen Anzahl der in diesen Beispielen verwendeten Defaults keine sonderlich schweren Konflikte zwischen den Defaults zu erwarten sind.

Es bleibt also ein offener Punkt, zu zeigen, inwiefern (2) wirklich schneller als (1) ist. Trotzdem kann (2) als eine Optimierung von (1) angesehen werden, weil es zumindest keinen Fall geben könnte, in dem (2) langsamer als (1)

⁴Im Nixon Beispiel gab es nur 2 Defaults, dessen β' s sich mit γ' s von den anderen widersprachen.

wäre, im Gegenteil kann es, so haben wir oben gesehen, Fälle geben, in welchen wir nur mit normalen Defaults zu tun haben, wo (2) dann deutlich schneller ist als (1). Es ist jedoch erwähnenswert, daß vom Verständnis her (1) einfacher ist als (2).

6.2.1 Maximal erfüllbare Teilmengen

Ein Optimierungspunkt ist das Bilden der maximal erfüllbaren Teilmengen, welches in dem Algorithmus von Baader und Hollunder gefordert wird. Hier kann man den naiven Weg gehen, und alle Teilmengen auf Erfüllbarkeit testen, und daraus die maximal erfüllbaren Teilmengen nehmen.

In der unten dargestellte Prozedur *maximal-erfüllbare-Teilmengen* wird zunächst die Potenzmenge für die betrachtete ABox A generiert. Dann wird jedes Element dieser generierten Menge auf Erfüllbarkeit geprüft. Zuletzt wird dann für jede erfüllbare Menge untersucht, ob es keine andere Menge dieser erfüllbaren Menge gibt, sodaß unsere betrachtete Menge Teilmenge dieser Menge ist. Wenn dies nicht der Fall ist, so ist unsere betrachtete Menge maximal. Dieser Weg ist deshalb ineffizient, da teure Operationen, wie der Erfüllbarkeitstest, unnötig oft genutzt werden. Wenn eine Menge a nicht erfüllbar ist, dann kann eine andere Menge b die a enthält (also $a \subseteq b$) nie erfüllbar sein. Mit dieser Idee kann die Anzahl der teuren Tests reduziert werden.

Algorithm 2 maximal-erfüllbare-Teilmengen($ABox$)

```

sat $p$  :=  $\emptyset$ 
 $P := 2^{ABox}$ 
for  $X \in P$  do
  if  $X$  ist erfüllbar then
    sat $P$  := sat $P$   $\cup$   $\{X\}$ 
  end if
end for
max-sat-set :=  $\emptyset$ 
for  $Y \in sat_p$  do
  if  $\neg \exists Z \text{ in } sat_P, Y \subset Z$  then
    max-sat-set := max-sat-set  $\cup$   $\{Y\}$ 
  end if
end for
return max-sat-set

```

Der naive Weg zur Berechnung der maximal erfüllbaren Teilmengen.

Besser ist es aber, beim Generieren der Teilmengen, beginnend bei den kleinsten (z.B. einelementigen Teilmengen), zu testen, ob sie erfüllbar sind, und solche, die nicht erfüllbar sind, gleich zu entfernen und in den restlichen Teilmengen, die wir generieren, diese im vorhinein zu vermeiden (eine Art „Vorbeugung“). Dieser Weg scheint insbesondere dann sinnvoll zu sein, wenn wir in unserer Menge viele Elemente haben, die sich widersprechen (zusammengenommen einen clash verursachen). In der unten dargestellte Prozedur *optimierte-maximal-erfüllbare-Teilmengen*⁵ wird genau dieses beachtet, und somit die Anzahl für die Verwendung der teuren Operationen (Erfüllbarkeit) auf einige beschränkt. Hier wird die Funktion *berechne-n-elementige-Teilmengen* verwendet, die aus einer ABox in Abhängigkeit von n n -elementige Teilmengen generiert. In diesem Fall wird die oben erwähnte Idee ausgenutzt. Die Elemente dieser berechneten n -elementigen Teilmengen werden nicht immer auf Erfüllbarkeit geprüft. Es werden nur die Elemente (Mengen) geprüft, die nicht Obermenge der nicht erfüllbaren Mengen (*negres*) sind.

Wenn aber die Anzahl der widersprüchlichen Elemente klein ist, so ist es besser, den umgekehrten Weg zu gehen. In diesem Fall fangen wir mit der Menge selbst an, und entfernen die Elemente, die eine Inkonsistenz verursachen. Da wir aber von einer unbestimmten Anzahl von Defaults reden, deren Inhalt, also α , β und γ in unserem Fall ABoxen sind, so ist es wahrscheinlicher, daß viele Konflikt-Situationen in der betrachteten Menge existieren, sodaß sich der erste Weg empfiehlt.

Eine Prozedur zur Berechnung der maximal erfüllbaren Teilmengen.

⁵Zugegeben ist diese Prozedur im schlechtesten Fall genau so schlecht bzw. gut wie die obigen Prozedur.

Algorithm 3 optimierte-maximal-erfüllbare-teilmengen($ABox$)

```

n := 1
max-sat-set :=  $\emptyset$ 
res :=  $\emptyset$ 
negres :=  $\emptyset$ 
ABox-laenge :=  $|ABox|$ 
for  $m = 1$  to ABox-laenge do
  nkt := berechne-n-elementige-Teilmengen(ABox, n)
  for  $X \in nkt$  do
    if  $\exists Y \in negres, \text{soda\ss} Y \subset X$  then
      negres := negres  $\cup$   $\{X\}$ 
    else
      if erfuellbar( $X$ ) then
        res := res  $\cup$   $\{X\}$ 
      else
        negres := negres  $\cup$   $\{X\}$ 
      end if
    end if
    n := n + 1
  end for
end for
for  $Y \in res$  do
  if  $\neg \exists Z \text{in } res, Y \subset Z$  then
    max-sat-set := max-sat-set  $\cup$   $\{Y\}$ 
  end if
end for
return max-sat-set

```

Kapitel 7

Zusammenfassung

In dieser Arbeit wurde die Repräsentation von räumlich-terminologischem Wissen durch beschreibungslogische Default-Theorien mit Spezifität untersucht. Das Repräsentieren und Schließen über räumliche Gegebenheiten ist mit der Sprache $\mathcal{ALCRP}(\mathcal{S}_2)$ (als Instanz der Sprache $\mathcal{ALCRP}(\mathcal{D})$) gut möglich, sodaß diese für Anwendungen, wie geographische Informationssysteme gut eingesetzt und gebraucht werden kann. Die Notwendigkeit Defaults in solche Systeme einzubeziehen wurde durch einige Beispiele verdeutlicht. Dabei wurde dann von den bestehenden Default-Theorien (allgemein Reiter's Default-Theorie) ausgegangen, und diese zur räumlich-terminologischen Default-Theorie erweitert. Dieser Ansatz der räumlich-terminologischen Default-Theorie kann nun $\mathcal{ALCRP}(\mathcal{S}_2)$ -Terme behandeln, und unterscheidet sich von den am Anfang ausgegangenen Theorien dadurch, daß statt Konzeptterme in α, β und γ ABoxen stehen. Es wurde hier gezeigt, daß die vorgenommenen Erweiterungen, aus der theoretischen Sicht auch machbar sind, und daß die behandelten Algorithmen weiterhin prinzipiell anwendbar bleiben.

Weiter wurde anhand von einigen Beispielen die Signifikanz dieser Erweiterung für einige Anwendungen angedeutet. Den Beispielen konnte man auch entnehmen, daß die Hinzunahme von Defaults zu den bestehenden Systemen zu Schlüssen führen kann, die mit den bisher betrachteten Inferenzen nicht möglich waren.

Zum Schluß wurde das Verfahren von Baader und Hollunder und das definitionsbasierte Verfahren gegenübergestellt und ihre unterschiedlichen Vorgehensweisen und damit verbundenen Konsequenzen herauskristallisiert. Es stellte sich dabei nochmal heraus, daß das Verfahren von Baader und Hollunder besser ist als das definitionsbasierte Verfahren, insbesondere wenn es um

größere Mengen von Defaults geht. Danach wurden Ideen zur Optimierung für die durch die Erweiterung neu entstandenen Schwächen besprochen und dargestellt.

Danksagung

In erster Linie danke ich meinem Betreuer, Herrn Dr. Ralf Möller, für seine ständige Gesprächsbereitschaft. Herrn Michael Wessel danke ich für zahlreiche Tips beim Formulieren und bei der Korrektur dieser Arbeit sowie Ratschläge im Umgang mit \LaTeX . Frau Anni-Yasmin Turhan danke ich für die Bereitstellung der $\mathcal{ALC}(\mathcal{D})$ und $\mathcal{ALCRP}(\mathcal{D})$ -Konsistenztesters. Herrn Dr. Volker Haarslev danke ich für die Bereitstellung des TBox- und ABox-Moduls sowie des RCC 8-Konsistenztesters. Allen Mitarbeitern des Arbeitsbereiches „Kognitive Systeme (KOGS)“ danke ich für ihre freundliche Hilfe und Unterstützung bei diversen Problemen sowie für die angenehme Arbeitatmosphäre.

Hiermit erkläre ich die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

25. Oktober 1999

Omar Soltan Nuri

Literaturverzeichnis

- [1] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 452–457, 1991.
- [2] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Principles of Knowledge Representation and Reasoning*. Morgan-Kaufmann, 1992.
- [3] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. In *International Joint Conference on Artificial Intelligence, IJCAI '93*. Morgan Kaufmann, 1993.
- [4] M.J. Egenhofer. Reasoning about binary topological relations. In *O. Günther and H.-J. Schek, editors, Advances in Spatial Databases, Second Symposium, SSD'91, Zurich, Aug. 28-30, 1991, volume 525 of Lecture Notes in Computer Science*, pages 143–160. Springer Verlag, Berlin, August 1991.
- [5] G. Görz. *Einführung in die Künstliche Intelligenz*. Addison-Wesley, 1993.
- [6] V. Haarslev, C. Lutz, and R. Möller. Foundation of spatioterminological reasoning with description logic. In *Proceedings of the Sixth International Conference (KR'98)*. Morgan-Kaufmann, 1998.
- [7] V. Haarslev, C. Lutz, and R. Möller. A description logic with concrete domains and role-forming predicate operator. In *Journal of Logic and Computation*, 1999.
- [8] C. Lutz. Repräsentation topologischer Informationen in Beschreibungslogiken. Master's thesis, Universität Hamburg, Fachbereich Informatik, 1998.

- [9] R. Möller and M. Wessel. Terminological default reasoning about spatial information: A first step. In *International Conference on Spatial Information Theory*. Springer-Verlag, 1999.
- [10] D.A. Randell, Z. Cui, and A.G. Cohn. A spatial logic based on regions and connections. In *B. Nebel and C. Rich, and W. Swartout, editors, Principles of Knowledge Representation and Reasoning, Cambridge, Oct. 25-29, 1992*, pages 165–176. Mass, Okt. 1992.
- [11] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. In *Artificial Intelligence 48 (1)*, pages 1–26, 1991.
- [12] C. Schwind and V. Risch. A tableau-based characterisation for default logic. In *Proceedings of the 1st European Conference on Symbolic and Quantitative Approaches for Uncertainty*, pages 310–317, 1991.
- [13] J. F. Sowa. Issues in knowledge representation. In *Principles of Semantic Networks*. Morgan Kaufmann, 1991.
- [14] N. Wahllöf. A default extension to description logics and its applications. Master's thesis, Linköping University, 1996.