Berechnung einer erweiterten Subsumtionshierarchie

Madjid Nassiri
Matrikelnummer 179512

Diplomarbeit
im Fach Informatik

Lehr- und Forschungsgebiet Theoretische Informatik
Rheinisch-Westfälische Technische Hochschule Aachen
Prof. Dr.-Ing. Franz Baader

Betreuung:
Prof. Dr.-Ing. Franz Baader

Aachen, im Juni 1997

Ich widme diese Arbeit meiner lieben Frau Zahra, deren Dasein mir die Kraft gegeben hat, die schwierigsten Phasen meines Lebens zu überwinden.

Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, im Juni 1997 .....

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Softwareentwicklung in den letzten Jahren ist geprägt durch den Übergang von der Datenverarbeitung zur Wissensverarbeitung. Die Grenzen der konventionellen Datenverarbeitung werden immer deutlicher. Die Wissensverarbeitung versucht, Problemlösungsmethoden des Menschen im Rechner zu modellieren und nachzubilden. Wissensbasierte Systeme sollen die Möglichkeit eröffnen, eine erweiterte Klasse von Problemstellungen, die durch konventionelle Software-Systeme nicht angegangen werden können, mit Hilfe des Rechners zu lösen. Konventionelle Systeme enthalten die vollständige Strategie zur Lösung des Anwendungsproblems in fest codierter Form. Zur Laufzeit eines solchen Systems kann nur dieser Lösungsalgorithmus abgearbeitet werden. Ändert sich die Umgebung des Systems oder die Anforderungsspezifikation, so ist eine Neukonstruktion erforderlich, die meist eine Neuprogrammierung bedingt. In einem konventionellen System ist also jeder Zwischenschritt durch die fest programmierte und vollständig geplante Lösungsstrategie vorgegeben und nicht veränderbar. Wissensbasierte Systeme enthalten das Wissen über die Lösung eines Problems innerhalb einer jederzeit manipulierbaren Wissensbasis. Sie transformieren nicht nur Eingangs- zu Ausgangsgrößen, sondern verarbeiten das Wissen aus der Wissensbasis zur Suche einer Lösungsstrategie für das gerade bearbeitete Problem. Die Wissensbasis enthält also nicht nur reines Faktenwissen, sondern auch Problemlösungswissen, das durch Experten eingebracht wird.

In der Wissensverarbeitung wird versucht, den menschlichen Experten durch Expertensysteme, wo immer möglich, zu unterstützen oder sogar zu ersetzen. Ein Expertensystem ist ein Wissensbasiertes System, das das Wissen eines qualifizierten Experten für ein bestimmtes Gebiet modelliert, um komplexe Probleme eines spezifischen Typs aus diesem Gebiet zu lösen [1]. Das Expertensystem muß natürlich über das Wissen und die Erfahrung des Experten verfügen und eine direkte Interpretation der gelieferten Daten vornehmen können. Ein Expertensystem besteht im wesentlichen aus einer Wissensbasis, einer Schlußfolgerungskomponente (Inferenzmaschine) und einer komfortablen Benutzeroberfläche, die

einen natürlichsprachlichen Dialog sowie eine Erklärung des Systemverhaltens anbieten soll. Zentrale Bedeutung kommt dabei der Wissensbasis und der Schlußfolgerungskomponente zu.

Einschränkend soll schon hier erwähnt werden, daß die Wissensverarbeitung bisher lediglich in streng begrenzten Wissensgebieten eingesetzt wird. Die Gründe dafür sind einerseits die Komplexität realer Probleme und die dadurch verursachte kombinatorische Explosion des Suchraumes und andererseits der derzeitige Entwicklungsstand der Hardware sowie die Grenzen, die durch die Formalismen der Wissensdarstellung gesetzt werden. Es ist leider in vielen Aufgabenbereichen der Fall, daß es entweder gar keine geeigneten Formalismen zur Wissensrepräsentation (knowledge representation) gibt, oder sie sind von so enormer Zeit- bzw. Speicherkomplexität, daß sie nicht praktisch anwendbar sind. Wissensverarbeitungssysteme werden in einem Teilbereich der Informatik entwickelt, der mit ,,künstliche Intelligenz" (kurz **KI**) bezeichnet wird. Die **KI** befaßt sich mit der Lösung von Problemen, die beim Menschen ,,intelligentes" Verhalten voraussetzen. Diese mehr informelle Beschreibung ist dadurch begründet, daß es bis heute keine allgemein anerkannte Definition von **KI** und auch keine exakte Definition des Begriffes ,,Intelligenz" gibt. Zur Charakterisierung dient meist eine Aufzählung wichtiger Teilbereiche, wie z. B. Expertensysteme.
Ein grundlegendes Problem bei der Erstellung von Expertensystemen ist die Repräsentation von Wissen im Rechner. Diese wird oft als definierendes Merkmal von KI-Systemen angesehen. Heute verfügbare Methoden sind meist Entwicklungen, die aus speziellen Implementierungen von wissensbasierten Systemen hervorgegangen sind. Allgemeine Ansätze und Integration verschiedener Methoden sind der Gegenstand heutiger Forschungen auf diesem Gebiet.

Die Wissensrepräsentationsformalismen lassen sich in folgende Kategorien einteilen[1]:

- logik-orientierte Repräsentation
- regelbasierte Repräsentation
- objekt-orientierte Repräsentation
- Repräsentation mit semantischen Netzen

Die Grenzen zwischen diesen Kategorien sind zumeist nicht exakt zu definieren, vielmehr gibt es gleitende Übergänge und Mischformen.
Eine wesentliche Anforderung an Wissensrepräsentationsysteme (kurz **KR**-Systeme) ist eine strukturierte Modellierung des Wissens. Die strukturierte Speicherung von Wissen in einem Expertensystem scheint insbesondere dort vorteilhaft, wo neues Wissen erforscht wird. Das Wissen kann auf diese Weise schnell

[1] Für mehr Information siehe z.B. [2, 3, 4, 5, 6, 7]

in die Praxis technischer Systeme eingebracht werden. Eine solche strukturierte Speicherung des Wissens bieten **terminologische** KR-Systeme als eine Teilklasse der logik-orientierten Repräsentation an. Das Wissen über einen bestimmten Anwendungsbereich wird u.a. als eine Menge von Eigenschaften beschrieben, die für Elemente (Objekte) des Wissensgebietes relevant sind. Zum Beispiel können, im Bereich ,,Blutinfektionen" als einem speziellen Gebiet der Medizin, die Infektionen als Objekte und die Symptome als Eigenschaften betrachtet werden, die diese Objekte besitzen[2]. Solche charakteristische Eigenschaften werden in terminologischen Repräsentationssystemen als sogenannte ,,Konzeptnamen" definiert. Konzeptnamen stehen für einfache Eigenschaften. Um ein bestimmtes Anwendungsgebiet realistisch zu modellieren, muß man aber auch zusammengesetzte (kombinierte) Eigenschaften beschreiben können. Als Beispiel betrachten wir eine Familie. Mitglieder einer Familie haben verschiedene Verwandtschaftseigenschaften wie Vater, Mutter, alleinerziehend und so weiter. Man will z.B. wissen, ob jemand, der alleinerziehend ist und keinen Sohn hat, eine Tochter hat. Zusammengesetzte Eigenschaften werden in terminologischen Sprachen durch ,,Konzeptterme" beschrieben. Man definiert sie, in dem man die einfachen Konzeptnamen mit geeigneten Operatoren verknüpft.

Die Folgerung einer Menge von Eigenschaften aus anderen Eigenschaften wird in terminologischen KR-Systemen durch die sogenannte ,,Subsumtion" beschrieben. Dabei entspricht ,,Konzept C wird durch das Konzept D subsumiert" der logischen Implikation ,,Die Eigenschaft C impliziert die Eigenschaft D". Die Berechnung einer Hierarchie zwischen allen für ein Gebiet relevanten Eigenschaften ermöglicht eine strukturierte Speicherung des Wissens über dieses Gebiet. In heutigen terminologischen KR-Systemen ist leider diese Berechnung nur für einfache Eigenschaften (Konzeptnamen) realisiert. D.h. die Abhängigkeiten zwischen zusammengesetzten Eigenschaften folgt aus dieser Anordnung nicht. Es ist also wünschenswert eine **erweiterte Subsumtionhierarchie** zu berechnen, die es ermöglicht, nicht nur einfache Eigenschaften, sondern auch kombinierte Eigenschaften zu behandeln (Merkmalexploration).

Die vorliegende Arbeit befaßt sich mit dem Problem der ,,Berechnung einer erweiterten Subsumtionshierarchie" mit Hilfe von Merkmalexploration. Als Beispiel soll eine terminologische KR-Sprache (knowledge representation language) Namens $\mathcal{ALC}$ dienen. Die Berechnung soll Grundlage für die Implementation eines Systemdienstes sein, der eine minimale, vollständige Hierarchie bzgl. der Subsumtionen zwischen Konjunktionen von Konzeptnamen (im folgenden auch Merkmale bzw. Eigenschaften genannt) liefert. Hierfür werden u.a. Resultate aus

[2]Das Wohl bekannteste Expertensystem **MYCIN** ist für die Diagnose und Behandlung von Blutinfektionen implementiert und wird weltweit mit Erfolg eingesetzt (siehe [8]).

dem Gebiet der „Formalen Begriffsanalyse“ herangezogen[3].
Im nächsten Kapitel wird die Syntax und Semantik der terminologischen Sprache $\mathcal{ALC}$ beschrieben und anschließend kurz auf das *Erfüllbarkeitsproblem* und seine *Komplexität* für $\mathcal{ALC}$ eingegangen. Im Kapitel 3 folgt die Einführung in das *Subsumtionsproblem* für $\mathcal{ALC}$. Das vierte Kapitel beschäftigt sich mit der *Formalen Begriffsanalyse*. Es werden zunächst die grundlegenden Begriffe definiert und die Zusammenhänge zwischen diesen Begriffen erläutert. Anschließend werden einige Algorithmen beschrieben, die die Berechnung der Begriffshierarchie eines Kontextes ermöglichen. Im Kapitel 5 wird auf den Zusammenhang zwischen formaler Begriffsanalyse und erweiterter Subsumtionshierarchie eingegangen. Im Kapitel 6 werden Optimierungen für das Verfahren untersucht. Im siebten Kapitel wird die Implementierung des Verfahrens beschrieben und eine statistische Auswertung für das Programm vorgenommen. Im letzten Kapitel folgt eine Zusammenfassung dieser Arbeit.

[3]Die Idee und Überlegung, Resultate aus der Welt der „Formale Begriffsanalyse“ zum Ziel der Berechnung einer erweiterten Subsumtionshierarchie einzusetzen stammt aus [9]. Diese Idee ist die Grundlage für diese Arbeit.

Kapitel 2

Terminologische KR-Systeme

Die Unzufriedenheit mit mangelhafter Semantik anderer KR-Methoden wie z.B. semantischer Netze hat eine Präzisierung und Formalisierung von Wissen unumgänglich gemacht. Diese Tatsache hat in den letzten Jahren zur Entwicklung einer Reihe von logik-basierten **Konzeptsprachen** geführt, die die Stärke der Logik, mächtiger Formalismus verbunden mit klarer Semantik, ausnutzen. Hierbei hat man u.a. die bekannten Ergebnisse der Logik angewendet, um das Wissen in einer Form darzustellen, die übersichtlich und leicht zu verarbeiten ist. Eine wesentliche Anforderung an solche Sprachen ist die Entscheidbarkeit von Erfüllbarkeit der zugehörigen logischen Formeln. Man will schließlich aus dem schon vorhandenen Wissen neues Wissen generieren. Das Problem der ,,logischen Konsequenz“ muß also entscheidbar sein, was bekanntlich auf das Problem der Erfüllbarkeit zurückzuführen ist. In der Praxis will man eine höchstens polynomiale Komplexität hinnehmen, was leider oft nicht der Fall ist. Terminologische KR-Systeme können benutzt werden, um das Wissen über einen bestimmten Anwendungsbereich strukturiert darzustellen. Konzeptsprachen verwenden *Konzepte*, die aus atomaren Konzepten und Rollen aufgebaut werden. Diese Elemente können durch verschiedene Operatoren (auch *Konstruktoren* genannt) verbunden werden. Verschiedene Konzeptsprachen unterscheiden sich genau durch die Konstruktoren, die in ihnen verwendet werden können. Mit einer gegebenen Grundmenge werden Konzepte als Untermenge dieser Grundmenge, und Rollen als zweistellige Relationen auf ihr interpretiert. Für diese Arbeit dient die Konzeptsprache $\mathcal{ALC}$ als eine Grundlage.

2.1 Konzeptsprache $\mathcal{ALC}$

$\mathcal{ALC}$ steht für *Attributive Language with Complement*. Die **Syntax** von $\mathcal{ALC}$ wird wie folgt definiert:

- Alle atomaren Konzepte A sind Konzepte,
- Die Symbole $\top$ und $\bot$ sind Konzepte,

Wenn C und D Konzepte sind, sowie R eine Rolle, dann sind auch folgende Konstrukte Konzepte:

- $C \sqcap D$
- $C \sqcup D$
- $\exists R.C$
- $\forall R.C$
- $\neg C$

Die **Semantik** von $\mathcal{ALC}$ ist folgende:
Eine Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \varphi)$ besteht aus einer Menge $\Delta^{\mathcal{I}}$ und einer Interpretationsfunktion φ, die jedem Konzept eine Teilmenge von $\Delta^{\mathcal{I}}$, und jeder Rolle eine Teilmenge von $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ zuordnet. Dabei schreiben wir anstatt $\varphi(C)$ bzw. $\varphi(R)$ $C^{\mathcal{I}}$ bzw. $R^{\mathcal{I}}$. Weiterhin gilt:

- $\bot^{\mathcal{I}} = \emptyset$
- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C \sqcap D) = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D) = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b : \ (a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$
- $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b : \ (a,b) \in R^{\mathcal{I}} \wedge \ b \in C^{\mathcal{I}}\}$

Prädikatenlogisch entsprechen Konzepte einstelligen, und Rollen zweistelligen Prädikaten. Eine *Konzeptbeschreibung* (Konzeptdefinition) wird durch Anwendung der Konstruktoren auf Konzept- bzw. Rollennamen gebildet. Z. B. ist

$$\text{Weiblich} \sqcap \exists \text{hat-kind.Mensch}$$

eine Konzeptbeschreibung. Ein terminologisches Axiom ist von der Form $A = D$, wobei A ein Konzeptname und D eine Konzeptbeschreibung ist. Zum Beispiel ist

$$\text{Mutter} = \text{Weiblich} \sqcap \exists \text{hat-kind.Mensch}$$

ein solches Axiom. Eine terminologische Box (Tbox) ist eine endliche Menge von terminologischen Axiomen.
Ein Axiom $A = D$ wird von einer Interpretation $\mathcal{I}$ erfüllt, falls $A^{\mathcal{I}} = D^{\mathcal{I}}$ gilt. Eine Interpretation $\mathcal{I}$ erfüllt eine Tbox, wenn sie alle Axiome der Tbox erfüllt. Eine Interpretation $\mathcal{I}$ heißt ein Modell für die Tbox T, falls T von $\mathcal{I}$ erfüllt wird.

Konzeptnamen, die in keiner linken Axiomseite vorkommen, heißen *primitiv*, und alle andere Konzeptnamen heißen *definierte* Konzeptnamen. Eine Tbox enthält eine Mehrfachdefinition, wenn sie zwei Axiome der Form $A = D$ und $A = D'$ enthält.

Ein Konzeptname A **verwendet** den Namen B, wenn B unmittelbar in der Beschreibung von A vorkommt. Ein Axiom $A = D$ heißt zyklisch, wenn (A, A) in der transitiven Hülle der **Verwendet**-Relation liegt, d.h. wenn A direkt oder indirekt in seiner Beschreibung verwendet wird. Zum Beispiel:

$$\begin{array}{lcll} \text{Männlich} & = & \text{Mensch} & \sqcap \neg\text{Weiblich} \\ \text{Mensch} & = & \text{Männlich} & \sqcup \text{Weiblich} \end{array}$$

In diesem Beispiel verwendet der Konzeptname Männlich den Namen Mensch, der wiederum den Konzeptnamen Männlich verwendet. Somit enthält diese Tbox den Zyklus Männlich-Mensch-Männlich.
Ein Konzept C ist erfüllbar genau dann, wenn es eine Interpretation $\mathcal{I}$ gibt, so daß $C^{\mathcal{I}} \neq \emptyset$ gilt ($\mathcal{I}$ erfüllt C).
Um Erfüllbarkeit eines Konzeptes zu testen, geht man von aufgefalteten Terminologien ($\widehat{T}$) aus, wobei Zyklen und Mehrfachdefinitionen ausgeschlossen werden. Der Auffaltungsprozess (Makro Expansion) läuft folgendermaßen: Man ersetzt alle Namen auf der rechten Seite der Axiome durch ihre Beschreibungen und iteriert dieses Verfahren so lange, bis auf den rechten Axiomseiten keine definierten Namen mehr vorkommen. Zur Tbox T bezeichnen wir eine so aufgefaltete Tbox mit $\widehat{T}$. Zum Beispiel:

T :

$$\begin{array}{lcl} \text{Männlich} & = & \neg\text{Weiblich} \\ \text{Mutter} & = & \text{Weiblich} \sqcap \exists \text{ hat-kind.Mensch} \\ \text{Mensch} & = & \text{Weiblich} \sqcup \text{Männlich} \end{array}$$

$\widehat{T}$:

$$\begin{array}{lcl} \text{Männlich} & = & \neg\text{Weiblich} \\ \text{Mutter} & = & \text{Weiblich} \sqcap \exists\text{hat-kind.(Weiblich} \sqcup \text{Männlich)} \\ & = & \text{Weiblich} \sqcap \exists\text{hat-kind.(Weiblich} \sqcup \neg\text{Weiblich)} \\ \text{Mensch} & = & \text{Weiblich} \sqcup \neg\text{Weiblich} \end{array}$$

Entsprechend erhalten wir zu einer gegebenen Tbox T und einem Konzeptterm C den expandierten Konzeptterm $\widehat{C}$, indem wir alle definierten Namen, die in C vorkommen, durch ihre Definitionen so lange ersetzen, bis in C keine definierten Namen mehr vorkommen.
Unter der Annahme, daß die Tbox T_0 nicht zyklisch ist, kann man zeigen, daß der Auffaltungsprozess terminiert[1].

[1] Für Beweis siehe [10].

Der Auffaltungsprozess beeinflußt die Semantik der Tbox nicht. Man kann nämlich unter der Voraussetzung, daß die Tbox keinen Zyklus und keine Mehrfachdefinition enthält, zeigen, daß die beiden Tboxen T und $\widehat{T}$ die gleichen Modelle besitzen. Außerdem kann man sich, nach der Suche eines Modells für eine Tbox, auf die Interpretation der primitiven Namen beschränken, denn eine solche Interpretation kann stets auf eindeutige Weise auf ein Modell für die Tbox erweitert werden.

Es ist zu beachten, daß die Größe der expandierten Konzeptterme exponentiell von der Größe der ursprünglichen Tbox abhängen kann. Zum Erfüllbarkeitstest muß man außerdem die Konzeptterme in Negation-Normalform (**NNF**) umwandeln. In NNF tritt Negation nur vor Konzeptnamen auf. Bei den expandierten Konzepttermen in NNF taucht dann die Negation nur unmittelbar vor den primitiven Namen auf. Die Umformungsregeln sind folgende:

- $\neg\top \rightarrow \bot$
- $\neg\bot \rightarrow \top$
- $\neg\neg C \rightarrow C$
- $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \rightarrow \neg C \sqcap \neg D$
- $\neg(\exists R.C) \rightarrow \forall R.\neg C$
- $\neg(\forall R.C) \rightarrow \exists R.\neg C$

Korrektheit und Linearität dieser Umformung sind offensichtlich.

2.2 Entscheidbarkeit

Die Entscheidbarkeit der Erfüllbarkeit von $\mathcal{ALC}$-Konzepttermen wurde in der Literatur durch verschiedene Ansätze gezeigt. Zum Beispiel wurde in [11] gezeigt, daß $\mathcal{ALC}$ eine syntaktische Variante der Multimodallogik K_n ist. Das heißt, daß jede $\mathcal{ALC}$-Konzeptterm C kann in eine K_n-Formel φ_C umgeformt werden, für die gilt: C ist genau dann erfüllbar, wenn φ_C erfüllbar ist. Das Erfüllbarkeitsproblem ist für K_n bekanntlich entscheidbar.

Borgida hat in [12] gezeigt, daß jeder $\mathcal{ALC}$-Konzeptterm in eine L_2-Formel übersetzt werden kann. Die Sprache L_2 ist eine Unterklasse der Prädikatlogik erster Stufe, welche neben der Gleichheit nur Prädikat- und Konstantensymbole (aber keine Funktionssymbole) unter Verwendung nur zwei logischen Variablen x, y

zuläßt. Die Entscheidbarkeit von L_2 wurde in [13] nachgewiesen. Diese Methoden sind aber für praktische Anwendung sehr aufwendig[2]. Deshalb wurden für $\mathcal{ALC}$ praktisch anwendbare Entscheidungsalgorithmen entwickelt. Sie sind oft regelbasierte Entscheidungsverfahren. Ein solches Verfahren wird im folgenden Abschnitt beschrieben. Die zugehörige Sätze lassen sich aus den Resultaten von Schmidt-Schauß und Smolka (siehe [14]) ableiten und sind die Grundlage für die Korrektheit und die Vollständigkeit des Algorithmus 2.7.

2.3 Ein regelbasierter Erfüllbarkeitstest

$\mathcal{ALC}$- Konzepte können unerfüllbar sein, d.h. es ist möglich, daß es keine Interpretation für ein Konzept gibt. So ist z. B. $C \sqcap \neg C$ offensichtlich unerfüllbar. Eine Idee für einen Erfüllbarkeitstest ist es, ein Konzept in seine Bestandteile zu zerlegen, um dann Widersprüche zu finden. Für ein Konzept $C \sqcap D$ muß also überprüft werden, ob sowohl C als auch D erfüllbar sind. Der Test arbeitet mit sogenannten *Constraints*, die aus Variablen, Konzepten und Rollen bestehen. Es wird ein Alphabet mit Variablen (z.B. x, y ...) vorausgesetzt. Ein *Constraint* hat folgende Syntax:

$$x : C \text{ oder } (x, y) : R$$

wobei C für ein Konzept, R für eine Rolle und x, y für Variablen stehen. Bezüglich des Constraints $(x, y) : R$ nennt man x den Rollenvorgänger von y. Entsprechend heißt y der Rollennachfolger von x. Bezüglich $x : C$ nennt man x eine Instanz von C.

Die Variablen sind Platzhalter für Elemente aus $\Delta^{\mathcal{I}}$, sogenannte *Individuen*. Es sei $\mathcal{I}$ eine Interpretation. Eine $\mathcal{I}-$Zuordnung α ist eine Funktion, die jeder Variable ein Individuum zuweist. Es heißt, α erfüllt $x : C$ wenn $\alpha(x) \in C^{\mathcal{I}}$, und α erfüllt xRy wenn $(\alpha(x), \alpha(y)) \in R^{\mathcal{I}}$. Ein *Constraint* c heißt erfüllbar, wenn es eine Interpretation $\mathcal{I}$ und eine $\mathcal{I}$ -Zuordnung α gibt, so daß c von α erfüllt wird. Eine $\mathcal{I}$ -Zuordnung erfüllt eine Constraintmenge S, wenn α jedes *Constraint* c $\in$ S erfüllt. Ein *Constraintsystem* $\mathcal{S}$ ist eine endliche Menge von Constraintmengen. Ein Constraintsystem $\mathcal{S}$ ist erfüllbar genau dann, wenn mindestens eine Constraintmenge $S \in \mathcal{S}$ erfüllbar ist.

Im folgenden wird ein regelbasierter Algorithmus für die Erfüllbarkeitsüberprüfung der $\mathcal{ALC}$-Konzepte vorgestellt. Auf Grund des $\sqcup$-Konstruktors in $\mathcal{ALC}$ arbeitet man mit Mengen von Constraintmengen. Wir gehen immer von einer aufgefalteten Tbox bzw. expandierten Konzepttermen in NNF aus. Dabei nehmen wir an, daß Tboxen keine zyklischen bzw. Mehrfachdefinitionen enthalten.

[2] Zum Beispiel ist die Komplexität von L_2 höher als von $\mathcal{ALC}$.

$\sqcap$-Regel :
Wenn $(x : C_1 \sqcap C_2) \in S$ und $(x : C_1) \notin S$ oder $(x : C_2) \notin S$, dann ersetze S in $\mathcal{S}$ durch $S' = \{x : C_1, x : C_2\} \cup S$

$\sqcup$-Regel :
Wenn $(x : C_1 \sqcup C_2) \in S$ und $(x : C_1) \notin S$ und $(x : C_2) \notin S$, dann ersetze S in $\mathcal{S}$ durch $\{S', S''\}$, wobei $S' = S \cup \{x : C_1\}$ und $S'' = S \cup \{x : C_2\}$ sind.

$\exists$-Regel :
Wenn $(x : \exists R.C) \in S$ und es gibt kein Individuum y mit $(y : C) \in S$ und $(x, y) : R$, dann ersetze S in $\mathcal{S}$ durch $S' = S \cup \{(x, y) : R, y : C\}$, wobei y eine neue Variable ist, d.h. in S nicht vorkommt.

$\forall$-Regel :
Wenn $(x : \forall R.C) \in S$ und es existiert ein Individuum y mit $(x, y) : R \in S$, $(y : C) \notin S$, dann ersetze S in $\mathcal{S}$ durch $S' = S \cup \{y : C\}$

Abbildung 2.1: Vervollständigungsregeln für $\mathcal{ALC}$

Sei $\mathcal{S} = \{S_0, S_1, S_2, \ldots, S_n\}$ eine endliche Menge von Constraintmengen. Es werden dann die sogenannten Vervollständigungsregeln definiert, die auf ein Element, also eine Constraintmenge $S \in \mathcal{S}$, angewendet werden (siehe Abbildung 2.1).
Durch Anwendung dieser Regeln, die zur Erzeugung neuer Individuen führen können, versucht man entweder versteckte Widersprüche aufzufinden oder ein Modell für den Konzeptterm zu konstruieren. Zum Beispiel kann der Konzeptterm $x : \exists R.C$ nur dann erfüllt sein, wenn eine Instanz von C mit x in R-Beziehung steht. Deshalb fügt man eine solche Instanz-Beziehung $y : C$ hinzu und testet weiter.

Satz 2.1
Ein Konzeptterm C ist erfüllbar gdw. das Constraint $x : C$ erfüllbar ist.

Satz 2.2
Die oben genannten Regeln sind konsistenzerhaltend, das heißt :

- *Wenn S', S'' durch die Anwendung der $\sqcup$-Regel aus S gewonnen werden, dann gilt:*
$$S \text{ erfüllbar} \Leftrightarrow S' \text{ erfüllbar oder } S'' \text{ erfüllbar}$$
- *Für die übrigen Regeln gilt:*
$$S \text{ erfüllbar} \Leftrightarrow S' \text{ erfüllbar}$$

Eine Constraintmenge S ist vollständig, wenn keine Regel mehr darauf anwendbar ist. Ein Constraintsystem $\mathcal{S}$ heißt vollständig, wenn alle $S \in \mathcal{S}$ vollständig sind. Eine Constraintmenge S enthält einen offensichtlichen Widerspruch (im Englischen *clash*), wenn es eine Variable x gibt mit

$$\{x : \bot\} \subseteq S \quad \text{oder für einen Konzeptnamen } A \text{ gilt: } \{x : A, x : \neg A\} \subseteq S.$$

Man kann zeigen, daß eine vollständige Constraintmenge ohne offensichtliche Widersprüche erfüllbar ist. Dazu definiert man zuerst ein kanonisches Modell zu einer solchen Menge:

Definition 2.3
Sei S eine vollständige Constraintmenge ohne offensichtliche Widersprüche. Die Menge aller Variablen x, die in S vorkommen, wird mit X_s bezeichnet. Die Interpretation $\mathcal{I}_s$ wird wie folgt definiert:

- $\Delta_{\mathcal{I}_s} = X_s$.
- $x^{\mathcal{I}_s} := x$.
- $R^{\mathcal{I}_s} := \{(x, y) \mid (x, y) : R \in S\}$, *für alle Rollennamen, die in S vorkommen.*
- $A^{\mathcal{I}_s} := \{x \mid x : A \in S\}$, *für alle Konzeptnamen, die in S vorkommen.*

Satz 2.4
Sei S eine vollständige Constraintmenge ohne offensichtliche Widersprüche. Dann ist $\mathcal{I}_s$ ein Modell für S.

Beweis
Zum Beweis ist zu zeigen, daß $\mathcal{I}_s$ jedes Constraint aus S erfüllt.

- $(x, y) : R \in S$ ist erfüllt, denn nach Definition gilt $(x, y) \in R^{\mathcal{I}_s}$.
- $x : C$
  Man muß zeigen, daß $x^{\mathcal{I}_s} \in C^{\mathcal{I}_s}$ gilt. Diese Behauptung kann durch Induktion über den Aufbau von $\mathcal{ALC}$-Konzepttermen bewiesen werden:
  - $x : A$ und A ein primitiver Konzeptname
    Die Behauptung gilt, denn nach Definition gilt $x \in A^{\mathcal{I}_s}$.
  - $x : \neg A$ und A ein Konzeptname
    Da S keinen offensichtlichen Widerspruch enthält, gilt $x : A \notin S$. Nach Definition folgt dann $x \notin A^{\mathcal{I}_s}$, also $x \in (\neg A)^{\mathcal{I}_s}$. Man muß beachten, daß wegen **NNF** Negation nur unmittelbar vor den primitiven Namen auftritt.

- $x : C_1 \sqcap C_2$
  Da S vollständig ist, ist die $\sqcap$-Regel nicht auf S anwendbar. Es gilt also $\{x : C_1, x : C_2\} \subseteq S$. Nach Induktionsvoraussetzung folgt dann $x \in C_1^{\mathcal{I}_s}$ und $x \in C_2^{\mathcal{I}_s}$, also $x \in (C_1 \sqcap C_2)^{\mathcal{I}_s}$.
- $x : C_1 \sqcup C_2$
  Der Beweis ist Analog zur $\sqcap$-Regel.
- $x : \exists R.D$
  Da S vollständig ist, gibt es ein $y \in X$ mit $\{(x, y) : R,\ y : D\} \subseteq S$. Aus $(x, y) : R \in S$ folgt dann $(x, y) \in R^{\mathcal{I}_s}$ und aus $y : D \in S$ folgt nach Induktionsvoraussetzung $y \in D^{\mathcal{I}_s}$, also $x \in (\exists R.D)^{\mathcal{I}_s}$.
- $x : \forall R.D$
  Sei $(x, y) \in R^{\mathcal{I}_s}$ beliebig. Nach Definition von $R^{\mathcal{I}_s}$ folgt $(x, y) : R \in S$. Da S vollständig ist, ist die $\forall$-Regel nicht auf $\{x : \forall R.D,\ (x, y) : R\} \subseteq S$ anwendbar, d,h, es ist $y : D \in S$. Somit folgt nach Induktionsvoraussetzung $y \in D^{\mathcal{I}_s}$. Da y mit $(x, y) \in R^{\mathcal{I}_s}$ beliebig gewählt war, folgt $x \in (\forall R.D)^{\mathcal{I}_s}$.

□

Satz 2.5 (Terminierung)
Wenn C_0 ein $\mathcal{ALC}$-Konzept in NNF ist, so wird aus dem Constraintsystem $\mathcal{S}_0 = \{\{x_0 : C_0\}\}$ nach einer endlichen Anzahl von Regelanwendungen ein vollständiges Constraintsystem $\mathcal{S}_n$ gewonnen.

Auf Grund der obigen Sätze, insbesondere der Sätze 2.2 und 2.4, ist leicht einzusehen, daß der Konzeptterm C_0 genau dann unerfüllbar ist, wenn alle Constraintmengen aus $\mathcal{S}_n$ offensichtliche Widersprüche enthalten. Dies ist die Aussage des nächsten Satzes:

Satz 2.6
Sei C_0 ein $\mathcal{ALC}$-Konzeptterm in NNF und $\mathcal{S}_n$ das vollständige Constraintsystem, das aus dem Constraintsystem $\{\{x :\ C_0\}\}$ abgeleitet wurde. Es gilt dann:

$$C_0 \text{ ist erfüllbar} \Leftrightarrow \mathcal{S}_n \text{ erfüllbar.}$$

Nach den Sätzen 2.6 und 2.5 können die Regeln als eine Entscheidungsprozedur für die (Un-)Erfüllbarkeit eines $\mathcal{ALC}$-Konzepts verwendet werden, indem die aus $\mathcal{S}_0$ abgeleiteten Constraintmengen auf offensichtliche Widersprüche überprüft werden.

Algorithmus 2.7
Es sei C_0 ein $\mathcal{ALC}$-Konzeptterm in NNF, dessen Erfüllbarkeit getestet werden soll.

- *Beginnend mit Constraintsystem* $\mathcal{S}_0 = \{\{x_0 : C_0\}\}$*, wende die Regeln solange an, bis ein vollständiges Constraintsystem* $\mathcal{S}_n$ *erreicht ist:*

$$\mathcal{S}_0 \rightarrow \mathcal{S}_1 \rightarrow \mathcal{S}_2 \rightarrow \ldots \rightarrow \mathcal{S}_n$$

- *Enthält jede Constraintmenge* $S \in \mathcal{S}_n$ *einen offensichtlichen Widerspruch, so ist* C_0 *unerfüllbar, sonst erfüllbar*

Beispiel 2.8

$\mathcal{S}_0 = \{S_0\} = \{\{x : \exists R.D \sqcap \forall R.\bot\}\} = \{\{x : C_0\}\}$
$\mathcal{S}_0 \rightarrow_{\sqcap} \mathcal{S}_1 = \{S_0 \cup \{x : \exists R.D,\ x : \forall R.\bot\}\} = \{S_1\}$
$\mathcal{S}_1 \rightarrow_{\exists} \mathcal{S}_2 = \{S_1 \cup \{(x, y) : R,\ y : D\}\} = \{S_2\}$
$\mathcal{S}_2 \rightarrow_{\forall} \mathcal{S}_3 = \{S_2 \cup \{y : \bot\}\} = \{S_3\}$

$\mathcal{S}_3$ ist vollständig und enthält keine erfüllbare Constraintmenge, somit ist C_0 unerfüllbar.
Es ist zu beachten, daß durch das Zusammenspiel von $\exists$- und $\forall$-Regel die Anzahl der eingeführten Variablen exponentiell zur Größe von C_0 wachsen kann. Zum Beispiel enthält das vollständige Constraintsystem zu dem folgenden Konzeptterm mindestens $2^n + 1$ Variablen:

Beispiel 2.9

$\exists R.C_{11} \sqcap \exists R.C_{12} \sqcap \forall R.(\ldots(\exists R.C_{n1} \sqcap \exists R.C_{n2} \sqcap \forall R.D)\ldots))$

2.4 Komplexität des Erfüllbarkeitsproblems

Die Komplexität des Erfüllbarkeitstests für terminologische Sprachen ist der Gegendstand vieler Forschungsarbeiten gewesen. Dieses Problem wurde für verschiedene Konzeptsprachen untersucht. Die Ergebnisse dieser Untersuchungen haben gezeigt, daß der Erfüllbarkeitstest ein schwer zu bearbeitendes Problem ist [14, 15, 16]. Schmidt-Schauß und Smolka haben gezeigt, daß der Erfüllbarkeitstest für $\mathcal{ALE}$-Konzepte (d.h. $\mathcal{ALC}$ ohne $\sqcup$ und mit eingeschränkter Negation) ein NP-vollständiges Problem ist [14]. Sie haben u.a. gezeigt, daß man die Erfüllbarkeit jedes $\mathcal{ALC}$-Konzepttermes mit linearem (in der Größe des Termes) Speicher entscheiden kann. Somit liegt das Erfüllbarkeitsproblem für $\mathcal{ALC}$ in PSPACE.
Anschließend wurde die PSPACE-Härte des Erfüllbarkeitsproblems für $\mathcal{ALC}$ bewiesen. Dazu wurde das Gültigkeitsproblem für quantifizierte Boole'sche Formeln auf das Erfüllbarkeitsproblem für $\mathcal{ALC}$ reduziert. Bekanntlich ist das Gültigkeitsproblem für quantifizierte Boole'sche Formeln ein PSPACE-vollständiges Problem (siehe [17]). Da das Erfüllbarkeitsproblem für $\mathcal{ALC}$ in PSPACE liegt und außerdem PSPACE-hart ist, folgt:

Das Erfüllbarkeitsproblem für $\mathcal{ALC}$ ist ein PSPACE-vollständiges Problem.

Einen anderen Ansatz für den Beweis, daß das Erfüllbarkeitsproblem für $\mathcal{ALC}$ in PSPACE liegt, liefert ein optimierter Erfüllbarkeitsalgorithmus. Dieser *funktionale* Algorithmus wird im nächsten Unterabschnitt beschrieben und diskutiert.

2.4.1 Eine Optimierung des Erfüllbarkeitstestes

In [18] wurde eine optimierte Version des Algorithmus 2.7 beschrieben. Die Idee dabei ist, den Algorithmus so zu gestalten, daß eine explizite Repräsentation von Individuen und ihren Rollenbeziehungen nicht mehr notwendig ist. Man braucht eine solche explizite Darstellung, um zum einen Widersprüche in Constraintmengen zu entdecken (sie haben ja die Form: $x : A, x : \neg A$) und zum anderen festzustellen, welche neuen Constraints sich aus den Constraints von der Form $x : \forall R.D$ ergeben. Wenn man ein solches Constraint hat, müßte man alle Individuen y betrachten, die mit x in R-Beziehung stehen, d.h. diejenigen y mit $(x, y) : R$, und dann müßte man überprüfen, ob diese Individuen Instanzen von D sind, d.h. , ob $y : D$ erfüllt ist. Um auf diese anscheinend notwendige Repräsentation verzichten zu können, überlegt man sich zuerst, woraus die Constraints bzgl. eines gegebenen Individuums x stammen können:

- Die Anwendung der $\sqcap$-Regel oder $\sqcup$-Regel auf x kann dazu führen, daß neue Constraints für x gebildet werden. Zum Beispiel:

$$\{\underline{x : C_1 \sqcap C_2},\ x : C_1\} \rightarrow_{\sqcap} \{x : C_1 \sqcap C_2, x : C_1,\ \underline{x : C_2}\}$$

- Die Anwendung der $\exists$-Regel auf ein anderes Individuum y kann dazu führen, daß ein neues Constraint für x gebildet wird. Zum Beispiel:

$$\{\underline{y : \exists R.C}\} \rightarrow_{\exists} \{y : \exists R.C,\ \underline{x : C},\ (y, x) : R\}$$

Es ist zu beachten, daß im zweiten Fall x der Rollennachfolger von y bzgl. der Rolle R ist. Für jedes Individuum $x \neq x_0$ gibt es genau einen Rollenvorgänger y. Das allererste Individuum x_0 hat keinen Rollenvorgänger. Dies bedeutet aber, daß alle Constraints an x_0 durch Anwendung der $\sqcap$- bzw. $\sqcup$-Regel auf x_0 gebildet werden können. Wir können also alle potentiellen Widersprüche, die mit x_0 verbunden sind, entdecken, indem wir so lange wie möglich $\sqcap$- bzw. $\sqcup$-Regel auf x_0 anwenden. Wenn auf x_0-Constraints keine $\sqcap$- oder $\sqcup$-Regeln mehr anwendbar sind, enthält das Constraintsystem alle möglichen Existenz- bzw. Werterestriktionen für x_0. Somit kann festgestellt werden, wieviele neue Individuen als Rollennachfolger von x_0 eingeführt werden müssen:

$$x_0 : \exists R.C \rightarrow (x_0, y) : R,\ y : C$$

Es kann außerdem festgestellt werden, welche neuen Constraints von x_0 aus propagiert werden:

$$\begin{array}{lll} \{x_0 : \exists R.C,\ x_0 : \forall R.D\} & \rightarrow_{\exists} & \{x_0 : \exists R.C,\ x_0 : \forall R.D,\ x_1 : C,\ (x_0, x_1) : R\} \\ & \rightarrow_{\forall} & \{\ldots, \underline{x_1 : D}\} \end{array}$$

Wenn man alle $\sqcap$- und $\sqcup$-Regeln auf x_0 angewendet und bis dahin keine offensichtlichen Widersprüche entdeckt hat, erreicht man die folgende Situation:

$$x_0 : \begin{cases} L_1,\ L_2,\ \ldots,\ L_q, \\ \exists R_1.C_{11},\ \exists R_1.C_{12},\ \ldots,\ \exists R_1.C_{1m_1},\ \forall R_1.D_{11},\ \ldots,\ \forall R_1.D_{1n_1} \\ \vdots \\ \exists R_k.C_{k1},\ \exists R_k.C_{k2},\ \ldots,\ \exists R_k.C_{km_k},\ \forall R_k.D_{k1},\ \ldots,\ \forall R_k.D_{kn_k} \end{cases}$$

dabei sind $L_1, \ldots, L_q$ Konzeptnamen oder ihre Negationen (im folgenden nennen wir sie Literale). Nun kann man schließen, daß $x_0 : C_0$ genau dann erfüllbar ist, wenn alle folgenden Constraints (einzeln, d.h. unabhängig voneinander) erfüllbar sind:

$$\begin{array}{|lcl} x_{11} & : & C_{11} \sqcap D_{11} \sqcap D_{12} \sqcap \cdots \sqcap D_{1n_1} \\ \vdots & & \\ x_{1m_1} & : & C_{1m_1} \sqcap D_{11} \sqcap D_{12} \sqcap \cdots \sqcap D_{1n_1} \\ \vdots & & \\ x_{k1} & : & C_{k1} \sqcap D_{k1} \sqcap D_{k2} \sqcap \cdots \sqcap D_{kn_k} \\ \vdots & & \\ x_{km_k} & : & C_{km_k} \sqcap D_{k1} \sqcap D_{k2} \sqcap \cdots \sqcap D_{kn_k} \end{array}$$

Man kann einsehen, daß hier die explizite Anwendung der $\exists$- bzw. $\forall$-Regeln nicht mehr notwendig sind. Dafür muß nur die Erfüllbarkeit der obigen Constraints getestet werden (diese Tests sind unabhängig voneinander). Insbesondere sind alle Constraints von der Form $x : \forall R.D$ völlig irrelevant, falls sich im Constraintsystem keine Constraints von der Form $x : \exists R.C$ befinden. Aus diesen Überlegungen ergibt sich ein *funktionaler* Algorithmus (siehe Abbildung 2.2), der viel leichter zu implementieren ist. Der Algorithmus nimmt als Eingabe eine Menge von Constraints und gibt *true* aus, falls diese Menge erfüllbar ist, sonst wird der Wert *false* ausgegeben.
Ein Konzeptterm C ist genau dann erfüllbar, wenn der Aufruf Sat($\{C\}$) **true** liefert.
Terminierung, Korrektheit und Vollständigkeit dieses Algorithmus wurde in der Literatur diskutiert und bewiesen (siehe [19] Abschnitt 4.3).

Bemerkung 2.11
Der funktionale Algorithmus liegt in PSPACE. Dies ist intuitiv klar, denn man kann sich den Gesamtverlauf des Algorithmus als einen Baum vorstellen, dessen Wurzel mit dem Eingabeterm C_0 (als Argument des ersten Aufrufs) beschriftet ist. Die anderen Knoten sind mit den Argumenten der rekursiven Sat-Aufrufe beschriftet. Die Pfade des Baumes werden unabhängig voneinander durchlaufen. Dabei ist die Rekursionstiefe sowie die Größe der Argumente je Sat-Aufruf linear beschränkt durch die Größe des Eingabetermes, denn: Die Argumente der

Algorithmus 2.10
$Sat(\mathcal{C})$ =
if $A \in \mathcal{C}$ *and* $\neg A \in \mathcal{C}$
then false
else if $C_1 \sqcap C_2 \in \mathcal{C}$
then $Sat(\{C_1, C_2\} \cup \mathcal{C} \setminus \{C_1 \sqcap C_2\})$
else if $C_1 \sqcup C_2 \in \mathcal{C}$
then $Sat(\{C_1\} \cup \mathcal{C} \setminus \{C_1 \sqcup C_2\})$ *or* $Sat(\{C_2\} \cup \mathcal{C} \setminus \{C_1 \sqcup C_2\})$
else if for all $\exists R.C \in \mathcal{C}$
$Sat(\{C\} \cup \{D \mid \forall R.D \in \mathcal{C}\})$
then true
else false

Abbildung 2.2: Funktionaler Erfüllbarkeitsalgorithmus für $\mathcal{ALC}$.

Sat-Aufrufe sind Mengen von Untertermen des Eingabeterms und damit linear in der Größe von C_0. Bei jedem Rekursionsschritt wird die Anzahl von Konzeptkonstruktoren in der Argumentmenge echt kleiner. Diese Anzahl ist ebenfalls linear durch die Größe des Eingabeterms beschränkt. Somit kommt man mit polynomialem Platz aus.

Kapitel 3

Das Subsumtionsproblem

3.1 Subsumtion

Das Folgern in Konzeptsprachen basiert auf **Subsumtion**. Das Konzept C wird vom Konzept D subsumiert ($C \sqsubseteq D$), wenn C in jeder Interpretation eine Teilmenge der durch D beschriebenen Menge ist, d.h., wenn $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ für jede Interpretation $\mathcal{I}$ gilt. Subsumtion definiert automatisch eine Hierarchie unter Konzepten. Die Berechnung dieser Hierarchie ist eine der wichtigsten Aufgaben bzgl. terminologischer Sprachen und ist häufig als ein Systemdienst implementiert. Prädikatenlogisch entspricht die Subsumtion der logischen Implikation:

$$C \sqsubseteq D \quad \text{gdw.} \quad \forall x : \; C(x) \rightarrow D(x).$$

Dabei ist x eine Variable und C, D sind einstellige Prädikate.
Subsumtionsberechnung ermöglicht es festzustellen, welche Eigenschaften aus anderen Eigenschaften folgen (wenn man in einer Tbox definierte Konzeptnamen als Eigenschaften betrachtet) und genau dies ist ein zentrales Problem der Wissensverarbeitung. Formal wird Subsumtion wie folgt definiert:

Definition 3.1
Seien C und D Konzepte und T eine Tbox:

- *C wird von D subsumiert ($C \sqsubseteq D$), wenn für alle Interpretationen $\mathcal{I}$ die Inklusionsbeziehung $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ gilt.*
- *C wird von D bzgl. der Tbox T subsumiert ($C \sqsubseteq_T D$), wenn für alle Modelle $\mathcal{I}$ von T die Inklusionsbeziehung $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ gilt.*

Es ist zu beachten, daß die allgemeine Subsumtionsdefinition der Definition bzgl. der leeren Tbox entspricht, da die leere Tbox von jeder Interpretation erfüllt wird. Es gilt also:

$$C \sqsubseteq D \Leftrightarrow C \sqsubseteq_{\emptyset} D$$

Für Subsumtion gelten folgende Aussagen[1]:

- Der Subsumtionstest kann auf den Erfüllbarkeitstest zurückgeführt werden, wenn die zugrunde liegende Konzeptsprache uneingeschränkte Negation (Negation auch vor den Konzepttermen) zuläßt. Es gilt dann nämlich:

$$C \sqsubseteq D \Leftrightarrow C \sqcap \neg D \text{ ist unerfüllbar.}$$

- Subsumtion bzgl. einer Tbox kann auf Subsumtion bzgl. der leeren Tbox reduziert werden. Es gilt nämlich:

$$C \sqsubseteq_T D \Leftrightarrow \widehat{C} \sqsubseteq_\emptyset \widehat{D}$$

  Dabei sind $\widehat{C}$ bzw. $\widehat{D}$ die zu C und D expandierten Konzepte (siehe Abschnitt 2.1).

3.2 Komplexität der Subsumtion

Der Subsumtionstest ist für Konzeptsprachen, die das $\bot$-Symbol zulassen, nicht leichter als der Erfüllbarkeitstest, denn das Konzept C ist unerfüllbar genau dann, wenn es von $\bot$ subsumiert wird. Somit ist der Erfüllbarkeitstest ein Speziellfall der Subsumtion. Falls eine Konzeptsprache die uneingeschränkte Negation zuläßt, ist die Subsumtion nicht schwieriger als die Erfüllbarkeit, denn wegen

$$C \sqsubseteq D \Leftrightarrow C \sqcap \neg D \text{ ist unerfüllbar}$$

kann jeder Subsumtionstest auf einen Erfüllbarkeitstest zurückgeführt werden. Das Subsumtionsproblem ist daher für $\mathcal{ALC}$ genau so schwierig wie das Erfüllbarkeitsproblem.
Wie schon im Abschnitt 2.4 diskutiert wurde, haben Schmidt-Schauß und Smolka einen PSPACE-vollständigen Algorithmus angegeben, der Subsumtion (bzw. Erfüllbarkeit) in $\mathcal{ALC}$ mit linearem Speicheraufwand entscheiden kann.

Hier soll noch erwähnt werden, daß die Entscheidbarkeit der Subsumtion bzgl. einer Konzeptsprache von den zulässigen Konstruktoren abhängt. Zum Beispiel hat Schmidt-Schauß für eine Konzeptsprache, die nur Rollengleichheit, Konjunktion, Werterestriktion und Rollenkomposition zuläßt, die Unentscheidbarkeit der Subsumtion auf die Unentscheidbarkeit des Wortproblems für Gruppen zurückgeführt (siehe [20]).

[1] Diese sind in der Vorlesung „Logische Methoden der Wissensrepräsentation“, die von Prof. Franz Baader an der RWTH Achen im SS 1996 gehalten wurde, bewiesen.

Männlich	=	¬Weiblich		
Mensch	=	Männlich	⊔	Weiblich
Elternteil	=	Mensch	⊓	∃hat-kind.Mensch
Hatkeinetochter	=	Mensch	⊓	∀hat-kind.Männlich
Hatkeinensohn	=	Mensch	⊓	∀hat-kind.Weiblich
Hatkeinbaby	=	Mensch	⊓	∀hat-kind.¬Baby

Abbildung 3.1: Eine Familien-Terminologie

3.3 Klassifikation

Unter Klassifikation versteht man die Berechnung der Subsumtionshierarchie zwischen den in der Tbox definierten Konzeptnamen. Klassifikation wird meist als ein Systemdienst implementiert.

Empirische Untersuchungen bzgl. der Subsumtionsberechnung haben gezeigt [21], daß die Komplexität für die Berechnung einzelner Subsumtionen nicht das entscheidende Problem ist. Um die Klassifikation für n Konzeptnamen zu berechnen, muß man im „worst case“ n^2 Subsumtionen testen. Durch Auswahl einer geeigneten Reihenfolge der Aufrufe und Ausnutzung des schon berechneten Teils der Hierarchie kann man einen beträchtlichen Teil dieser Aufrufe vermeiden[2]. Solche Optimierungsmethoden sind in vielen Implementationen erfolgreich angewendet worden [22, 23, 24, 25].

Die Informationen, die aus der einfachen Subsumtionshierarchie gewonnen werden können, sind sehr eingeschränkt. Das folgende Beispiel[3] bzgl. einer Tbox beschrieben in Abbildung 3.1 soll diese Tatsache verdeutlichen:
Die Namen Baby und Weiblich sind primitive Namen, hat-kind ist ein Rollenname und alle anderen Namen sind definierte Namen. Hatkeinetochter steht für alle Menschen, die kein weibliches Kind haben, Hatkeinensohn steht für alle Menschen, die kein männliches Kind haben und entsprechend Hatkeinbaby für Menschen, die kein Baby haben. Es gilt die Subsumtion:

$$\text{Hatkeinetochter} \sqcap \text{Hatkeinensohn} \sqsubseteq \text{Hatkeinbaby}$$

Das heißt jeder Mensch, der keine Tochter und keinen Sohn hat, hat kein Baby. Zum Beispiel erzeugt der Algorithmus 2.7 zu der obigen Subsumtionsfrage eine Interpretation und ein Individuum d so, daß entweder d nicht gleichzeitig in den Interpretationsmengen von Hatkeinensohn und Hatkeinetochter liegt oder auch

[2] Für die Tboxen in [21] konnten zwischen 80% bis 90% dieser Aufrufe vermieden werden
[3] Dieses Beispiel ist in [9] beschrieben.

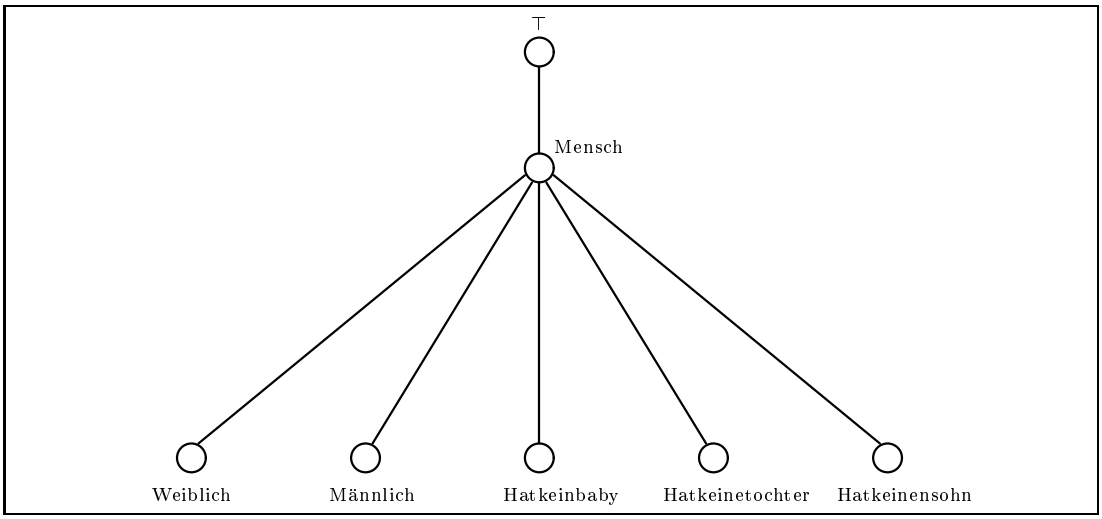

Abbildung 3.2: Klassifikation für die Tbox in Abbildung 3.1

in der Interpretationsmenge von Hatkeinbaby.

Diese logische Folgerung läßt sich aber aus der Subsumtionshierarchie, die mittels einfacher Klassifikation berechnet wird (Unterkonzept/Oberkonzept-Beziehung zwischen allen definierten Namen), nicht ableiten, weil es offensichtlich keine Subsumtionsrelation zwischen diesen definierten Konzeptnamen gibt (siehe Abbildung 3.2).

Das obige Beispiel zeigt, daß solche Schlußfolgerungen während der Laufzeit wesentlich vereinfacht werden können, wenn man die Hierarchie zwischen Konjunktionen von Konzeptnamen vorher berechnet hat, das heißt, eine erweiterte Subsumtionshierarchie, die nicht nur die definierten Konzeptnamen $A_1, A_2, \ldots, A_n$, sondern auch alle Konjunktionen zwischen definierten Konzeptnamen, d.h. Konzeptterme von der Form:

$$C_1 \sqcap C_2 \ \sqcap \ldots \sqcap \ C_k \text{ mit } C_j \in \{A_1, A_2, \ldots, A_n\} \text{ und } 1 \leq j \leq k, \ 1 \leq k \leq n$$

einschließt. Dabei besteht die Menge $A = \{A_1, A_2, \ldots, A_n\}$ aus allen in der Tbox definierten Namen. Im folgenden bezeichnen wir für eine Teilmenge

$$C = \{C_1, \ C_2, \ \ldots, \ C_k\} \subseteq A$$

die Konjunktion $C_1 \sqcap C_2 \ \sqcap \ldots \sqcap \ C_k$ mit $\sqcap C$.
Gewünscht ist also eine solche erweiterte Subsumtionshierarchie, die es ermöglicht, jede Subsumtionsfrage von der Form $\sqcap B \sqsubseteq \sqcap D$, leicht zu beantworten. Eine solche Hierarchie könnte z.B. durch ein *Precomputing* berechnet und als ein Systemdienst zur Verfügung gestellt werden.

Eigentlich ist die Berechnung einer erweiterten Hierarchie ein Speziellfall der Klassifikation. Man kann nämlich für jede Konjunktion einen neuen Namen einführen und dann die erweiterte Tbox klassifizieren. Diese Methode ist aber in der Praxis nicht anwendbar, denn die Tbox wächst exponentiell in der Größe der anfänglichen Tbox[4].

Es gibt also so viele Konjunktionen, daß man nicht für alle Subsumtionstests durchführen möchte. Für n Konzeptnamen müßte man $(2^n)^2$ mal den teuren Subsumtionsalgorithmus aufrufen, was offensichtlich kein vertretbarer Aufwand ist. Man versucht also, Methoden zu finden, die die Berechnung der erweiterten Subsumtionshierarchie mit einem besseren Aufwand ermöglichen. Die Methoden, die in dieser Arbeit untersucht werden, sind zwar auch im ,,worst case“ exponentiell, aber es besteht die Hoffnung, daß sich für typische Tboxen ein besserer Aufwand ergibt.

Das Ziel ist es also, eine minimale Repräsentation der Subsumtionshierarchie zwischen den Konjunktionen von definierten Konzeptnamen mit möglichst geringem Aufwand zu berechnen. Diese Repräsentation muß vollständig sein, d.h. alle Subsumtionen von der Form $\sqcap C \sqsubseteq \sqcap D$ müssen sich daraus mit einem akzeptablem Aufwand ableiten lassen. Außerdem sollte man für ihre Berechnung nicht zu oft den Subsumtionsalgorithmus aufrufen müssen.

Wie schon erwähnt wurde, zieht man hierfür die Resultate aus der Welt der ,,Formalen Begriffsanalyse“ heran. Im nächsten Kapitel führen wir die notwendigen Definitionen und Sätze aus dieser Umgebung ein, in der Hoffnung, daß bestimmte begriffsanalytische Verfahren zusammen mit einem modifizierten Subsumtionsalgorithmus zum Ziel führen können.

[4]Für eine Tbox mit n definierten Namen muß man größenordnungsmäßig 2^n neue Namen einführen.

Kapitel 4

Formale Begriffsanalyse

4.1 Begriffsverbände

Begriffsverbände sind das zentrale Untersuchungsobjekt der ,,Formalen Begriffsanalyse". Für ihre Bestimmung, Darstellung und Analyse sind vielfältige Methoden entwickelt worden. Begriffsverbände können für verschiedene Zwecke eingesetzt werden (siehe [26, 27]). Hierarchische Klassifikation von Gegenständen und Vorlagen zum Anordnen von Gegenständen zählen zu diesen Aufgaben. Eine interessante Anwendung der Begriffsverbände ist die Strukturierung und Abfrage von Wissen. Das Wissen über ein Anwendungsgebiet kann z.B. durch sogenannte ,,Formale Kontexte" und ,,Formale Begriffe" formalisiert, gespeichert und abgefragt werden. Ein Kontext beinhaltet eine Menge von Objekten (Gegenständen) und eine Menge von Eigenschaften (Merkmale), die diese Objekte besitzen können. Die Information über Objekt-Merkmal-Beziehungen ist ebenfalls im Kontext enthalten. Der Begriffsverband, der zu einem bestimmten Kontext konstruiert wird, ermöglicht Anfragen von der Art: folgt aus Eigenschaften a,b,c,... die Eigenschaften f,g,... ?; welche Objekte haben die Eigenschaften a,b,c,... ? oder welche Eigenschaften haben die Objekte x,y,z gemeinsam?
Wie sinnvoll und praktisch die begriffsanalytischen Methoden für die Wissensdarstellung eingesetzt werden können, hängt sehr stark von dem konkreten Wissensgebiet und den Anforderungen, die an die Lösungsmethoden des spezifischen Problems gestellt werden, ab.

Definition 4.1
Eine Menge M mit einer auf M definierten partiellen Ordnung heißt eine geordnete Menge.

Definition 4.2
Sei $(M, \leq)$ eine geordnete Menge. Für Elemente $a, b \in M$ nennt man a einen unteren Nachbarn von b, falls $a < b$ ist und es kein Element $c \in M$ mit $a < c < b$ gibt. Ist a ein unterer Nachbar von b, so heißt b ein oberer Nachbar von a.

Definition 4.3
Es sei $(M, \leq)$ eine geordnete Menge und die Menge $A \subseteq M$ beliebig. Ein Element $s \in M$ heißt eine untere Schranke von A, falls $\forall a \in A : \ s \leq a$. Entsprechend wird g eine obere Schranke von A genannt, falls $\forall a \in A : \ a \leq g$. Existieren die größte (kleinste) untere (obere) Schranke für A, so heißen sie Supremum ($\bigvee A$) bzw. Infimum ($\bigwedge A$) von A.

Es ist leicht einzusehen, daß Supremum und Infimum, falls sie überhaupt existieren, eindeutig sind.

Definition 4.4
Eine geordnete Menge $(V, \leq)$ heißt ein Verband, wenn zu je zwei Elemente $x, y \in V$ stets das Supremum $x \vee y$ und das Infimum $x \wedge y$ von $\{x, y\}$ existieren. V ist ein vollständiger Verband, falls jede beliebige Teilmenge $X \subseteq V$ ein Supremum $\bigvee X$ und ein Infimum $\bigwedge X$ besitzt.

Zum Beispiel ist die Potenzmenge einer beliebigen Menge M zusammen mit Inklusionsordnung $(\mathcal{P}(M), \subseteq)$ ein vollständiger Verband, denn für jede Menge $X \subseteq \mathcal{P}(M)$ gilt:

$$\bigvee X = \bigcup_{x \in X} x \text{ und } \bigwedge x = \bigcap_{x \in X} x$$

Bemerkung 4.5
Eigentlich ist die explizite Bedingung der Existenz des Supremums bei der Definition des vollständigen Verbandes überflüssig, weil diese aus der Existenz des Infimums folgt. Die Menge S besitzt als die Menge aller oberen Schranken von $X \subseteq V$ das Infimum s. Dann ist jedes $x \in X$ eine untere Schranke von S, also gilt $x \leq s$ für alle $x \in X$ und somit ist s das Supremum der Menge X.

Definition 4.6
Eine Abbildung $\varphi : V \to W$ zwischen zwei vollständigen Verbänden V und W heißt **supremum-erhaltend**, *wenn für jede Teilmenge $X \subseteq V$*

$$\varphi(\bigvee X) \ = \ \bigvee \varphi(X)$$

gilt. Entsprechend wird eine **infimum-erhaltende** *Abbildung definiert. Ist eine Abbildung sowohl infimum- als auch supremum-erhaltend, so heißt sie ein Vollhomomorphismus. Ein bijektiver Vollhomomorphismus heißt dann ein Verbandisomorphismus.*

Definition 4.7
Sei $(V, \leq)$ ein Verband, dann heißt $(V, \leq)^d := (V, \geq)$ der zu V duale Verband. Ist $(V, \leq)$ isomorph zu einem dualen Verband $(W, \leq)^d$, so werden V und W dual isomorphe Verbände genannt: $V \cong_d W$

4.2 Hüllensysteme

Ein Hüllensystem auf einer Menge G ist eine Menge von Teilmengen, die G enthält und gegen Durchschnitt abgeschlossen ist. Formal:

Definition 4.8
$H \subseteq \mathcal{P}(G)$ heißt ein Hüllensystem auf G, falls $G \in H$ und $\forall\ X \subseteq H : \bigcap X \in H$

Definition 4.9
Ein Hüllenoperator φ auf der Menge G ist eine Abbildung, die jede Teilmenge $X \subseteq G$ eine Hülle $\varphi(X) \subseteq G$ zuordnet, wobei für alle $X, Y \subseteq G$ gelten muß:

Monotonie $X \subseteq Y \Rightarrow \varphi(X) \subseteq \varphi(Y)$

Extensität $X \subseteq \varphi(X)$

Idempotenz $\varphi(\varphi(X)) = \varphi(X)$

Wenn $\varphi : \mathcal{P}(G) \to \mathcal{P}(G)$ ein Hüllenoperator auf G ist, dann nennt man jede Menge $\varphi(X)$ eine *Hülle* von φ. Jedes Hüllensystem H kann als die Menge aller Hüllen eines Hüllenoperators betrachtet werden und umgekehrt, d.h. jedes Hüllensystem kann durch einen Hüllenoperator beschrieben werden, und jeder Hüllenoperator definiert ein Hüllensystem. Das ist die Aussage des folgenden Lemmas.

Lemma 4.10

1. Jedes Hüllensystem ist die Menge aller Hüllen eines Hüllenoperators.
2. Umgekehrt ist die Menge aller Hüllen eines Hüllenoperators ein Hüllensystem.

Beweis

1. Sei $H \subseteq \mathcal{P}(G)$ ein Hüllensystem auf G. Wir zeigen, daß der Operator $\varphi_H(X) = \bigcap\{A \in H \mid X \subseteq A\}$ ein Hüllenoperator auf G ist und die Menge aller Hüllen dieses Operators gleich H ist.

   - Der Operator φ_H ist ein Hüllenoperator auf G, denn:

     (Monotonie): Aus $X \subseteq Y$ folgt: $\{A \in H \mid X \subseteq A\} \supseteq \{A \in H \mid Y \subseteq A\}$ und somit $\varphi_H(X) = \bigcap\{A \in H | X \subseteq A\} \subseteq \bigcap\{A \in H \mid Y \subseteq A\} = \varphi_H(Y)$.

     (Extensität und Idempotenz): Die Extensität ist trivial (da X $\subseteq$ X) und die Idempotenz gilt, denn nach Definition von φ_H enthält jedes Element $X \in H$, welches A enthält, auch $\varphi_H(A)$ und umgekehrt. Formal heißt dies:

$$\{A \in H \mid X \subseteq A\} \ = \ \{A \in H \mid \varphi_H(X) \subseteq A\}$$

- Die Menge aller Hüllen des Operators φ_H ist gleich der Menge H, denn die Menge aller Hüllen ist beschrieben durch $H_{\varphi_H} = \{\varphi_H(X) \mid X \subseteq G\}$. Nun gilt einerseits nach Definition:
  $X = \bigcap\{A \in H \mid X \subseteq A\} \Leftrightarrow X = \varphi_H(X) \Leftrightarrow X \in H_{\varphi_H}$ und andererseits $X \in H \Leftrightarrow X = \bigcap\{A \in H \mid X \subseteq A\}$, denn:
  $\star \Leftarrow$: Aus $X = \bigcap\{A \in H \mid X \subseteq A\}$ folgt sofort nach Definition 4.8: $X \in H$.
  $\star \Rightarrow$: Die Menge $\{A \in H \mid X \subseteq A\}$ besteht aus allen Mengen, die in H liegen und X enthalten. Also insbesondere liegt X in dieser Menge, denn nach Voraussetzung $X \in H$ und außerdem $X \subseteq X$. Somit ist der Schnitt über diese Menge genau gleich der Menge X.

2. Nun sei $H_\varphi = \{\varphi(X) \mid X \subseteq G\}$ die Menge aller Hüllen eines Hüllenoperators $\varphi: \mathcal{P}(G) \to \mathcal{P}(G)$. Dann ist H_φ ein Hüllensystem auf G, denn:
   Sei $X \subseteq H_\varphi$ beliebig. Wegen der Extensität von φ ist $\bigcap X \subset \varphi(\bigcap X)$. Mit Monotonie und Idempotenz von φ folgt aus $A \in X$ stets $\varphi(\bigcap X) \subseteq \varphi(A) = A$, was $\varphi(\bigcap X) \subseteq \bigcap X$ impliziert. Also es gilt $\varphi(\bigcap X) = \bigcap X$. Somit liegt die Schnittmenge über X in H_φ und somit ist H_φ nach Definition 4.8 ein Hüllensystem auf G.

□

Satz 4.11
Ist $H \subseteq \mathcal{P}(G)$ ein Hüllensystem, so ist $(H, \subseteq)$ ein vollständiger Verband mit Supremum und Infimum:

$$\bigvee X = \varphi_H(\bigcup X) \text{ und } \bigwedge X = \bigcap X \text{ für alle } X \subseteq H.$$

Dabei ist φ_H definiert durch $\varphi_H(X) = \bigcap\{A \in H \mid X \subseteq A\}$ ein Hüllenoperator für H. Umgekehrt ist jeder vollständige Verband isomorph zum Verband aller Hüllen eines Hüllensystems.

Beweis
Im Satz 4.10 wurde bewiesen, daß φ_H ein Hüllenoperator für das Hüllensystem H ist, d.h. H ist genau die Menge aller Hüllen von φ_H.
Jede geordnete Menge, in der zu jeder Teilmenge das Infimum existiert, ist ein vollständiger Verband, denn nach Bemerkung 4.5 impliziert die Existenz des Infimums auch die des Supremums. Daß $(H, \subseteq)$ eine solche geordnete Menge ist, ist klar, da $\bigcap X$ für alle $X \subseteq H$ das Infimum von X ist. Es ist ebenso offensichtlich, daß für alle $X \subseteq H \subseteq \mathcal{P}(G)$ die Menge $\varphi_H(\bigcup X)$ eine obere Schranke von X ist, denn nach der Definition des Hüllenoperators gilt für alle $A \in X$ die Inklusionsbeziehung $A \subseteq \varphi_H(A) \subseteq \varphi_H(\bigcup X)$.
Es muß noch gezeigt werden, daß diese obere Schranke die kleinste obere Schranke ist. Sei $S \subseteq \mathcal{P}(G)$ die Menge aller oberen Schranken von X. Nach dem ersten

Teil des Beweises ist $\bigwedge S = \bigcap S$ das Infimum von S. Jedes Element von X ist also eine Teilmenge von $\bigcap S$. Somit gilt $\bigcup X \subseteq \bigcap S$. Demzufolge gilt $\bigcap S \in \{A \in H \mid \bigcup X \subseteq A\}$. Also folgt $\varphi_H(\bigcup X) = \bigcap\{A \in H \mid \bigcup X \subseteq A\} \subseteq \bigcap S$. Andererseits gilt $\bigcap S \subseteq \varphi_H(\bigcup X)$, da $\varphi_H(\bigcup X)$ eine obere Schranke von X ist und somit in S liegt. Es gilt also $\varphi_H(\bigcup X) = \bigcap S$.

Sei nun umgekehrt $(V, \leq)$ ein vollständiger Verband. Dann ist das Mengensystem $S = \{\{x \in V \mid x \leq a\} \mid a \in V\}$ ein Hüllensystem, denn für jede Teilmenge $T \subseteq V$ gilt:

$$\bigcap_{a \in T} \{x \in V \mid x \leq a\} = \{x \in V \mid x \leq s, s = \bigwedge T\}.$$

Somit sind die Mengen der Form $\{x \in V \mid x \leq a\}$ für alle $a \in V$ Hüllen und nach Lemma 4.10 ist das System S ein Hüllensystem. □

4.3 Kontexte und Begriffe

Definition 4.12
Ein Kontext $K = (O, M, \rho)$ besteht aus einer Menge O von Objekten, einer Menge M von Merkmalen und einer binären Relation $\rho \subseteq O \times M$, die angibt, welches Objekt welches Merkmal besitzt.

Definition 4.13
Zwei Kontexte (O_1, M_1, ρ_1) und (O_2, M_2, ρ_2) sind ***isomorph****, wenn es bijektive Abbildungen $\alpha : O_1 \to O_2,\ \beta : M_1 \to M_2$ gibt mit $(g, m) \in \rho_1 \Leftrightarrow (\alpha(g), \beta(m)) \in \rho_2$ für alle $g \in O_1$ und alle $m \in M_1$.*

Ein Kontext läßt sich z. B. durch eine Kreuztabelle darstellen, deren Zeilen nach Objekten und deren Spalten nach Merkmalen benannt sind. Wo ein Objekt ein Merkmal besitzt (erfüllt), wird ein Kreuz eingetragen. Zum Beispiel ist in der Tabelle 4.1 der Kontext der reellen Zahlen dargestellt. Dabei sind einige Eigenschaften reeller Zahlen wie *natürlich, ganz, rational, algebraisch,* usw. betrachtet.

Definition 4.14
Für beliebige Mengen $A \subseteq O$ und $B \subseteq M$ definiert man :

$$A' = \{m \in M \mid \forall g \in A : \quad (g, m) \in \rho\}$$

$$B' = \{g \in O \mid \forall m \in B : \quad (g, m) \in \rho\}$$

A' ist also die Menge der Merkmale, die von allen Objekten in A erfüllt werden. Entsprechend ist B' die Menge der Objekte, die alle Merkmale in B besitzen.

Beispiel: $\{\pi\}' = \{IRR, TRANS, R\}$ und $\{\pi\}'' = \{\pi, e\}$ und $\{\pi\}''' = \{\pi\}'$

Tabelle 4.1: Kontext „Reelle Zahlen“

	N	Z^-	Z	Q^+	Q^-	Q	IRR	ALG	$TRANS$	R
1	×		×	×		×		×		×
-1		×	×		×	×		×		×
1/2				×		×		×		×
-1/2					×	×		×		×
$\sqrt{2}$							×	×		×
π							×		×	×
e							×		×	×

Definition 4.15
Ein „Formaler Begriff“ eines Kontextes $K = (O, M, \rho)$ ist ein Paar (A, B) mit $A \subseteq O$ und $B \subseteq M$ und $A' = B,\ B' = A$.
A heißt dann der Umfang und B der Inhalt des Begriffes (A, B).

Bemerkung 4.16
Direkt aus der Definition folgt, daß für alle $A \subseteq O$, $B \subseteq M$ das Paar (A, B) genau dann ein Begriff des Kontextes $K = (O, M, \rho)$ ist, wenn eine Menge $X \subseteq O$ bzw. eine Menge $Y \subseteq M$ existiert mit $(A, B) = (X'', X')$ bzw. $(A, B) = (Y', Y'')$. Die Mengen O und M tauchen als Umfang bzw. Inhalt der trivialen Grenzbegriffe auf: Die Menge aller Gegenstände ist Umfang des größten Begriffs, $(\emptyset', \emptyset'') = (O, O')$, dual ist M der Inhalt des kleinsten Begriffes, $(\emptyset'', \emptyset') = (M', M)$.

Definition 4.17
Sei $K = (O, M, \rho)$ ein Kontext. Für $g \in O$ heißt der Begriff $\gamma g := (g'', g')$ Gegenstandsbegriff zu g. Entsprechend heißt $\mu m := (m', m'')$ Merkmalbegriff zu einem $m \in M$. Die Objektsmenge m' (steht für $\{m\}'$) und Merkmalsmenge g' (steht für $\{g\}'$) heißen dann Merkmalsumfang bzw. Gegenstandsinhalt.

In Kreuztabellendarstellung spricht man von Spaltenumfängen bzw. Zeileninhalten anstatt von Merkmalsumfängen bzw. Gegenstandsinhalten.

Lemma 4.18
Sei $K = (O, M, \rho)$ ein Kontext und $B, B_1, B_2 \subseteq M$ und $A, A_1, A_2 \subseteq O$ beliebig. Dann gilt:

1. $B_1 \subseteq B_2 \Rightarrow B_2' \subseteq B_1'$
2. $B \subseteq B''$
3. $B' = B'''$
4. $A_1 \subseteq A_2 \Rightarrow A_2' \subseteq A_1'$

5. $A \subseteq A''$

6. $A' = A'''$

7. $A \subseteq B' \Leftrightarrow B \subseteq A' \Leftrightarrow A \times B \subseteq \rho$

Beweis

Zu 1) und 4): Ist $g \in B_2'$, so ist $(g, m) \in \rho$ für alle $m \in B_2$. Wegen $B_1 \subseteq B_2$ gilt insbesondere $(g, m) \in \rho$ für alle $m \in B_1$. Somit liegt g auch in B_1'. Der Beweis für Punkt (4) ist völlig analog.

Zu 2) und 5): Ist $m \in B$, so ist $(g, m) \in \rho$ für alle $g \in B'$ und damit folgt $m \in B''$. Der Beweis für Punkt (5) ist völlig analog.

Zu 3) und 6): Aus $B \subseteq B''$ folgt wegen (1) $B''' \subseteq B'$. Die Mengen B''' und B' sind Gegenstandsmengen, also $B' = A \subseteq O$ und $B''' = A'' \subseteq O$. Es gilt dann wegen (5) $A \subseteq A''$, was gleichbedeutend zu $B' \subseteq B'''$ ist. Es folgt also $B' = B'''$. Beweis für (6) ist völlig analog.

Zu 7): Die erste Äquivalenz folgt aus 1. und 2. ($\Leftarrow$) bzw. aus 4. und 5. ($\Rightarrow$). Zur zweiten Äquivalenz:

„$\Rightarrow$“ Sei $B \subseteq A'$. Dann gilt:
$A \times B \subseteq A \times A' \subseteq A \times \{m \in M \mid \forall g \in A : \ (g, m) \in \rho\} = \{(g, m) \mid g \in A, \ m \in M, \ (g, m) \in \rho\} \subseteq \rho$

„$\Leftarrow$“ Sei $A \times B \subseteq \rho$. Dann gilt für alle $g \in A, m \in B : \ (g, m) \in \rho$. Also folgt $\forall m \in B : \ m \in A'$, was $B \subseteq A'$ impliziert.

□

Korollar 4.19 *Die Operatoren $A \to A''$ und $B \to B''$ sind Hüllenoperatoren auf den Potenzmengen von O bzw. M.*

Beweis
Extensität folgt direkt aus Punkt (2) des Lemmas 4.18. Idempotenz folgt aus Punkt (3): $B'''' = (B''')' = (B')' = B''$. Monotonie folgt aus den Punkten (1) und (5): $B_1 \subseteq B_2 \Rightarrow B_2' \subseteq B_1' \Rightarrow B_1'' \subseteq B_2''$. □

Lemma 4.20
Die Menge aller Begriffsinhalte eines Kontextes $K = (O, M, \rho)$ mit Inklusionsbeziehung ist ein vollständiger Verband. Entsprechend gilt diese Aussage für die Menge aller Begriffsumfänge. Diese beiden Verbänden sind außerdem dual isomorph.

Beweis
Nach Bemerkung 4.16 ist jeder Begriff von der Form (A'', A') für eine Menge $A \subseteq O$ bzw. von der Form (B', B'') für eine Menge $B \subseteq M$. Somit bestehen die Mengen $\{B'' \mid B \subseteq M\}$ und $\{A'' \mid A \subseteq O\}$ aus allen Begriffsinhalten bzw. Begriffsumfängen des Kontextes. Andererseits sind nach Korollar 4.19 die Operatoren $A \to A''$ und $B \to B''$ Hüllenoperatoren auf den Potenzmengen von O bzw. M. Somit bilden diese Mengen, nach Satz 4.11 mit Inklusionsordnung vollständige Verbände. Daß diese beiden Verbände dual isomorph sind, folgt direkt aus der Definition 4.7 und der Tatsache, daß nach Lemma 4.18 für beliebige Begriffe (A_1, B_1) und (A_2, B_2) gilt: $A_1 \subseteq A_2 \Leftrightarrow B_2 \subseteq B_1$. □

Was uns im Hinblick auf unser Ziel besonders intressiert, ist die Berechnung der Inklusionsordnung unter den Begriffsinhalten[1], denn jeder Begriffsinhalt ist eine Merkmalsmenge und jede Merkmalsmenge entspricht einer Konjunktion von in einer Tbox definierten Konzeptnamen, falls man bzgl. dieser Tbox einen geeigneten Kontext definiert[2].

Die duale Isomorphie zwischen beiden Verbänden hat zur Folge, daß man alle Algorithmen zur Bestimmung der Begriffsumfänge völlig analog auch für Begriffsinhalte verwenden kann. Dafür braucht man nur die Rolle von Merkmalen und Gegenständen zu vertauschen. D.h. der Verband der Umfänge gespiegelt um eine horizontale Gerade ergibt den Verband der Inhalte. Dieses Dualitätsprinzip vereinfacht Definitionen und Beweise. Zu einer ordnungstheoretischen Aussage A bekommt man die duale Aussage A^d, wenn man in A das Zeichen $\leq$ durch $\geq$ ersetzt. Behauptet ein Lehrsatz zwei zueinander duale Aussagen, so wird in der Regel nur die eine bewiesen, die andere ergibt sich dann dual, d.h. mit dem gleichen Beweis für die duale Ordnung.

Lemma 4.21
Eine Menge $B \subseteq M$ ist genau dann ein Begriffsinhalt, wenn $B = B''$ ist. Entsprechend ist eine Menge $A \subseteq O$ genau dann ein Begriffsumfang, wenn $A = A''$ ist.

Beweis
Sei $B \subseteq M$ ein Begriffsinhalt mit entsprechendem Umfang $X \subseteq O$. Nach Bemerkung 4.16 muß ein $Y \subseteq M$ existieren so, daß $(X, B) = (Y', Y'')$ ist. Daraus folgt:

[1] Wie wir später sehen werden, möchten wir nicht diese ganze Hierarchie berechnen, sondern nur einen kleinen repräsentativen Teil davon.

[2] Ein solcher Kontext für Tboxen wird in Kapitel 5 angegeben (siehe Definition 5.1).

$X = Y'$ und $B = Y''$. Es folgt weiter nach Lemma 4.18 $B' = Y''' = Y' = X$ also: $B'' = X' = Y'' = B$. Die Umkehrung ist trivial. Der Beweis für Begriffsumfänge ist völlig analog. □

Definition 4.22
Der Begriff (A_1, B_1) heißt ein Unterbegriff von (A_2, B_2) (schreibe $(A_1, B_1) \leq (A_2, B_2)$) , wenn $A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1)$ ist. Die Relation $\leq$ heißt dann die hierarchische Ordnung unter den Begriffen.

Satz 4.23 (Hauptsatz über Begriffsverbände, 1.Teil)
Sei $K = (O, M, \rho)$ ein Kontext. Die hierarchische Ordnung auf der Menge aller Begriffe ist eine partielle Ordnung, und der Begriffsverband $\mathcal{V}(O, M, \rho)$ zusammen mit dieser Ordnung ist ein vollständiger Verband, bei dem die Suprema und Infima für beliebige Begriffsmengen

$$X = \{(A_j, B_j) \mid j \in J \text{ , } (A_j, B_j) \text{ ist ein Begriff}\}$$

folgendermaßen definiert sind:

$$\bigvee X = \left((\bigcup_{j \in J} A_j)'' \text{ , } \bigcap_{j \in J} B_j \right)$$

$$\bigwedge X = \left(\bigcap_{j \in J} A_j \text{ , } (\bigcup_{j \in J} B_j)'' \right)$$

Dieser Satz läßt sich folgendermaßen interpretieren:
Ein Unterbegriff hat weniger Gegenstände, die unter ihn fallen, wird aber dafür durch mehr Merkmale beschrieben als der Oberbegriff. Das Infimum einer Menge X von Begriffen ist gerade der größte gemeinsame Unterbegriff. Es ist gerade der Begriff, der die gemeinsamen Gegenstände aller Begriffe aus X hat. Dual dazu ist das Supremum der kleinste gemeinsame Oberbegriff, dessen Merkmale genau die gemeinsamen Merkmale aller Begriffe aus X sind.
Mit formalen Kontexten lassen sich auf die oben angegebene Weise vollständige Verbände erzeugen. Tatsächlich entsteht jeder vollständige Verband auf diese Weise. Dies ist die Aussage des folgenden Satzes, der zusammen mit Satz 4.23 den *Hauptsatz über Begriffsverbände* bildet (vgl. [28]).

Satz 4.24 (Hauptsatz über Begriffsverbände, 2.Teil)
Ein vollständiger Verband $(V, \leq)$ ist genau dann isomorph zum Begriffsverband eines Kontextes $K = (O, M, \rho)$, wenn es Abbildungen γ von O nach V und μ von M nach V gibt, so daß gilt:

(i) *Für jedes Element $x \in V$ gibt es eine Teilmenge $H \subseteq O$ mit $x = \bigvee\{\gamma(h) \mid h \in H\}$.*

(ii) *Für jedes Element $x \in V$ gibt es eine Teilmenge $N \subseteq M$ mit $x = \bigwedge\{\mu(n) \mid n \in N\}$.*

(iii) *Für alle $g \in O$ und $m \in M$ ist $(g, m) \in \rho$ äquivalent zu $\gamma(g) \leq \mu(m)$. Insbesondere ist jede vollständige Verband $(V, \leq)$ isomorph zum Begriffsverband $\mathcal{V}(V, V, \leq)$.*

Für graphische Darstellung von Verbänden gibt es viele effiziente Algorithmen, die sich in der Praxis bewährt haben. Daß die Menge aller Begriffe eines Kontextes ein vollständiger Verband ist, hat also u.a. den Vorteil, daß man diese Algorithmen zur Darstellung der Begriffshierarchie übernehmen kann. Eine übliche Darstellungsform der Begriffsverbände ist die Darstellung mittels der Liniendiagramme. Ganter und Wille haben beschrieben (siehe [28, 29]), wie man aus der Menge aller Begriffe eines Kontextes ein solches Diagramm gewinnen kann.
Eine interessante Folgerung des Hauptsatzes ist die Reduktion der Beschriftung der Knoten des vollständigen Verbandes eines Kontextes, dargestellt als ein Liniendiagramm. Ist nämlich $(V, \leq) = \mathcal{V}(O, M, \rho)$, so sind die Abbildungen γ und μ gegeben durch: $\gamma(g) = (g'', g')$ und $\mu(m) = (m', m'')$. Damit ist $\gamma(g)$ der kleinste Begriff, der den Gegenstand g in seinem Umfang enthält (auch der Gegenstandsbegriff von g genannt) und $\mu(m)$ ist der größte Begriff, der das Merkmal m in seinem Inhalt enthält (auch Merkmalsbegriff von m genannt). Man reduziert nun die Beschriftung, indem nur noch die Gegenstands- und Merkmalsbegriffe mit den zugehörigen Gegenständen und Merkmalen beschriftet werden. Zum Beispiel ist die Abbildung 4.1 eine solche vereinfachte Darstellung des Begriffsverbandes des Kontextes „Reelle Zahlen“, der in der Tabelle 4.1 beschrieben ist.
Wie liest man ein solches Liniendiagramm?

Die Begriffe des Kontextes werden durch kleine Kreise dargestellt und ein Begriff ist Unterbegriff eines anderen Begriffes genau dann, wenn ein aufsteigender Linienzug von dem den Unterbegriff darstellenden Kreis zu dem den Oberbegriff darstellenden Kreis führt.

Das Liniendiagramm ist folgendermaßen beschriftet:

Der Name des Gegenstands g wird etwas unterhalb des Kreises des Gegenstandsbegriffes $\gamma(g) = (g'', g')$ notiert, während der Name des Merkmals m etwas oberhalb des Kreises des Merkmalsbegriffes $\mu(m) = (m', m'')$ notiert wird. Die anderen Begriffe werden als einfache Kreise ohne Beschrifftung dargestellt (in dem Beispiel gibt es, bis auf den kleinsten Begriff $(\emptyset, M)$, nur Merkmals- und Gegenstandsbegriffe). Man sieht, daß dies eine wesentliche Vereinfachung der Verbandsdarstellung ist, insbesondere wenn man beachtet, daß z.B. die Beschriftung eines Liniendiagramms sogar für kleine Verbände sehr unübersichtlich werden kann.

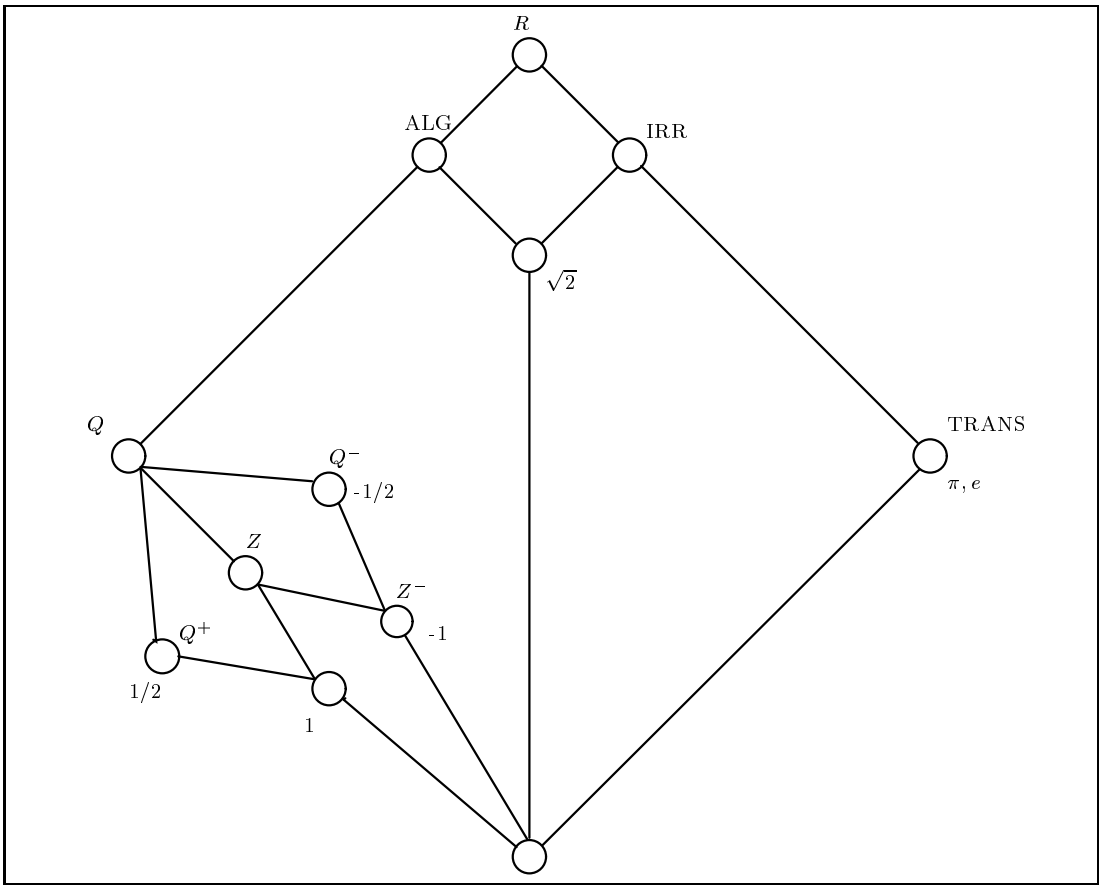

Abbildung 4.1: Begriffsverband des Kontextes „Reelle Zahlen“.

Das so beschriftete Liniendiagramm enthält die volle Information des Kontextes, denn nach dem Hauptsatz gilt: $(g, m) \in \rho \Leftrightarrow \gamma(g) \leq \mu(m)$.
Im Beispiel hat etwa der Gegenstand $g = -1/2$ die Merkmale Q^-, Q, ALG und R, aber nicht das Merkmal IRR, da von dem Kreis von $\gamma(g)$ zwar aufsteigende Linienzüge zu den Kreisen $\mu(Q^-)$, $\mu(Q)$, $\mu(\text{ALG})$ und $\mu(R)$ existieren, aber kein aufsteigender Linienzug zu dem Kreis von $\mu(\text{IRR})$ vorhanden ist.

Die vereinfachte Darstellung enthält auch die volle Information des Begriffsverbandes. Jeder Begriff eines Kontextes hat die Form (Y', Y'') für eine Menge $Y \subseteq M$. Für $Y \subseteq M$ und $X \subseteq O$ gilt $Y' = \bigcap_{m \in Y} m'$ und $X' = \bigcap_{g \in X} g'$. Das heißt: der Umfang eines Begriffes ist Durchschnitt bestimmter Spaltenumfänge (Ein Spaltenumfang ist der Umfang eines Merkmalsbegriffs $\mu(m) = (m', m'')$) und der Inhalt eines Begriffes ist Durchschnitt bestimmter Zeileninhalte (Ein Zeileninhalt ist der Inhalt eines Gegenstandsbegriffes $\gamma(g) = (g'', g')$). Es gilt also $(Y', Y'') = \bigwedge_{m \in Y} \mu(m)$. Aus dieser Tatsache zusammen mit dem Hauptsatz folgt, daß man jeden Begriff (Y', Y'') des Kontextes leicht aus der vereinfach-

ten Darstellung wiedergewinnen kann. Sucht man zu einem kleinen Kreis, der ja einen Begriff repräsentiert, den zugehörigen Begriffsumfang, so besteht dieser aus den Gegenständen, die an diesem Kreis oder an von diesem Kreis absteigenden Linienzügen liegen. Entsprechend findet man den zugehörigen Begriffsinhalt, indem man alle von dem Kreis aus aufsteigenden Linienzüge verfolgt und die daran eingetragenen Merkmale notiert. So erkennt man im Liniendiagramm 4.1 z.B. für den Begriff $\mu(Z) = (Z', Z'')$ leicht den Umfang $\{1, -1\}$ und den Inhalt $\{Z, Q, R, \text{ALG}\}$.

4.4 Bestimmung des Begriffsverbandes eines Kontextes

Wie wir in früheren Abschnitten gesehen haben, ist jeder Begriff eines Kontextes $K = (O, M, \rho)$ von der Form (X'', X') für eine Teilmenge $X \subseteq O$ und von der Form (Y', Y'') für eine Teilmenge $Y \subseteq M$. Umgekehrt ist jedes solches Paar ein Begriff. Allerdings ergibt sich aus dieser Aussage kein praktikables Verfahren für die Berechnung aller Begriffe eines Kontextes, da man solche Paare für alle Teilmengen berechnen müßte, was für größere Kontexte nur schwer möglich ist. Außerdem bräuchte man noch eine zusätzliche Nacharbeit, um die Begriff-Unterbegriff-Beziehungen zu bestimmen. Man nutzt nun die Charakterisierung vollständiger Verbände durch Hüllensysteme (siehe den Satz 4.11) aus. Es geht also um einen Algorithmus zur Erzeugung aller Hüllen eines Hüllenoperators. Es ist schon klar, daß es sich dabei um den Hüllenoperator $B \to B''$ (oder analog $A \to A''$ für Objektsmengen) handelt.

Definition 4.25
Sei $K = (O, M, \rho)$ ein Kontext. Es wird eine Aufzählung $p_1, p_2, p_3, \ldots$ der Elemente von M vorausgesetzt. Wir schreiben dann $p_i < p_j$, falls $i < j$ ist. Seien weiter $B_1 \neq B_2$ beliebige Teilmengen von M. Die Menge B_1 heißt lektisch kleiner als die Menge B_2, falls das bzgl. der $<$-Relation kleinste Element von M, in dem sich B_1 und B_2 unterscheiden, zu B_2 gehört. Formal:

$$B_1 \prec B_2 \Leftrightarrow \exists p_i \in B_2 \setminus B_1 \text{ mit } B_1 \cap \{p_1, p_2, \ldots p_{i-1}\} = B_2 \cap \{p_1, p_2, \ldots p_{i-1}\}$$

Lemma 4.26
Sei $K = (O, M, \rho)$ ein Kontext und M endlich. Die Ordnung $\prec$ definiert auf der Potenzmenge von M eine strikte, totale und Noethersche partielle Ordnung.

Beweis
Transitivität und Irreflexivität sind klar. Aus der Irreflexivität folgt sofort die Antisymmetrie, also ist $\prec$ eine strikte partielle Ordnung. Offensichtlich kann $A \prec B$ nur für $A \neq B$ gelten. Nun seien A und B verschiedene Teilmengen von M

und $B \not\prec A$. Nach Definition gehört dann das erste Element, in dem sich diese beiden Mengen unterscheiden, nicht zu A, also muß es zu B gehören, somit folgt $A \prec B$. Also gilt für $A \neq B$ entweder $A \prec B$ oder $B \prec A$, was zeigt, daß $\prec$ total ist.
Terminierung ist auch klar. Denn, angenommen, es gibt eine unendliche Kette $A_1 \succ A_2 \succ \ldots$, so muß es ein A_i geben, das in dieser Kette wiederholt vorkommt, da es nur endlich viele Teilmengen von M gibt. Wegen Transitivität folgt dann $A_i \prec A_i$. Dies kann aber nicht der Fall sein, denn nach Definition kann $A \prec B$ nur für $A \neq B$ gelten. □

Die lektische Ordnung stellt sicher, daß im Laufe der Berechnung der Begriffshierarchie alle Begriffsinhalte geordnet gefunden werden, d.h. der nächste gefundene Inhalt ist tatsächlich der nächst größere Inhalt (siehe Algorithmus 4.30). Ein weiteres Ziel ist es nun, für eine beliebige Menge $B \subseteq M$ den bzgl. lektischer Ordnung kleinsten Begriffsinhalt zu finden, der größer als B ist. Ist dieses Problem gelöst, so kann man alle Begriffsinhalte des Kontextes finden, indem man ausgehend vom kleinsten Begriffsinhalt $\emptyset''$ den nächst größeren Begriffsinhalt findet, bis man schließlich den größten Begriffsinhalt M erreicht hat. Dazu definieren wir für $p_i \in M,\ B_1, B_2 \subseteq M$:

Definition 4.27

- $B_1 <_i B_2 \Leftrightarrow p_i \in B_2 \setminus B_1$ *und* $B_1 \cap \{p_1, p_2, \ldots p_{i-1}\} = B_2 \cap \{p_1, p_2, \ldots p_{i-1}\}$.
- $B \oplus i := ((B \cap \{p_1, p_2, \ldots p_{i-1}\}) \cup \{p_i\})''$.

Hilfssatz 4.28
Ist $M = \{1, \ldots, n\}$, dann gelten für Mengen $A, B \subseteq M$ folgende Aussagen:

1. $i \in A \oplus i$
2. $A \prec B \Leftrightarrow A <_i B$ *für ein* $i \in M$.
3. $A <_i B$ *und* $A <_j C$ *mit* $i < j \Rightarrow C <_i B$.
4. $i \notin A \Rightarrow A \prec A \oplus i$.
5. $A <_i B$ *und* B *Begriffsinhalt* $\Rightarrow A \oplus i = B$ *oder* $A \oplus i \prec B$
6. $A <_i B$ *und* B *Begriffsinhalt* $\Rightarrow A <_i A \oplus i$.

Beweis

zu 1): Nach Definition und Lemma 4.18 Teil (2).

zu 2): Ist trivial.

zu 3): Aus $A <_i B$ und $A <_j C$ folgt:

$i \in B \setminus A,\ j \in C \setminus A,\ \ A \cap \{1, 2, \ldots, i-1\} = B \cap \{1, 2, \ldots i-1\}$ und $A \cap \{1, 2, \ldots, j-1\} = C \cap \{1, 2, \ldots, j-1\}$.
Wäre nun $i \in C$, dann (wegen $i < j$) $i \in C \cap \{1, 2, \ldots j-1\}$, aber wegen $i \notin A$ nicht in $A \cap \{1, 2, \ldots, j-1\}$, somit wären diese beiden Mengen nicht gleich. Also gilt $i \notin C$. Wegen $i \in B$ folgt dann: $i \in B \setminus C$.
Weiter gilt:

$$\begin{aligned} C \cap \{1, 2, \ldots, i-1\} &= C \cap \{1, 2, \ldots, j-1\} \cap \{1, 2, \ldots, i-1\} \\ &= A \cap \{1, 2, \ldots, j-1\} \cap \{1, 2, \ldots, i-1\} \\ &= A \cap \{1, 2, \ldots, i-1\} \\ &= B \cap \{1, 2, \ldots, i-1\}. \end{aligned}$$

zu 4) Nach (1) gilt $i \in A \oplus i$ und wegen $i \notin A$ folgt $i \in A \oplus i \setminus A$. Somit sind die Mengen A und $A \oplus i$ verschieden. Es gibt also ein kleinstes Element j in dem sich diese beiden Mengen unterscheiden. Ist $j = i$, so ist die Behauptung bewiesen, sonst muß $j < i$ sein. Liegt j in $A \oplus i$, so ist die Behauptung bewiesen, sonst muß gelten $j \in A \setminus A \oplus i$, daraus würde folgen: $j \in A \cap \{1, \ldots, i-1\}$ (wegen $j < i$ und $j \in A$). Daraus folgt wiederum $j \in ((A \cap \{1, \ldots, i-1\}) \cup \{i\})'' = A \oplus i$. Das ist aber ein Widerspruch zu $j \in A \setminus A \oplus i$. Also das kleinste Element, in dem sich diese beiden Mengen unterscheiden, kann nicht zu A gehören, also muß es zu $A \oplus i$ gehören, was nach Definition $A \prec A \oplus i$ impliziert.

zu 5) Es genügt zu zeigen, daß $A \oplus i \subseteq B$ gilt, denn aus $A \oplus i \subset B$ folgt direkt nach Definition der lektischen Ordnung: $A \oplus i \prec B$.

Aus $A <_i B$ folgt nach Definition:

$$i \in B \setminus A \text{ und } A \cap \{1, \ldots, i-1\} = B \cap \{1, \ldots, i-1\}.$$

Somit sind $A \oplus i$ und $B \oplus i$ gleiche Mengen. Andererseits gilt wegen $i \in B$:

$$B \oplus i = ((B \cap \{1, \ldots, i-1\}) \cup \{i\})'' = (B \cap \{1, \ldots, i\})'' \subseteq B''$$

Da die Menge B ein Begriffsinhalt ist, gilt nach Lemma 4.21 $B = B''$, woraus sofort $A \oplus i = B \oplus i \subseteq B$ folgt.

zu 6) Wegen $A <_i B$ gilt: $i \in B \setminus A$ und somit $i \in A \oplus i \setminus A$. Also ist nur zu zeigen, daß

$A \oplus i \ \cap \ \{1, \ldots, i-1\} = A \cap \{1, \ldots, i-1\}$ gilt. Es folgt mit:

$$
\begin{aligned}
A \oplus i \ \cap \ \{1, \ldots, i-1\} &= ((A \cap \{1, \ldots, i-1\}) \cup \{i\})'' \cap \{1, \ldots, i-1\} \\
&= ((B \cap \{1, \ldots, i-1\}) \cup \{i\})'' \cap \{1, \ldots, i-1\} \\
&= ((B \cap \{1, \ldots, i\}))'' \cap \{1, \ldots, i-1\} \\
&\subseteq B'' \cap \{1, \ldots, i-1\} \\
&= B \cap \{1, \ldots, i-1\} \\
&= A \cap \{1, \ldots, i-1\}.
\end{aligned}
$$

Die $\supseteq$-Richtung ist wegen $((A \cap \{1, \ldots, i-1\}) \cup \{i\})'' \supseteq A \cap \{1, \ldots, i-1\}$ klar.

□

Satz 4.29
Der kleinste Begriffsinhalt, der lektisch größer ist als eine gegebene Menge $A \subseteq M$, ist $A \oplus i$, wobei p_i das größte Element von M ist mit $A <_i A \oplus i$.

Beweis
Zum Beweis genügt der Hilfssatz 4.28:
Sei A^* der kleinste Begriffsinhalt nach A bzgl. der lektischen Ordnung. Wegen $A \prec A^*$ ist $A <_i A^*$ für ein $i \in M$ nach (2) und damit $A <_i A \oplus i$ nach (6). Nach (5) folgt $A \oplus i \leq A^*$, also $A \oplus i = A^*$ wegen $A \prec A \oplus i$ und der Tatsache, daß A^* als kleinster Begriffsinhalt nach A gewählt war. Daß i das größte Element ist mit $A <_i A \oplus i$, ergibt sich aus (3), denn $A <_j A \oplus j$ mit $i < j$ liefert nach (3) $A \oplus j \prec A^* = A \oplus i$ im Widerspruch zur Minimalität von A^*. □

Dieser Satz ist die Grundlage für einen gut programmierbaren Algorithmus zur Berechnung der Begriffshierarchie eines Kontextes. Der lektisch kleinste Begriffsinhalt ist $\emptyset''$. Für eine gegebene Menge $B \subset M$ findet man den lektisch nächsten Begriffsinhalt, indem man alle $p_i \in M \setminus B$ prüft, beginnend mit dem größten und in absteigender Folge, bis erstmals $B <_i B \oplus i$ ist. $B \oplus i$ ist dann der gesuchte Inhalt. Man iteriert dieses Verfahren so lange, bis der lektisch größte Inhalt (M) erzeugt wird. Dann hat man alle Begriffsinhalte in geordneter Reihenfolge gefunden. Formal für den Kontext $K = (O, M, \rho)$ mit endlicher Merkmalsmenge $M = \{1, 2, 3, \ldots n\}$:

Algorithmus 4.30

1. *Setze $j := n$.*
2. *Solange $j \in B$, setze $j := j - 1$.*
3. *Falls $j = 0$, beenden (dann war $B = M$, und es gibt keinen lektisch größeren Begriffsinhalt).*
4. *Berechne $\hat{B} := B \oplus j$.*

5. *Falls es ein $i < j$ mit $i \in \hat{B} \setminus B$ gibt, weiter mit (2).*

6. *$\hat{B}$ ist der lektisch nächste Inhalt zu B.*

Im Schritt (4) wird der aus B und j abgeleitete Inhalt $B \oplus j$ berechnet. Im Schritt (5) wird überprüft, ob der in (4) berechnete Inhalt tatsächlich der nächste Inhalt ist, oder ein lektisch zu großer Inhalt berechnet wurde. Wenn ja, dann wird der Algorithmus bei Schritt (2) mit einem kleineren j fortgesetzt. Der lektisch größte Inhalt ist immer M.

Besondere Aufmerksamkeit verdient der Schritt (5). Die Zahl j ist das größte Element von M, in dem sich B und $B \oplus j$ unterscheiden. Wegen $j \notin B$ folgt sofort nach Satz 4.28 Teil (4) $B \prec B \oplus j$. D.h. $\hat{B} = B \oplus j$ ist wirklich ein Begriffsinhalt, der lektisch größer als B ist. Nach Satz 4.29 ist $\hat{B}$ genau dann der lektisch nächste Inhalt zu B, wenn $B <_j \hat{B}$ gilt. Gibt es nun ein $i \in \hat{B} \setminus B$ mit $i < j$, so gilt $B \not<_j \hat{B}$, da die Bedingung

$$\hat{B} \cap \{1, 2, \ldots, j-1\} \;=\; B \cap \{1, 2, \ldots, j-1\}$$

wegen $i \in \hat{B} \cap \{1, 2, \ldots, j-1\}$ und $i \notin B \cap \{1, 2, \ldots, j-1\}$ nicht erfüllt ist. Demzufolge ist $\hat{B}$ zwar ein größerer Begriffsinhalt aber nicht der nächste Begriffsinhalt zu B.

Die Zeitkomplexität dieses Algorithmus ist $\mathcal{O}(n^3)$, wobei $n = max\{|O|, |M|\}$ ist (siehe [30]). Reuter und Ganter haben in [31] gezeigt, daß durch die Auswahl einer geeigneten Reihenfolge für die Merkmalsliste eine bessere Komplexität erreicht werden kann. Im Idealfall, d.h. wenn gewisse Zusatzbedingungen erfüllt sind, wird $\mathcal{O}(n^2)$ erreicht. Diese Optimierung ist aber für uns nicht von Interesse, denn, wie wir im Abschnitt 4.5.2 sehen werden, benutzen wir eine modifizierte Version des Algorithmus im Zusammenhang mit einer Implikationsbasis $\mathcal{L}$, wobei der Aufwand $\mathcal{O}(|\mathcal{L}|^2)$ beträgt.
Man kann nun mit Hilfe des Algorithmus 4.30 alle Begriffsinhalte eines Kontextes folgenderweise bestimmen:

- Berechne $B := \emptyset''$ (Dieses ist der lektisch kleinste Inhalt).
- Solange $B \neq M$, berechne $\hat{B}$ als lektisch nächsten Inhalt zu B und setze $B := \hat{B}$.

Die Korrektheit und Vollständigkeit dieses Algorithmus ist eine fast unmittelbare Folge des Satzes 4.29. Die Terminierung ist auch klar, da es für einen Kontext mit endlicher Merkmalsmenge nur endlich viele Begriffe gibt.
Wegen der Dualität zwischen Gegenständen und Merkmalen gelten obige Aussagen entsprechend auch für Mengen von Gegenständen.

Da ein Kontext $K := (O, M, \rho)$ im schlimmsten Fall $\mathcal{O}(2^{min\{|O|,|M|\}})$ Begriffe enthalten kann, ist dieser Algorithmus von exponentiellem Aufwand.

Wenn wir unser Hauptziel in Betracht ziehen, sehen wir ein, daß uns die Berechnung des Begriffsverbandes mittels der beschriebenen Methoden noch nicht zum Ziel führen kann, denn bis jetzt sind wir immer von einem bestehenden Kontext ausgegangen. Wir werden im Kapitel 5 sehen, daß man bei der Berechnung der erweiterten Subsumtionshierarchie einen geeigneten Kontext zu einer gegebenen Tbox definieren muß. Dieser Kontext wird dann ein unendlicher Kontext sein, deshalb müssen wir von einem leeren Kontext (oder einem kleinen Teilkontext des unendlichen Kontextes) ausgehen, um einen endlichen Kontext aufzubauen, dessen Begriffshierarchie einer Hierarchie für alle Konjunktionen zwischen Konzeptnamen entspricht.

Im nächsten Abschnitt betrachten wir die beschriebenen Methoden im Zusammenhang mit *Implikationen* und *Implikationsbasen*, was uns zum Ziel führen kann.

4.5 Merkmalexploration

Es kommt öfter vor, daß eine große (oder sogar unendliche) Menge von Gegenständen bzgl. einer kleinen Anzahl von Merkmalen klassifiziert werden soll. Es ist dann häufig undurchführbar (oder sogar unmöglich im Falle eines unendlichen Kontextes) den gesamten Kontext aufzuschreiben und dann die im Abschnitt 4.4 diskutierten Methoden zur Bestimmung des Begriffsverbandes anzuwenden. Zum Beispiel betrachten wir die unendliche Menge aller reellen Zahlen. Es gibt verschiedene Unterklassen der Klasse aller reellen Zahlen, z. B. die Klasse der natürlichen Zahlen (N), ganzen Zahlen (Z), rationalen Zahlen (Q) und so weiter. Man will zum Beispiel wissen, ob jede ganze Zahl auch algebraisch ist. Um solche Fragen mittels Begriffsanalyse zu beantworten, müssen wir einen Kontext betrachten, dessen Gegenstandsmenge die Menge aller reellen Zahlen und dessen Merkmale die Eigenschaften wie natürlich, algebraisch, ganz usw. sind.
Eine direkte Anwendung der bis jetzt diskutierten Methoden kann diese Aufgabe nicht lösen, da es unendlich viele reelle Zahlen gibt. Ausgangspunkt der Begriffsanalyse ist also in diesem Fall nicht ein explizit angegebener Kontext. Vielmehr erschließen wir den Kontext und damit auch das Begriffssystem aus der „Merkmalslogik“, also aus den Regeln, wie die Merkmale kombiniert werden können. Die Idee ist, das Expertenwissen über das Problemgebiet auszunutzen, um einen möglichst kleinen Teil des großen (bzw. unendlichen) Kontextes herauszufiltern, der alle relevanten Informationen für die Berechnung der Begriffshierarchie beinhaltet.
Um dies zu realisieren, brauchen wir einen Algorithmus, der, basierend auf bis-

herigen Ergebnissen, das Gewünschte leistet. Dieser arbeitet mit Implikationen, die durch sogenannte *Merkmalexploration* gewonnen werden. Diese Implikationen werden als Anfrage an einen Experten gestellt, der sie verneint oder bejaht. Mit Hilfe des Expertenwissens erzeugt dann der Algorithmus eine kleine Menge von Implikationen, die den Begriffsverband vollständig repräsentiert.

4.5.1 Implikation zwischen Merkmalen

Oft bemerkt man, daß es in einem Kontext $K_0 = (O_0, M, \rho_0)$ zwischen zwei Merkmalsmengen $A, B \subseteq M$ die Beziehung gibt, daß jeder Gegenstand mit den Merkmalen von A auch alle Merkmale von B hat. D.h. daß $A' \subseteq B'$ im Kontext K gilt. Man sagt dann: A impliziert B (in K).
Eine Merkmalsimplikation (im folgenden Implikation genannt) ist eine binäre Relation auf der Potenzmenge einer Merkmalsmenge M. Eine Implikation $A \to B$ wird von einer Menge $T \subseteq M$ erfüllt ($T \models A \to B$), falls $B \subseteq T$ oder $A \not\subseteq T$ ist. $T \subseteq M$ erfüllt eine Menge $\mathcal{L}$ von Implikationen, wenn T jede Implikation aus $\mathcal{L}$ erfüllt. Eine Implikation $A \to B$ gilt in einem Kontext $K = (O, M, \rho)$, falls gilt:

$$\forall g \in O : \; g' \text{ erfüllt } A \to B$$

Eine Implikation $A \to B$ gilt also in einem Kontext genau dann, wenn sie von allen Zeileninhalten (für jedes $g \in O$ heißt die Menge $g' \subseteq M$ ein Zeileninhalt, entsprechend heißt für jedes $m \in M$ die Menge $m' \subseteq O$ Spaltenumfang) des Kontextes erfüllt wird. Die im Kontext K geltenden Implikationen bezeichnen wir mit $\mathcal{I}mp(K)$. Die Menge $\mathcal{I}mp(K)$ besteht also genau aus denjenigen Implikationen, die von jeder Menge g' mit $g \in O$ erfüllt sind.

Im Kontext „Reelle Zahlen“ (Tabelle 4.1) gilt z.B. $\{Q\} \to \{ALG\}$, da jedes Objekt des Kontextes (bekanntlich sogar jede reelle Zahl) mit dem Merkmal Q auch das Merkmal ALG hat.

Wir haben im Abschnitt 4.3 eine vereinfachte Darstellung des Begriffsverbandes eines Kontextes diskutiert. Hier soll zur Ergänzung noch erwähnt werden, daß man alle gültigen Implikationen des Kontextes am vereinfachten Liniendiagramm ablesen kann. Denn eine Implikation $A \to B$ gilt in dem Kontext genau dann, wenn alle Implikationen $A \to m$, $m \in B$ im Kontext gelten. $A \to m$ gilt im Kontext genau dann, wenn $(m', m'') \geq (A', A'')$ gilt, also wenn $\mu(m) \geq \wedge\{\mu(n) \mid n \in A\}$ ist. Im Begriffsverband hat man also zu prüfen, ob der mit m bezeichnete Begriff ein Oberbegriff des Infimums aller mit $n \in A$ bezeichneten Begriffe ist. Zum Beispiel im Kontext „Reelle Zahlen“ 4.1 gilt die Implikation $A = \{Q, N, Z^-\} \to \text{IRR}$ nicht, denn $\wedge A = \mu(Z^-) \neq \mu(\text{IRR})$.

Ganter und Wille haben gezeigt, daß man mit Hilfe einer geeigneten Menge $\mathcal{L} \subseteq \mathcal{I}mp(K)$ den Begriffsverband des Kontextes berechnen kann, auch wenn der Kontext unendlich ist. Die Idee ist folgende:

Man berechnet mit Hilfe der Menge $\mathcal{L}$ einen Teilkontext $K = (O, M, \rho)$ des (möglicherweise unendlichen) Kontextes K_0, mit endlicher Objektsmenge $O \subseteq O_0$ und $\rho = \rho_0 \cap (O \times M)$ so, daß der Begriffsverband von K isomorph zu dem Begriffsverband von K_0 ist. Diese Idee wird durch Merkmalexploration realisiert. Zum Beispiel ist der Kontext „Reelle Zahlen“ (siehe Tabelle 4.1) durch Merkmalexploration erzeugt. Der Begriffsverband dieses Kontextes (siehe Abbildung 4.1) ist isomorph zu dem des unendlichen Kontexts aller reellen Zahlen. Der Kontext „Reelle Zahlen“ hat nur sieben Gegenstände, aber er liefert die volle Merkmalslogik aller in der Tabelle 4.1 definierten Merkmale.
Eine solche Implikationsmenge $\mathcal{L}$ muß natürlich bestimmte Bedingungen erfüllen. Im folgenden werden wir sehen, daß es eine solche geeignete Implikationsmenge gibt und, daß sie explizit erzeugt werden kann.

Satz 4.31
Eine Implikation $A \to B$ gilt in einem Kontext $K = (O, M, \rho)$ gdw. $B \subseteq A''$ ist. Sie ist automatisch von allen Begriffsinhalten des Kontextes erfüllt.

Beweis

$$\begin{aligned} B \not\subseteq A'' &\Leftrightarrow \text{Es ex. ein } m \in B \text{ mit } m \notin A'' \\ &\Leftrightarrow \text{Es ex. } m \in B,\ g \in A' \text{ mit } m \notin g' \\ &\Leftrightarrow \text{Es ex. } g \in A' \subseteq O \text{ mit } B \not\subseteq g' \\ &\Leftrightarrow \text{Es ex. } g \in O \text{ mit } A \subseteq g' \text{ und } B \not\subseteq g' \\ &\Leftrightarrow A \to B \text{ ist nicht in K erfüllt.} \end{aligned}$$

Es muß noch gezeigt werden, daß jede Implikation $A \to B \in \mathcal{I}mp(K)$ von jedem Begriffsinhalt erfüllt wird. Sei C ein Begriffsinhalt in K. Nach Bemerkung 4.16 gibt es dann ein $X \subseteq O$ mit $X' = C$. Aus $A \subseteq C = X'$ folgt nach Lemma 4.18 $A'' \subseteq X''' = X'$. Es gilt nach dem ersten Teil des Beweises $B \subseteq A'' \subseteq X' = C$. Also jede in K gültige Implikation ist von allen Begriffsinhalten erfüllt. □

Definition 4.32
Für eine Teilmenge $A \subseteq M$ ist $\mathcal{L}(A)$ die kleinste Teilmenge von M, die $\mathcal{L}$ erfüllt und A enthält. Sie heißt dann Implikationshülle von A .

Definition 4.33
Eine Implikation $A \to B$ folgt (semantisch) aus einer Menge $\mathcal{L}$ von Implikationen, falls jede Menge $T \subseteq M$, die $\mathcal{L}$ erfüllt, auch $A \to B$ erfüllt. Die Menge aller Implikationen, die aus $\mathcal{L}$ folgen, bezeichnet man mit $\mathcal{C}ons(\mathcal{L})$.

Bemerkung 4.34
Es ist leicht einzusehen, daß $\mathcal{C}ons(\mathcal{L})$ genau aus Implikationen $A \to B$ besteht, die der Bedingung $B \subseteq \mathcal{L}(A)$ genügen.

Eine Menge von Implikationen $\mathcal{L}$ heißt eine Basis von $\mathcal{I}mp(K)$ genau dann, wenn $\mathcal{C}ons(\mathcal{L}) = \mathcal{I}mp(K)$ und keine echte Teilmenge von $\mathcal{L}$ diese Eigenschaft hat. $\mathcal{L}$ ist also genau dann eine Basis von $\mathcal{I}mp(K)$, wenn alle in K geltenden Implikationen aus $\mathcal{L}$ folgen und umgekehrt alle aus $\mathcal{L}$ folgenden Implikationen in K gelten und $\mathcal{L}$ sich nicht reduzieren läßt, ohne diese Eigenschaft zu verlieren.

Eine Basis $\mathcal{L}$ eines Kontextes heißt minimal, falls sie unter allen Basen des Kontextes die kleinste Kardinalität hat. Die Berechnung einer Basis für einen Kontext ist eine zweite Möglichkeit, den Begriffsverband zu bestimmen. Aus der Menge $\mathcal{I}mp(K)$ kann man nämlich den ganzen Begriffsverband rekonstruieren. Umgekehrt lassen sich am Begriffsverband alle geltenden Implikationen ablesen[3]. Allerdings ist $\mathcal{I}mp(K)$ viel zu groß, um sie vollständig zu berechnen. Deshalb sind wir an einer minimalen Implikationsbasis interessiert, die den ganzen Informationsgehalt des Verbandes beinhaltet. Die Bestimmung des Begriffsverbandes durch eine Implikationsbasis hat gegenüber der im vorherigen Abschnitt diskutierten Methode den Vorteil, daß man erstens vom leeren Kontext ausgehen und mit Hilfe der Merkmalexploration die Begriffshierarchie vollständig erzeugen kann. Zweitens hat man dann eine Basis $\mathcal{L}$ für den Kontext zur Verfügung, die es ermöglicht, für jede beliebige Implikation effizient zu entscheiden, ob sie im Kontext gilt oder nicht. Wir können nämlich, wie es in [9] beschrieben ist, die Merkmale als propositionale Variablen betrachten. Eine Implikation $A \to B$ und eine Menge $\mathcal{L}$ von Implikationen entsprechen dann den Formeln bzw. der Formelmenge :

$$\Phi_{A\to B} := \bigwedge_{a\in A} a \to \bigwedge_{b\in B} b \quad \text{und} \quad \Gamma_{\mathcal{L}} = \{\Phi_{A\to B} \mid A \to B \in \mathcal{L}\}$$

Es besteht die Äquivalenz :

$$A \to B \in \mathcal{C}ons(\mathcal{L}) \Leftrightarrow \Gamma \models \Phi_{A\to B} \Leftrightarrow \Gamma \cup \{\neg\Phi_{A\to B}\} \text{ unerfüllbar}$$

Wenn nun $\mathcal{L}$ eine Basis für $\mathcal{I}mp(K)$ ist (d.h. $\mathcal{I}mp(K) = \mathcal{C}ons(\mathcal{L})$), so kann man für jede beliebige Implikation die Frage:

$$A \to B \in \mathcal{I}mp(K)?$$

in $\mathcal{O}(|\mathcal{L} \cup \{A \to B\}|)$ entscheiden, da die Erfüllbarkeit bzgl. propositionaler Hornklauselformeln bekanntlich linear in der Größe der Formeln entschieden werden kann [32].

Das Ziel ist also eine minimale Implikationsbasis zu erzeugen, die den ganzen Begriffsverband repräsentiert.
Guigues und Duquenne [33] haben gezeigt, daß zu jedem Kontext mit endlicher Merkmalsmenge eine solche Basis existiert. Ganter und Wille [28] haben beschrieben, wie diese Basis erzeugt werden kann.

[3] siehe [28] Seite 80.

Zur Erinnerung soll wieder erwähnt werden, daß wir einen geeigneten Kontext zu einer vorgegebenen Tbox definieren werden, der die Berechnung einer erweiterten Subsumtionshierarchie ermöglichen soll (siehe Kapitel 5, Definition 5.1). In diesem Kontext wird die Menge aller in der Tbox definierten Konzeptnamen als die Merkmalsmenge M definiert. Wie wir im Kapitel 5 sehen werden, entspricht jede Teilmenge $A \subseteq M$ einer Konjunktion zwischen Konzeptnamen. Man kann dann aufgrund der oben erwähnten Äquivalenz jede beliebige Subsumtion zwischen Konjunktionen von Konzeptnamen in linearer Zeit mit Hilfe der für diesen Kontext erzeugten Basis entscheiden.
Wir beschäftigen uns zuerst mit der Frage, wie eine solche Basis für einen beliebigen Kontext erzeugt werden kann.

4.5.2 Berechnung einer Guigues-Duquenne-Basis

Definition 4.35
Eine Menge $\mathcal{L} \subseteq \mathcal{I}mp(K)$ von Implikationen heißt **vollständig**, *falls jede in K geltende Implikation aus $\mathcal{L}$ folgt. $\mathcal{L}$ heißt* **nichtredundant**, *falls keine Implikation $A \to B \in \mathcal{L}$ aus den übrigen folgt.*

Hilfssatz 4.36
Eine Menge $\mathcal{L} \subseteq \mathcal{I}mp(K)$ ist eine Basis für den Kontext K gdw. $\mathcal{L}$ nichtredundant und vollständig ist.

Beweis

- „$\Longleftarrow$“
  Sei $\mathcal{L}$ nichtredundant und vollständig. Wegen Nichtredundanz ändert die Entfernung einer Implikation aus $\mathcal{L}$ die Menge $\mathcal{C}ons(\mathcal{L})$. Denn: angenommen es gibt eine Implikation $A \to B \in \mathcal{L}$ und $\mathcal{C}ons(\mathcal{L}) = \mathcal{C}ons(\mathcal{L} \setminus \{A \to B\})$. Das heißt alles, was aus $\mathcal{L}$ folgt, folgt auch aus $\mathcal{L} \setminus \{A \to B\}$ und umgekehrt. Wegen $A \to B \in \mathcal{L}$ folgt insbesondere $A \to B$ aus $\mathcal{L}$, also auch aus $\mathcal{L} \setminus \{A \to B\}$. Somit folgt die Implikation $A \to B$ aus den übrigen Implikationen von $\mathcal{L}$. Also ist $\mathcal{L}$ redundant, was der Voraussetzung widerspricht.
  Direkt aus der Definition der Vollständigkeit folgt: $\mathcal{I}mp(K) \subseteq \mathcal{C}ons(\mathcal{L})$. Es muß also nur noch die Eigenschaft $\mathcal{C}ons(\mathcal{L}) \subseteq \mathcal{I}mp(K)$ gezeigt werden.
  Angenommen es gibt eine Implikation $A \to B \in \mathcal{C}ons(\mathcal{L}) \setminus \mathcal{I}mp(k)$. Dann gibt es auch einen Zeileninhalt g' von K mit $g' \not\models A \to B$. Es folgt nach Definition der Erfüllbarkeit für Implikationen: $A \subseteq g'$ und $B \not\subseteq g'$. Aus $A \to B \in \mathcal{C}ons(\mathcal{L})$ folgt nach Bemerkung 4.34: $B \subseteq \mathcal{L}(A)$, wobei $\mathcal{L}(A)$ nach Definition die kleinste Menge ist, die $\mathcal{L}$ erfüllt und A enthält. Andererseits enthält g' die Menge A und erfüllt $\mathcal{L}$ (da $\mathcal{L} \subseteq \mathcal{I}mp(K)$). Es folgt also: $g' \supseteq \mathcal{L}(A) \supseteq B$. Das ist aber ein Widerspruch zur Annahme $B \not\subseteq g'$.
- „$\Longrightarrow$“
  Angenommen die Basis $\mathcal{L}$ ist redundant. Also gibt es eine Implikation $A \to B \in \mathcal{L}$ mit $\mathcal{L} \setminus \{A \to B\} \models A \to B$. Es folgt dann $\mathcal{C}ons(\mathcal{L}) =$

$\mathcal{C}ons(\mathcal{L} \setminus \{A \to B\})$. Denn jede Menge, die $\mathcal{L} \setminus \{A \to B\}$ erfüllt, erfüllt auch $\mathcal{L}$ und umgekehrt, somit folgt alles, was aus $\mathcal{L}$ folgt auch aus $\mathcal{L} \setminus \{A \to B\}$ und umgekehrt. Aus dieser Aussage und der Tatsache, daß $\mathcal{L}$ eine Basis ist, folgt dann $\mathcal{I}mp(K) = \mathcal{C}ons(\mathcal{L} \setminus \{A \to B\})$. Somit kann $\mathcal{L}$ keine Basis sein, was ein Widerspruch zur Annahme ist. Also ist $\mathcal{L}$ nichtredundant.
Direkt aus der Definition folgt $\mathcal{I}mp(K) = \mathcal{C}ons(\mathcal{L})$, insbesondere $\mathcal{I}mp(K) \subseteq \mathcal{C}ons(\mathcal{L})$. Somit ist eine Basis auch vollständig.

□

Es ist zu beachten, daß jede nichtredundante und vollständige Implikationsmenge zwar eine Basis ist, aber nicht unbedingt eine minimale Basis. Wir möchten natürlich eine minimale Basis für einen Kontext erzeugen.

Ein erster Ansatz, eine möglichst kleine Basis zu erhalten, besteht darin, diejenigen Implikationen wegzulassen, die trivialerweise aus den anderen folgen oder in jedem Kontext gelten. Zum Beispiel Implikationen der Form $A \to B$ gelten immer, falls $B \subseteq A$ ist. Ebenso folgt aus $C \subseteq B$ und $A \to B$ immer $A \to C$, also kann man in diesem Falle $A \to C$ weglassen, wenn man bereits $A \to B$ hat. Wenn man solche überflüssigen Implikationen aus $\mathcal{I}mp(K)$ entfernt hat, erhält man eine vollständige Implikationsmenge, die zwar kleiner als $\mathcal{I}mp(K)$, aber im allg. noch lange nicht redundanzfrei und minimal ist.

Hilfssatz 4.37
Sei $K = (O, M, \rho)$ ein Kontext. Für eine beliebige Implikationsmenge $\mathcal{L}$ ist die Menge

$$\mathcal{H}(\mathcal{L}) := \{X \subseteq M \mid X \text{ erfüllt } \mathcal{L}\}$$

ein Hüllensystem auf M.

Beweis
Sei $S \subseteq \mathcal{H}(\mathcal{L})$ beliebig. Nach Definition 4.9 ist nur zu zeigen, daß $\bigcap S \subseteq \mathcal{H}(\mathcal{L})$ gilt.
Ist $A \to B$ eine Implikation aus $\mathcal{L}$ und $A \subseteq \bigcap S$, so ist $A \subseteq T$ für alle $T \in S \subseteq \mathcal{H}(\mathcal{L})$. Da T in $\mathcal{H}(\mathcal{L})$ liegt, erfüllt es die Menge $\mathcal{L}$ und insbesondere die Implikation $A \to B$. Also folgt $B \subseteq T$ für alle Mengen $T \in S$ und somit $B \subseteq \bigcap S$. Demzufolge erfüllt die Menge $\bigcap S$ alle Implikationen von $\mathcal{L}$ und damit liegt sie in $\mathcal{H}(\mathcal{L})$. □

Den zugehörigen Hüllenoperator beschreibt man folgendermaßen: Für eine Menge $X \subseteq M$ sei

$$X^{\mathcal{L}} := X \cup \bigcup \{B \mid A \to B \in \mathcal{L}, A \subseteq X\}.$$

Man bildet die Mengen $X^{\mathcal{L}}, X^{\mathcal{L}\mathcal{L}}, X^{\mathcal{L}\mathcal{L}\mathcal{L}}$, ..., bis schließlich eine Menge $\mathcal{L}(X) := X^{\mathcal{L}\ldots\mathcal{L}}$ mit $\mathcal{L}(X) = \mathcal{L}(X)^{\mathcal{L}}$ entsteht.

Hilfssatz 4.38
Eine Menge $\mathcal{L} \subseteq \mathcal{I}mp(K)$ ist vollständig gdw. jede Menge $T \subseteq M$, die $\mathcal{L}$ erfüllt, ein Begriffsinhalt des Kontextes $K = (O, M, \rho)$ ist.

Beweis

- „⇒“
  Sei $\mathcal{L}$ vollständig. Angenommen es gibt eine Menge $T \subseteq M$, die $\mathcal{L}$ erfüllt und kein Begriffsinhalt ist. Die Menge T liegt in $\mathcal{H}(\mathcal{L}) = \{X \mid X \text{ erfüllt } \mathcal{L}\}$. Da T kein Begriffsinhalt ist, folgt $T \neq T''$. Wegen $T'' \subseteq T''$ folgt sofort nach Satz 4.31 $T \to T'' \in \mathcal{I}mp(K)$. Andererseits $T \not\models T \to T''$, denn es gilt $T \subseteq T$ und $T'' \not\subseteq T$. Somit gibt es eine Menge T, die $\mathcal{L}$ erfüllt aber nicht die Implikation $T \to T''$. Demzufolge $T \to T'' \notin \mathcal{C}ons(\mathcal{L})$ und damit $Imp(K) \not\subseteq \mathcal{C}ons(\mathcal{L})$. Das ist aber ein Widerspruch zur Vollständigkeit der Menge $\mathcal{L}$.

- „⇐“
  Umgekehrt sei jede Menge $X \subseteq M$ mit $X \models \mathcal{L}$ ein Begriffsinhalt. Angenommen $\mathcal{L}$ ist nicht vollständig. Es gibt dann ein $T \subseteq M$ und eine in K gültige Implikation $A \to B$ mit $T \not\models A \to B$ und $T \models \mathcal{L}$. Es folgt dann $A \subseteq T$ und $B \not\subseteq T$. Da die Menge T die Implikationsmenge $\mathcal{L}$ erfüllt, muß sie (nach Voraussetzung) ein Begriffsinhalt sein und somit $T = T''$. Aus $A \subseteq T$ folgt $A'' \subseteq T'' = T$. Andererseits gilt $B \subseteq A''$, da $A \to B$ eine gültige Implikation in K ist, also folgt $B \subseteq A'' \subseteq T$, was ein Widerspruch zu $B \not\subseteq T$ ist.

□

Lemma 4.39
Ist $\mathcal{L}$ eine Basis für den Kontext K, so ist das Hüllensystem

$$\mathcal{H}(\mathcal{L}) = \{X \mid X \textit{ erfüllt } \mathcal{L}\}$$

gleich der Menge aller Begriffsinhalte des Kontextes K.

Beweis
Sei $\mathcal{L}$ eine Basis für K. Dann ist $\mathcal{L}$ insbesondere vollständig. Es folgt dann nach Satz 4.38, daß jede Menge, die $\mathcal{L}$ erfüllt, d.h. jedes $X \in \mathcal{H}(\mathcal{L})$, ein Begriffsinhalt ist. Andererseits ist, wegen $\mathcal{I}mp(K) = \mathcal{C}ons(\mathcal{L})$, jede Implikation $A \to B \in \mathcal{L}$ in K und damit nach Satz 4.31 automatisch von jedem Begriffsinhalt erfüllt. Somit liegt jeder Begriffsinhalt in $\mathcal{H}(\mathcal{L})$. Also ist $\mathcal{H}(\mathcal{L})$ die Menge aller Begriffsinhalte des Kontextes K. □

Definition 4.40
*Eine Menge $P \subseteq M$ heißt ein **Pseudoinhalt** des Kontextes $K = (O, M, \rho)$ genau dann, wenn $P \neq P''$ ist und für jeden Pseudoinhalt $Q \subset P$ schon $Q'' \subseteq P$ gilt.*

Es ist zu beachten, daß die Implikationen von der Form $X \to X''$ nach Satz 4.31 stets im Kontext gelten.

Der nächste Satz aus [28] ist die Grundlage eines Algorithmus, der zur Erzeugung einer minimalen, nichtredundanten und vollständigen Implikationsmenge führt.

Satz 4.41
Die Menge der Implikationen

$$\mathcal{L} = \{P \to P'' \mid P \text{ Pseudoinhalt des Kontext } K\}$$

ist nichtredundant und vollständig.

Beweis
$\mathcal{L}$ wird von K erfüllt, denn nach Satz 4.31 gilt $P \to P''$ in K gdw. $P'' \subseteq P''$ gilt. Um zu zeigen, daß $\mathcal{L}$ vollständig ist, müssen wir nach Hilfssatz 4.38 zeigen, daß jede Menge $T \subseteq M$, die $\mathcal{L}$ erfüllt, ein Begriffsinhalt ist. Angenommen, T ist kein Begriffsinhalt. Dann ist nach Lemma 4.21 $T \neq T''$. Sei nun $Q \subset T$ ein Pseudoinhalt von K. Dann liegt $Q \to Q''$ in $\mathcal{L}$. Da $\mathcal{L}$ von T erfüllt ist, folgt $Q'' \subseteq T$. Es folgt also, daß T selbst ein Pseudoinhalt ist, d.h. $T \to T'' \in \mathcal{L}$. Da T die Implikation $T \to T''$ erfüllt, folgt: $T'' \subseteq T$ und somit $T'' = T$, also ein Widerspruch zur Annahme.
Um zu zeigen, daß $\mathcal{L}$ nichtredundant ist, genügt es für jeden Pseudoinhalt P eine Menge $T \subseteq M$ anzugeben, die die Menge $\mathcal{L} \setminus \{P \to P''\}$ erfüllt, aber die Implikation $P \to P''$ nicht erfüllt. Somit ist dann für jede Implikation $P \to P'' \in \mathcal{L}$ gezeigt :

$$\mathcal{L} \setminus \{P \to P''\} \not\models P \to P''.$$

Man kann leicht einsehen daß die Menge P diese Eigenschaft hat, denn

$$P \neq P'' \Rightarrow P \subset P'' \Rightarrow P'' \not\subseteq P \Rightarrow P \not\models P \to P''.$$

Andererseits erfüllt P die Menge $\mathcal{L} \setminus \{P \to P''\}$. Sei $Q \to Q'' \in \mathcal{L} \setminus \{P \to P''\}$ beliebig und $Q \subset P$. Da P ein Pseudoinhalt ist, folgt $Q'' \subseteq P$ und somit erfüllt P jede Implikation $Q \to Q'' \in \mathcal{L} \setminus \{P \to P''\}$. □

Nach diesem Satz müssen wir nur die Menge aller Pseudoinhalte eines Kontextes bestimmen, um eine vollständige und nichtredundante Implikationsmenge für einen Kontext zu erhalten.
Das weitere Ziel ist also, alle Pseudoinhalte eines Kontextes mit endlicher Merkmalsmenge zu bestimmen. Die rekursive Definition der Pseudoinhalte ergibt eine erste Möglichkeit, sie zu bestimmen. Diese ist aber sehr ineffizient. Mit Hilfe des Algorithmus 4.30 und der lektischen Ordnung (siehe Definition 4.25) und des Lemmas 4.43 haben Ganter und Wille [28] den Algorithmus 4.44 entworfen, der diese Aufgabe löst. Zuerst einige Sätze :

Satz 4.42
Sind P und Q Begriffs- oder Pseudoinhalte mit $P \not\subseteq Q$ und $Q \not\subseteq P$, so ist $P \cap Q$ ein Begriffsinhalt.

Beweis
Sowohl P als auch Q, und damit auch $P \cap Q$, erfüllen alle Implikationen aus der nichtredundanten und vollständigen Implikationsmenge

$$\mathcal{L} = \{P \to P'' \mid P \text{ Pseudoinhalt vom Kontext } K\}$$

mit Ausnahme von $P \to P''$ und $Q \to Q''$. Ist nun $P \neq P \cap Q \neq Q$, so erfüllt die Menge $P \cap Q$ auch diese Implikationen (da die Vorbedingung schon nicht erfüllt ist), und somit ist sie ein Begriffsinhalt. □

Unmittelbar aus diesem Satz folgt, daß die Menge aller Inhalte eines Kontextes ein vollständiger Verband ist. Somit gilt nach Satz 4.11 folgendes Lemma:

Lemma 4.43
Die Menge aller Pseudoinhalte und Begriffsinhalte eines Kontextes $K = (O, M, \rho)$ mit endlicher Merkmalsmenge ist ein Hüllensystem.

Der Hüllenoperator zu diesem System entsteht durch eine Modifikation des bzgl. des Satzes 4.37 angegebenen Hüllenoperators (vgl. [28]). Ausgehend von einer Menge $X \subseteq M$ und einer vollständigen Implikationsmenge $\mathcal{L} \subseteq \mathcal{I}mp(K)$ bildet man sukzessive

$$X^{\mathcal{L}^*} := X \cup \bigcup\{B \mid A \to B \in \mathcal{L}, A \subset X\}$$

$$X^{\mathcal{L}^*\mathcal{L}^*} := X^{\mathcal{L}^*} \cup \bigcup\{B \mid A \to B \in \mathcal{L}, A \subset X^{\mathcal{L}^*}\}$$

und so fort, bis schließlich eine Menge $\mathcal{L}^*(X)$ mit $\mathcal{L}^*(X) = (\mathcal{L}^*(X))^{\mathcal{L}^*}$ erreicht wird.
Diese Menge ist dann eine Hülle des Hüllensystems und somit ein Pseudoinhalt oder ein Begriffsinhalt. Es ist klar, daß dieser Prozeß terminiert, da nur endlich viele Implikationen in einem Kontext mit endlicher Merkmalsmenge existieren. Es ist zu beachten, daß zur Erzeugung aller Begriffs- und Pseudoinhalte nur eine vollständige Implikationsmenge $\mathcal{L} \subseteq \mathcal{I}mp(K)$ ausreicht. Wie wir später sehen werden, kann man diese Menge schrittweise durch die Merkmalexploration aufbauen. Durch die lektische Ordnung ist dann gewährleistet, daß $\mathcal{L}$ in jedem Schritt alle relevanten Implikationen enthält, die für die Berechnung des nächsten Inhalts notwendig sind (siehe Algorithmus 4.45). Es ist also möglich, alle Inhalte eines Kontextes zu berechnen, auch wenn der Kontext unendlich viele Gegenstände enthält.

Wenn man den Hüllenoperator genauer betrachtet, bemerkt man, daß für die Erzeugung dieser Inhalte nur die Implikationen $A \to B$ benötigt werden, deren Prämissen A eine echte Teilmenge des Pseudo- oder Begriffsinhalts sind. Diese Tatsache erlaubt eine rekursive Erzeugung aller Begriffs- oder Pseudoinhalte eines Kontextes mit Hilfe einer ziemlich kleinen Teilmenge $\mathcal{L} \subseteq \mathcal{I}mp(K)$.

Algorithmus 4.44
Es wird angenommen, daß $M = \{1, 2, 3, \ldots, n\}$ ist.
Der lektisch kleinste Begriffs- oder Pseudoinhalt ist die leere Menge $\emptyset \subseteq M$. Für eine gegebene Menge B findet man den lektisch nächsten Begriffs- oder Pseudoinhalt, indem man alle Elemente $j \in M$ prüft, beginnend mit dem größten und dann in absteigender Folge, bis erstmals $B <_j \mathcal{L}^*((B \cap \{1, 2, \ldots j-1\}) \cup \{j\})$ ist. Gibt es kein solches $j \in M$, so ist B der größte Begriffs- oder Pseudoinhalt, sonst ist dann die Menge
$\mathcal{L}^*((B \cap \{1, 2, \ldots j-1\}) \cup \{j\})$ der nächste Begriffs- oder Pseudoinhalt.

Wir sind natürlich nur an allen Pseudoinhalten des endgültigen Kontextes interessiert (siehe Satz 4.41). Es ist aber leicht, die Pseudoinhalte von den Begriffsinhalten zu unterscheiden. Man bestimmt nämlich den nächsten Pseudo- oder Begriffsinhalt B. Ist $B = B''$, so ist B ein Begriffsinhalt, sonst ein Pseudoinhalt des aktuellen Kontexts und somit nach Satz 4.48 und Lemma 6.12 auch Begriffs- bzw. Pseudoinhalt des endgültigen Kontexts.

Die Ausgangsfrage war: Wie erhält man eine nichtredundante und vollständige Implikationsmenge? Insbesondere, wenn ein unendlicher Kontext K vorliegt. Diese Frage ist nun leicht zu beantworten:

Das Verfahren für den Kontext K wird mit leerer Implikationsmenge $\mathcal{L}$ begonnen. Immer, wenn ein neuer Pseudo- oder Begriffsinhalt B gefunden wird, wird überprüft, ob $B = B''$. Ist es der Fall, so ist B ein Begriffsinhalt des aktuellen Kontextes K_i, sonst ist B ein Pseudoinhalt von K_i. Nun muß überprüft werden, ob dieses B auch ein Pseudoinhalt des Kontextes K ist. Ein Experte beantwortet diese Frage, indem er entscheidet, ob die Implikation $B \to B'' \setminus B$ in K gilt oder nicht. Ist es der Fall, so ist B ein Pseudoinhalt von K und die Implikation $B \to B'' \setminus B$ wird zu der Basis $\mathcal{L}$ hinzugenommen. Andernfalls ist B kein Pseudoinhalt von K, und der Kontext wird um das vom Experten erzeugte Gegenbeispiel als neuer Gegenstand erweitert. Anschließend wird nach dem nächsten Pseudo- oder Begriffsinhalt gesucht. Das Verfahren wird iteriert, bis der größte Inhalt gefunden wird. Auf dieser Weise werden der Kontext und die Basis Schritt für Schritt erweitert, bis schließlich alle Pseudoinhalte von K und die dazugehörigen Implikationen berechnet worden sind. Durch die lektische Ordnung ist gewährleistet, daß $\mathcal{L}$ jeweils alle Implikationen enthält, die für die Bestimmung des nächsten Inhalts notwendig sind. Formal ist dieses Verfahren im Algorithmus 4.45 beschrieben.

Algorithmus 4.45

1. *Initialisierung: Man startet mit leerer Implikationsbasis* $\mathcal{L}_0 = \emptyset$ *und leerem Kontext* $K_0 = (\emptyset, M, \emptyset)$, *also* $O_0 = \emptyset$, *und lektisch kleinstem Inhalt* $B_0 = \emptyset$.

2. *Iteration:*

   - *Berechne* B_i'' *bzgl. des aktuellen Kontextes* K_i.
   - *Ist* $B_i = B_i''$, *so setze* $\mathcal{L}_{i+1} := \mathcal{L}_i$ *und* $O_{i+1} := O_i$ *und weiter mit 3.*
   - *Ist* $B_i \neq B_i''$, *dann frage den Experten, ob* $B_i \to B_i'' \setminus B$ *in* K *gilt und:*
     - *Falls die Antwort Ja ist, setze:*

       $$\begin{array}{lcl} \mathcal{L}_{i+1} & := & \mathcal{L}_i \cup \{B_i \to B_i'' \setminus B\} \\ O_{i+1} & := & O_i \\ i & := & i+1 \end{array}$$

       und gehe zu (3)
     - *Falls die Antwort Nein ist, setze:*

       $$\begin{array}{lcl} \mathcal{L}_{i+1} & := & \mathcal{L}_i \\ B_{i+1} & := & B_i \\ O_{i+1} & := & O_i \cup \{g_i\} \\ i & := & i+1 \end{array}$$

       Dabei ist g_i *das vom Experten erzeugte Gegenbeispiel. Dann gehe mit* O_{i+1}, B_{i+1} *und* $\mathcal{L}_{i+1}$ *zu 2.*

3. *Bestimme mit Hilfe des Algorithmus 4.44 den zu* B_i *nächsten Inhalt* B_{i+1}. *Ist* B_{i+1} *der größte Begriffsinhalt*[4] *von* K, *terminiere mit der Ausgabe* $\mathcal{L}_{i+1}$, *sonst gehe mit* $\mathcal{L}_{i+1}, B_{i+1}$ *und* O_{i+1} *zu 2.*

Bemerkung 4.46
Sei $K = (O, M, \rho)$ *ein Kontext. Für beliebige Mengen*

$$A, B \subseteq M \text{ gilt: } A \subset B \Rightarrow A \prec B$$

Das ist sehr leicht einzusehen, denn aus $A \subset B$ folgt, daß das erste Element, in dem sich A und B unterscheiden, in B liegt.

Hilfssatz 4.47
Sei $K = (O, M, \rho)$ *ein Kontext. Der Kontext* K_g *entsteht aus* K, *indem man einen neuen Gegenstand* g *zu der Gegenstandsmenge des Kontextes* K *hinzufügt. Formal:* $K_g = (O \cup \{g\}, M, \rho \cup \{(g, m) \mid m \in M \text{ und } g \text{ besitzt das Merkmal } m\})$. *Weiter bezeichnen wir zu jede Teilmenge* $B \subseteq M$ *die Hülle* B'' *im erweiterten Kontext* K_g *mit* B_g''. *Es gelten dann:*

[4] Der Algorithmus 4.44 erkennt den größten Begriffsinhalt, indem er den entsprechenden Zähler j auf Null überprüft.

1. *Für jedes $B \subseteq M$ gilt $B'' = B''_g$, falls die Menge g' die Implikation $B \to B''$ erfüllt.*

2. *Ist P der lektisch erste (kleinste) Pseudoinhalt in K, so gibt es im K_g keinen Pseudoinhalt Q mit $Q \prec P$.*

3. *Ist P der lektisch erste (kleinste) Pseudoinhalt in K und erfüllt g' die Implikation $P \to P''$, so ist P auch der lektisch kleinste Pseudoinhalt in K_g.*

Beweis

Zu 1) Ist $B \not\subseteq g'$, so bleibt B'' unverändert, d.h. $B'' = B''_g$. Ist $B \subseteq g'$, so gilt wegen $g' \models B \to B''$ auch $B'' \subseteq g'$ und somit $B''_g = B'' \cap g' = B''$.

Zu 2) Angenommen es gibt in K_g einen Pseudoinhalt Q mit $Q \prec P$. Dann muß Q auch ein Pseudoinhalt in K sein. Denn wäre dies nicht der Fall, müßte nach Definition 4.40 entweder $Q = Q''$ (in K) sein oder es müßte in K einen Pseudoinhalt $W \subset Q$ mit $W'' \not\subseteq Q$ geben.
Ist $Q = Q''$, so gilt $g' \models Q \to Q''$ und somit folgt nach (1) $Q'' = Q''_g$. Also $Q = Q''_g$. Das ist aber ein Widerspruch dazu, daß Q ein Pseudoinhalt in K_g ist.
Gibt es in K einen Pseudoinhalt $W \subset Q$, so folgt nach Bemerkung 4.46 $W \prec Q$ und wegen $Q \prec P$ auch $W \prec P$. Das ist aber ein Widerspruch dazu, daß P der lektisch kleinste Pseudoinhalt in K ist.

Zu 3) Aus (1) folgt $P''_g = P''$. Da P ein Pseudoinhalt in K ist, gilt $P \neq P''$ und somit $P \neq P''_g$. Aus (2) folgt, daß es in K_g keinen Pseudoinhalt Q mit $Q \prec P$ gibt. Insbesondere gibt es in K_g keinen Pseudoinhalt Q mit $Q \subset P$ (sonst wäre nach Bemerkung 4.46 $Q \prec P$). Daraus folgt zusammen mit der Definition 4.40, daß P ein Pseudoinhalt in K_g und insbesondere der lektisch kleinste Pseudoinhat in K_g ist.

□

Satz 4.48
Es sei $K = (O, M, \rho)$ ein Kontext und $P_1, P_2, \ldots, P_n$ die ersten n Pseudoinhalte von K bzgl. der lektischen Ordnung. Wird nun K um ein Objekt g erweitert, dessen Inhalt g' die Implikationen $P_i \to P_i''$ für alle $i \in \{1, 2, \ldots, n\}$ erfüllt, so sind $P_1, P_2, \ldots, P_n$ auch die lektisch ersten n Pseudoinhalte des erweiterten Kontextes K_g.

Beweis
Man kann diesen Satz durch Induktion über n beweisen:

- *Induktionsanfang* **(n = 1):** Direkt aus dem Hilfssatz 4.47 Teil (3) folgt, daß P_1 der lektisch erste (kleinste) Pseudoinhalt im erweiterten Kontext K_g ist.

- *Induktionsschluß:* Sei die Behauptung für die $(n-1)$ ersten Pseudoinhalte des Kontextes K $(P_1, \ldots, P_{n-1})$ bewiesen. Es ist zu zeigen, daß P_n als lektisch n-ter Pseudoinhalt in K auch der lektisch n-te Pseudoinhalt in K_g ist. Es gibt in K_g keinen Pseudoinhalt P mit $P_{n-1} \prec P \prec P_n$. Wäre dies nicht der Fall, müßte P auch ein Pseudoinhalt in K sein (was ein Widerspruch dazu ist, daß P_n der n-te Pseudoinhalt in K ist). Denn wäre P kein Pseudoinhalt in K, so müßte in K entweder $P = P''$ und somit nach Hilfssatz 4.47 Teil (1) $P = P_g''$ gelten (was ein Widerspruch dazu ist, daß P ein Pseudoinhalt in K_g ist) oder es müßte nach Definition 4.40 in K einen Pseudoinhalt W mit $W \subset P$ und $W'' \not\subseteq P$ geben. Gäbe es einen solchen Pseudoinhalt W in K, so wäre er nach Bemerkung 4.46 lektisch kleiner als P, d.h. $W \in \{P_1, \ldots, P_{n-1}\}$. Es würde dann nach Induktionsvoraussetzung folgen, daß W auch ein Pseudoinhalt in K_g ist. Dies kann aber nicht der Fall sein, denn P und W sind beide Pseudoinhalte in K_g und daraus folgt zusammen mit $W \subset P$ und der Definition 4.40 $W'' \subseteq P$, was ein Widerspruch zu $W'' \not\subseteq P$ ist.

  Also gibt es in K_g keinen Pseudoinhalt P mit $P_{n-1} \prec P \prec P_n$. Daß P_n ein Pseudoinhalt in K_g ist, ist klar. Denn wäre dies nicht der Fall, gäbe es (nach Definition 4.40 und Bemerkung 4.46) wegen $P \neq P_{n_g}''$ (da P ein Pseudoinhalt in K ist, gilt in K $P \neq P''$ und andererseits gilt nach Hilfssatz 4.47 Teil (1) $P'' = P_{n_g}''$) einen Pseudoinhalt $Q \prec P_n$ in K_g mit $Q_g'' \not\subseteq P_n$ und somit mit Hilfssatz 4.47 Teil (1) $Q'' \not\subseteq P_n$. Nach Induktionsvoraussetzung ist Q ein Pseudoinhalt in K. Da auch P_n ein Pseudoinhalt in K ist, folgt $Q'' \subseteq P_n$. Also ein Widerspruch.
  Es ist gezeigt, daß P_n als der n-te Pseudoinhalt in K auch der n-te Pseudoinhalt in K_g ist.

□

Lemma 4.49
Sei g ein Gegenbeispiel, das während der Laufzeit des Algorithmus 4.45 zu einer Implikation erzeugt wurde. Die Menge g' erfüllt alle Implikationen aus der berechneten Basis $\mathcal{L}$.

Beweis
Der Beweis ist trivial. Nach Konstruktion des Algorithmus entscheidet der Experte zu jeder berechneten Implikation, ob sie im unendlichen Kontext K gilt. Eine Implikation wird nur dann zur Basis hinzugefügt, wenn sie in K gilt. Dies bedeutet, daß sie von allen Zeileninhalten des Kontextes erfüllt wird. Die erzeugten Gegenbeispiele gehören ja zu der Objektsmenge des Kontextes. Also für jedes Gegenbeispiel g und jede Implikation $A \to B \in \mathcal{L}$ gilt: $g' \models A \to B$. □

Bemerkung 4.50
Nach Konstruktion des Algorithmus 4.45 hat jede Implikation, die zu der Basis hinzugenommen wird, die Form $P \to P''$, wobei P ein Pseudoinhalt des aktuellen Kontextes K_i ist. Aus dieser Tatsache zusammen mit Lemma 4.49 und Satz 4.48 folgt die folgende Aussage:
Sind $P_1, \ldots, P_n$ die lektisch ersten n berechneten Pseudoinhalte des aktuellen Kontextes K_i, so sind sie auch die lektisch ersten n Pseudoinhalte des unendlichen Kontestes K.

Hilfssatz 4.51
Sei $K = (O, M, \rho)$ ein Kontext und I eine Indexmenge und $A_i \subseteq O$ für alle $i \in I$. Dann gilt:

$$\bigcap_{i \in I} A_i' = (\bigcup_{i \in I} A_i)'$$

Entsprechend gilt diese Aussage für Mengen von Merkmalen.

Beweis

$$\begin{aligned} m \in (\bigcup_{i \in I} A_i)' &\Leftrightarrow (g, m) \in \rho \text{ für alle } g \in \bigcup_{i \in I} A_i \\ &\Leftrightarrow (g, m) \in \rho \text{ für alle } g \in A_i \text{ für alle } i \in I \\ &\Leftrightarrow m \in A_i' \text{ für alle } i \in I \\ &\Leftrightarrow m \in \bigcap_{i \in I} A_i'. \end{aligned}$$

□

Lemma 4.52
Sei $K = (O, M, \rho)$ ein Kontext, für den der Begriffsverband mittels des Algorithmus 4.45 bestimmt werden soll. D.h. der Experte hat zu jeder erzeugten Implikation $A \to B$ zu entscheiden, ob $A \to B \in \mathcal{I}mp(K)$. Sei weiter $K_n = (O_n, M, \rho_n)$ der endliche, endgültige Kontext, den der Algorithmus erschlossen hat. Es gelten dann:

1. *Die durch den Algorithmus erzeugte vollständige und nichtredundante Implikationsmenge $\mathcal{L}$ ist eine Basis für den Kontext K.*

2. *Die Begriffsverbände von K_n und K sind isomorph.*

Beweis

Zu 1): Sei die Menge $\{P_1, \ldots, P_n\}$ die Menge aller Prämissen der Implikationen aus der Basis $\mathcal{L}$. Nach Lemma 4.49 für jedes Gegenbeispiel g erfüllt die Menge g' alle Implikationen aus $\mathcal{L}$. Somit sind nach Satz 4.48 $P_1, \ldots, P_n$ bzgl. der lektischen Ordnung die n ersten Pseudoinhalte in K_n bzw. in K.
Andererseits terminiert der Algorithmus genau dann, wenn der große Begriffsinhalt erzeugt wurde. Dies bedeutet, daß der Kontext K_n keinen Pseudoinhalt hat, der lektisch größer als P_n ist. Somit hat auch K keinen Pseudoinhalt, der lektisch

größer als P_n ist. Gäbe es in K einen Pseudoinhalt $P \succ P_n$, so wäre P ein solcher Pseudoinhalt auch in K_n. Wäre dies nicht der Fall, so müßte in K_n entweder $P = P''$ sein (was nach Hilfssatz 4.47 Teil (1) $P = P''$ auch in K zur Folge hätte) oder es müßte in K_n einen Pseudoinhalt $P_i, i \in \{1, \ldots n\}$ geben mit $P_i \subset P$ und $P_i'' \not\subseteq P$. Das ist aber ein Widerspruch, da P und P_i beide Pseudoinhalte in K sind.
Somit ist gezeigt, daß die Menge $\{P_1, \ldots, P_n\}$ aus allen Pseudoinhalten des unendlichen Kontextes K besteht. Nach Satz 4.41 ist dann die Implikationsmenge $\mathcal{L}$ auch für K eine nichtredundante und vollständige Implikationsmenge und nach Hilfssatz 4.36 eine Basis für K.

Zu 2): Seien $\mathcal{V}$ bzw. $\mathcal{V}_n$ Begriffsverbände der Kontexte K bzw. K_n. Also jedes Element $x \in \mathcal{V}$ ist ein Begriff des Kontextes K und hat nach Bemerkung 4.16 die Form (X'', X') bzw. (Y', Y'') für Mengen $X \subseteq O$ bzw. $Y \subseteq M$. Es ist nun zu zeigen: $\mathcal{V} \cong \mathcal{V}_n$. Nach dem ersten Teil des Beweises ist die erzeugte Basis $\mathcal{L}$ eine Basis sowohl für K_n als auch für K. Es folgt also nach Lemma 4.39, daß diese beiden Kontexte die gleiche Menge von Begriffsinhalten besitzen.
Sei nun $x = (X'', X') = (Y', Y'') \in \mathcal{V}$ beliebig gewählt. Die Menge X' und Y'' sind also Begriffsinhalte in den beiden Kontexten. Somit gibt es eine Gegenstandsmenge $H = \{h_1, \ldots, h_r\} \subseteq O_n$ mit $X' = \bigcap_{h \in H} h'$ und eine Merkmalsmenge $N = \{m_1, \ldots, m_s\} \subseteq M$ mit $Y' = \bigcap_{m \in N} m'$.
Es ist nun zu zeigen, daß die Abbildungen $\gamma : O_n \to \mathcal{V}$ bzw. $\mu : M \to \mathcal{V}$ definiert durch $\gamma(h) = (h'', h')$ bzw. $\mu(m) = (m', m'')$ und die Mengen $H \subseteq O_n$ und $N \subseteq M$ den erforderlichen Bedingungen im zweiten Teil des Hauptsatzes 4.24 genügen.
Es gelten

$$(*) \quad \bigvee\{\gamma(h) \mid h \in H\} = \bigvee\{(h'', h') \mid h \in H\}$$

$$(**) \quad \bigwedge\{\mu(m) \mid m \in N\} = \bigwedge\{(m', m'') \mid m \in N.\}$$

Nach dem ersten Teil des Hauptsatzes 4.23 ist $(*)$ gleich $((\bigcup_{h \in H} h'')'', \bigcap_{h \in H} h')$ und $(**)$ gleich $(\bigcap_{m \in N} m', (\bigcup_{m \in N} m'')'')$. Es gelten weiter nach Hilfssatz 4.51 und dem Lemma 4.18 und den vorher gezeigten Gleichungen $X' = \bigcap_{h \in H} h'$ und $Y' = \bigcap_{m \in N} m'$ die folgenden Gleichungen:

$$(\bigcup_{h \in H} h'')'' = (\bigcap_{h \in H} h''')' = (\bigcap_{h \in H} h')' = X''$$

$$(\bigcup_{m \in N} m'')'' = (\bigcap_{m \in N} m''')' = (\bigcap_{m \in N} m')' = Y''$$

Es ist also gezeigt, daß es Abbildungen γ, μ gibt, so daß für jedes $x \in \mathcal{V}$ Mengen $H \subseteq O_n$ und $N \subseteq M$ mit $x = \bigvee\{\gamma(h) \mid h \in H\}$ und $x = \bigwedge\{\mu(n) \mid n \in$

$N\}$ existieren. Somit folgt nach dem Hauptsatz, daß der Begriffsverband des Kontextes K_n isomorph zu dem Begriffsverband des Kontextes K ist. □

Die Vollständigkeit und Korrektheit des Algorithmus 4.45 folgt aus den in diesem Kapitel beschriebenen Sätzen, insbesondere aus den Sätzen 4.41 und 4.48 und Lemma 4.52. Zu der Terminierung soll hier kurz erwähnt werden, daß aufgrund der Endlichkeit der Merkmalsmenge und somit der Menge aller Begriffs- und Pseudoinhalte und der Tatsache, daß diese Inhalte in lektischer Ordnung berechnet werden, in endlicher Zeit der größte Inhalt berechnet wird und somit der Algorithmus terminiert.

Wie schon früher erwähnt wurde, ist jede vollständige und nichtredundante Implikationsmenge zwar eine Basis aber nicht unbedingt eine minimale Basis. Man kann aber zeigen, daß die durch den Algorithmus 4.45 berechnete Basis minimal ist.

Hilfssatz 4.53
Jede vollständige Implikationsmenge $\mathcal{L} \subseteq \mathcal{I}mp(K)$ enthält zu jedem Pseudoinhalt P eine Implikation $A \to B$ mit $A'' = P''$.

Beweis
Da ein Pseudoinhalt P immer ungleich P'' ist, kann er kein Begriffsinhalt sein. Da $\mathcal{L}$ vollständig ist, muß es wenigstens eine Implikation $A \to B \in \mathcal{L}$ geben, die von P nicht erfüllt wird, denn sonst wäre $P \in \mathcal{H}(\mathcal{L}) = \{X \mid X \text{ erfüllt } \mathcal{L}\}$ und somit wäre P nach Satz 4.38 ein Begriffsinhalt. Es gibt in $\mathcal{L}$ also eine solche Implikation $A \to B$ mit $A \subseteq P$ und $B \not\subseteq P$. Andererseits kann die Menge $A'' \cap P$ kein Begriffsinhalt sein. Angenommen sie ist ein Begriffsinhalt. Dann folgt sofort nach Satz 4.38 $A'' \cap P \in \mathcal{H}(\mathcal{L})$ und damit $A'' \cap P \models \mathcal{L}$. Die Menge $A'' \cap P$ erfüllt dann insbesondere $A \to B$. Daraus folgt wegen $A \subseteq A'' \cap P$ (da $A \subseteq A''$ und $A \subseteq P$) auch $B \subseteq A'' \cap P$. Das ist aber ein Widerspruch zu $B \not\subseteq P$.
Da $\mathcal{L}$ vollständig ist, gilt $A \to B$ in K, also folgt $B \subseteq A''$, was $A'' \not\subseteq P$ impliziert (da $B \not\subseteq P$). Daraus folgt $P \subseteq A''$, denn sonst würde nach Satz 4.42 folgen, daß $A'' \cap P$ ein Begriffsinhalt ist. Aus $P \subseteq A''$ folgt $P'' \subseteq A'''' = A''$ und aus $A \subseteq P$ folgt $A'' \subseteq P''$, also folgt $A'' = P''$. □

Korollar 4.54 *Sei K der Kontext, für den eine Basis durch den Algorithmus 4.45 erzeugt werden soll. Die durch den Algorithmus erzeugte Basis ist eine minimale Basis für den Kontext K.*

Daß diese Basis eine Basis für K ist, wurde durch Lemma 4.52 gezeigt. Die Minimalität folgt direkt aus dem Hilfssatz 4.53, denn jede andere Basis ist mindestens so groß wie die Anzahl der Pseudoinhalte, also mindestens so groß wie die durch den Algorithmus erzeugte Basis.

Es ist zu beachten, daß der Algorithmus genau dann terminiert, wenn der größte Begriffsinhalt von K und somit alle Pseudoinhalte berechnet wurden. Die angegebenen Eigenschaften (Basis, minimal, eindeutige Pseudoinhalte als Prämisse der Implikationen) der erzeugten Implikationsmenge ist also völlig unabhängig davon, welche Gegenbeispiele betrachtet werden. Die Wahl der Gegenbeispiele, wie wir in den nächsten Kapiteln sehen werden, bewirkt lediglich eine Beschleunigung oder Verlangsamung des Erzeugungsprozess der Pseudoinhalte des Kontextes K.

Wir haben nun einen Algorithmus zur Verfügung, der es uns ermöglicht, eine minimale, nichtredundante und vollständige Implikationsmenge (d.h. eine Minimalbasis) zu erzeugen. Ganter und Wille haben diesen Algorithmus im Zusammenhang mit einem interaktiven menschlichen Experten angegeben. Man möchte dem Menschen nicht zumuten, bzgl. einer terminologischen Box komplizierte und unübersichtliche Subsumtionen zu entscheiden. Im nächsten Kapitel sehen wir, wie eine modifizierte Version des funktionalen Algorithmus 2.10 als ein Experte eingesetzt werden kann.

Kapitel 5

Funktionaler Subsumtionsalgorithmus als ein Experte

Mit den in früheren Kapiteln beschriebenen Methoden hoffen wir nun, unser Hauptziel, die effiziente Berechnung einer erweiterten Subsumtionshierarchie, erreichen zu können. Um diese Methoden für diesen Zweck einzusetzen, benötigen wir einen geeigneten Kontext, dessen Minimalbasis einer minimalen Repräsentation der erweiterten Subsumtionshierarchie entspricht.
Dazu brauchen wir noch einen Experten, der im Algorithmus 4.45 eingesetzt werden kann. Im folgenden sei $\mathcal{T}$ die Tbox, für die die Hierarchie berechnet werden muß. Der Kontext $K_{\mathcal{T}} = (O, M, \rho)$ wird folgenderweise definiert:

Definition 5.1
Sei $M = \{m_1, m_2, \ldots, m_n\}$. Dabei ist m_i für alle $i \in \{1, 2, \ldots, n\}$ ein in $\mathcal{T}$ definierter Konzeptname.

$O = \{\ [\mathcal{I}, d] \mid \mathcal{I}$ *ist ein Modell von* $\mathcal{T}$ *und* $d \in \Delta^{\mathcal{I}}\}$

$\rho = \{\ ([\mathcal{I}, d], m) \mid d \in m^{\mathcal{I}}\}$

Für eine Teilmenge $B = \{m_{i1}, m_{i2}, \ldots, m_{ir}\} \subseteq M$ bezeichnen wir die Konjunktion $m_{i1} \sqcap m_{i2} \sqcap \ldots \sqcap m_{ir}$ mit $\sqcap B$, wobei für die leere Menge: $\sqcap\emptyset := \top$ vereinbart wird. Das folgende Lemma[1] zeigt uns, daß die durch den Algorithmus 4.45 erzeugte Minimalbasis zum Kontext $K_{\mathcal{T}}$ die Berechnung einer erweiterten Subsumtionshierarchie für die Tbox $\mathcal{T}$ ermöglicht.

Lemma 5.2
Für $B_1, B_2 \subseteq M$ gilt:

$$(K_{\mathcal{T}} \models B_1 \rightarrow B_2) \Leftrightarrow (\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2)$$

[1] Für den Beweis siehe [9].

Die Minimalbasis $\mathcal{L}_\mathcal{T}$ zu dem Kontext $K_\mathcal{T}$ ergibt also eine Repräsentation aller Subsumtionsrelationen der Form $\sqcap B_1 \sqsubseteq_\mathcal{T} \sqcap B_2$ für beliebige Teilmengen $B_1, B_2 \subseteq M$.
Wie im Abschnitt 4.5.1 erläutert wurde, kann dann jede Anfrage $\sqcap B_1 \sqsubseteq \sqcap B_2$ in $\mathcal{O}(|\mathcal{L}_\mathcal{T} \cup \{B_1 \rightarrow B_2\}|)$ mit Hilfe dieser Repräsentation entschieden werden.
Für die Berechnung von $\mathcal{L}_\mathcal{T}$ müssen wir außerdem einen Experten einsetzen. In [9] wurde bewiesen, daß der Erfüllbarkeitsalgorithmus 2.7 als ein Experte für den Kontext $K_\mathcal{T}$ eingesetzt werden kann. Der Experte muß zwei wesentliche Anforderungen erfüllen:

1. Er muß im Falle der Nichterfüllbarkeit der Subsumtion

   $$C = \sqcap B_1 \sqsubseteq \sqcap B_2 = D$$

   ein Gegenbeispiel $[\mathcal{I},\ d]$ liefern, das alle Merkmale aus B_1 besitzt und mindestens ein Merkmal aus B_2 nicht erfüllt. Der Algorithmus 2.7 erfüllt diese Anforderung, da er ausgehend vom Constraint $d : C \sqcap \neg D$ immer neue Individuen erzeugt, bis die Erfüllbarkeit entschieden ist. Somit erzeugt er gleichzeitig ein kanonisches Modell $\mathcal{I}$ (siehe Definition 2.3 und Satz 2.4), dessen Domain aus allen während der Laufzeit erzeugten Individuen besteht. Im Falle der Erfüllbarkeit der Eingabe (d.h. die Subsumtionsbeziehung gilt nicht) ist dann $[\mathcal{I}, d]$ das Gegegenbeispiel zu der Eingabe.

2. Um den Hüllenoperator $B \rightarrow B''$ bzgl. des aktuellen Kontextes anwenden zu können, muß bestimmt werden, welche Objekte welche Merkmale erfüllen. Der Experte muß also zu jedem Gegenbeispiel $[\mathcal{I},\ d]$ und jedem definierten Konzeptname $m \in M$ entscheiden, ob $d \in m^\mathcal{I}$ oder $d \notin m^\mathcal{I}$. Dieses Problem ist ein Spezialfall des *Model-checking*, das für $\mathcal{ALC}$ in polynomialer Zeit entscheidbar ist[2].

Das Einsetzen des Algorithmus 2.7 als ein Experte ist aber problematisch. Denn zum einen brauchen wir das vollständige Modell. Dadurch können wir nicht mit polynomialen Platz auskommen, sondern wir brauchen exponentiell viel Platz, denn wie im Abschnitt 2.4 diskutiert wurde, können das Constraintsystem und somit das Modell exponentiell wachsen (in der Größe des expandierten Eingabetermes). Zum anderen können auch die expandierten Terme exponentiell (in der Größe der Tbox) groß werden. Es ist also wünschenswert, einen Experten zu haben, der mit polynomialem Platz (in der Größe der Tbox) auskommt. Es liegt nahe, statt Algorithmus 2.7 die optimierte Version davon, den funktionalen Algorithmus 2.10, als Grundlage für die Realisierung eines PSPACE-Experten

[2]Schild [11] hat gezeigt, daß $\mathcal{ALC}$ eine syntaktische Variante der Multi-Modallogik **K** ist. Aus Resultaten in [34] folgt, daß *Model-checking* für **K** in Zeit $\mathcal{O}(nm)$ durchgeführt werden kann, wobei n die Größe der Formel und m die Größe des Modells ist.

(polynomil in der größe der Tbox) einzusetzen.
Das folgende Lemma aus [9] zeigt, daß ein solcher Experte realisierbar ist:

Lemma 5.3
Es ist möglich, einen „PSPACE-Experten" für den Kontext $K_\mathcal{T}$ zu implementieren, wobei alle relevanten Informationen über das Gegenbeispiel durch einen Algorithmus berechnet werden können, der polynomial (in der Größe von $\mathcal{T}$) viel Platz braucht.

Im folgenden wird gezeigt, daß eine modifizierte Version des funktionalen Algorithmus die Anforderungen an den Experten für den Kontext $K_\mathcal{T}$ vollständig erfüllt. Dieser erweiterte Algorithmus erhält als Eingabe außer der Menge $\{C_0\}$ für den Konzeptterm C_0 noch eine Liste $(E_1, E_2, \ldots, E_n)$ von Eigenschaften als einen zusätzlichen Parameter und gibt den Wert *false* aus, falls C_0 unerfüllbar ist und einen Booleanvektor $V = (b_1, b_2, \ldots, b_n) \in \{0,1\}^n$, sonst. Der Vektor V enthält dann alle relevanten Informationen, die für den weiteren Ablauf des Algorithmus 4.45 notwendig sind. Es gilt (wie später bewiesen wird) nämlich:

$$\forall b_u \in V: \quad b_u = 1 \Leftrightarrow d_0 \in E_u^\mathcal{I},$$

wobei $\mathcal{I}$ die durch den Algorithmus generierte Interpretation mit $d_0 \in C_0^\mathcal{I}$ ist.
Es ist zu beachten, daß für die Implementierung keine explizite Darstellung von Individuen und dem Modell notwendig ist, denn das Gegenbeispiel $[d_0,\ \mathcal{I}]$ kann mit dem zugehörigen Booleanvektor identifiziert werden. Demzufolge erzeugen wir während der Laufzeit des Algorithmus 4.45, in dem der Algorithmus 5.4 als Experte eingesetzt wird, einen Kontext $K = (O, M, \rho)$ mit:

$$\begin{aligned} O &= \{V \mid V \text{ ein Booleanvektor erzeugt durch einen Expertenaufruf}\} \\ M &= \{m_1, \ldots, m_n\} = \{m_i \mid m_i \text{ ein in } \mathcal{T} \text{ definierter Konzeptname }\} \\ \rho &= \{(V, m_i) \mid V[i] = 1\} \end{aligned}$$

Dieser Kontext ist ein endlicher Teilkontext des unendlichen Kontextes $K_\mathcal{T}$. Der Begriffsverband von Kontext $K_\mathcal{T}$ ist isomorph zu dem von K (siehe Lemma 4.52). Der endliche Kontext K, der während der Laufzeit des Algorithmus 4.45 aufgebaut wird, ist natürlich nicht eindeutig, da zu jeder Anfrage an den Experten mehrere Gegenbeispiele existieren können. Dies ist aber irrelevant (bzgl. der Korrektheit des Verfahrens), weil diese Kontexte isomorphe Begriffsverbände besitzen. Die Wahl der Gegenbeispiele ist insofern relevant (wie in Abschnitt 6.1 gezeigt wird), als sie die Anzahl der Expertenaufrufe beeinflußt.

Im folgenden gehen wir (o. B. d. A.) davon aus, daß sowohl die Tbox $\mathcal{T}$ als auch die Merkmalsliste M in NNF umgeformt sind. Wir haben also nur Boole'sche Kombinationen[3] aus Elementen von der Form: A, $\exists R.C$, $\forall R.D$, wobei A

[3] An dieser Stelle ist der Ausdruck „Boole'sche Kombination" nicht ganz korrekt, da es sich hier im engeren Sinne um $\mathcal{ALC}$-Konstruktoren handelt. Diese entsprechen aber auf der semantischen Ebene den Boole'schen Operatoren.

ein primitiver Konzeptname, R ein Rollenname und C, D $\mathcal{ALC}$-Konzeptterme in NNF sind.

Algorithmus 5.4 (Experte)
Sat $(S \mid M)$ =
if $A \in S$ *and* $\neg A \in S$
then false
else if $C_1 \sqcap C_2 \in S$
then Sat$(\{C_1, C_2\} \cup S \setminus \{C_1 \sqcap C_2\} \mid M)$
else if $C_1 \sqcup C_2 \in S$
then Sat$(\{C_1\} \cup S \setminus \{C_1 \sqcup C_2\} \mid M)$ *or Sat*$(\{C_2\} \cup S \setminus \{C_1 \sqcup C_2\} \mid M)$
else Some-Prozess$(S \mid M)$

Die Constraintmenge $\{C_0\}$ und M als die geordnete Menge aller in der Tbox definierten Konzeptnamen werden als Parameter eingegeben. So lange wie möglich werden $\sqcap$- und $\sqcup$-Regel angewendet. Sobald ein Widerspruch entdeckt wird, terminiert der Algorithmus mit Ausgabe *false*, andernfalls ruft er die Prozedur Some-Prozess auf, welche rekursiv Sat und einige andere Prozeduren aufruft. Die Endausgabe ist *false*, wenn der Konzeptterm C_0 unerfüllbar ist und ein Booleanvektor BV, falls C_0 erfüllbar ist, wobei für das Individuum $d_0 \in C_0^{\mathcal{I}}$ und einen in der Tbox definierten Konzeptname A_i gilt: $d_0 \in A_i^{\mathcal{I}}$ genau dann, wenn der i-te Eintrag des Ausgabevektors, d.h. BV[i], true ist. Im folgenden werden die weiteren Prozeduren, die durch Sat aufgerufen werden, beschrieben.

Prozedur 5.5
Eval $(S \mid M)$:=
Für alle Konzeptterme $C \in M$, *ersetze* C *in* M *durch Value* $(S \mid C)$ *und gib die neue Merkmalsliste aus*

Prozedur 5.6
Value $(S \mid C)$:=

- *Wenn* C *ein primitiver Name ist, dann* $\begin{cases} 1 & \text{falls } C \in S \\ 0 & \text{sonst} \end{cases}$
- *Wenn* $C = \forall R.D$, *dann*

$$\begin{cases} 1 & \text{falls } S \text{ kein Element der Form: } \exists R.D' \text{ enthält.} \\ C & \text{sonst} \end{cases}$$

- *Wenn* $C = \exists R.D$, *dann*

$$\begin{cases} 0 & \text{falls } S \text{ kein Element der Form: } \exists R.D' \text{ enthält.} \\ C & \text{sonst} \end{cases}$$

- *Wenn* $C = C_1 \sqcap C_2$ *, dann (Value* $(S \mid C_1)$ *and Value* $(S \mid C_2)$*)*
- *Wenn* $C = C_1 \sqcup C_2$ *, dann (Value* $(S \mid C_1)$ *or Value* $(S \mid C_2)$*)*
- *Wenn* $C = \neg C_1$ *, dann not (Value* $(S \mid C_1)$*)*

Es ist zu beachten, daß der Experte im Falle der Erfüllbarkeit des Eingabetermes immer einen Booleanvektor zu der eingegebenen Merkmalsliste ausgeben soll, auch wenn diese Liste leer ist. Im folgenden wird der leere Booleanvektor mit $\vec{\emptyset}$ bezeichnet.

Prozedur 5.7
Some-Prozess $(S \mid M)$ $=$
if $M = \emptyset$
then if for all $\exists R.C \in S$*:* *Sat* $(\{C\} \cup \{D \mid \forall R.D \in S\} \mid \emptyset) \neq$ *false*
then $\vec{\emptyset}$ *else false*
else if not exists a some-expression $\exists R.C \in S$
then Eval $(S \mid M)$ *else Continue* $(S \mid M)$

Prozedur 5.8
Continue $(S \mid M)$ $:=$ *Res,*
wobei das Resultat Res durch Ausführung folgender Schritte ermittelt wird:

1. $E :=$ *Eval* $(S \mid M)$
2. *S hat nun die Form:*

$$S = \left\{ \begin{array}{l} L_1 \ , L_2 \ , \ldots \ , L_q \ , \\ \exists R_1.C_{11} \ , \exists R_1.C_{12} \ , \ldots \ , \exists R_1.C_{1m_1} \ , \forall R_1.D_{11} \ , \ldots \ , \forall R_1.D_{1n_1} \\ \vdots \\ \exists R_k.C_{k1} \ , \exists R_k.C_{k2} \ , \ldots \ , \exists R_k.C_{km_k} \ , \forall R_k.D_{k1} \ , \ldots \ , \forall R_k.D_{kn_k} \end{array} \right.$$

Dabei ist L_i *für ein* $i \in \{1, 2, \ldots, q\}$ *ein Literal, also ein primitiver Konzeptname oder die Negation davon. Nun bestimme für alle*

$$i \in \{1, 2, \ldots, k\} \quad \textit{und} \quad j \in \{1, 2, \ldots, m_i\}$$

alle Constraintmengen: $\mathcal{C}_{ij} = \{C_{ij}, D_{i1}, \ldots, D_{in_i}\}$ *und zu jedem Rollenname* R_i *das Constraintsystem:*

$$\mathcal{C}_i = \{\mathcal{C}_{i1}, \mathcal{C}_{i2}, \ldots, \mathcal{C}_{im_i}\}$$

3. *Bestimme zu jeder Menge* $\mathcal{C}_i$ *mit* $i \in \{1, 2, \ldots, k\}$ *die zugehörige Merkmalsmenge:*

$$M\mathcal{C}_i := \{C'_{i1}, C'_{i2}, \ldots, C'_{il_i}, D'_{i1}, \ldots D'_{is_i} \mid \exists R_i.C'_{iv}, \ \forall R_i.D'_{iw} \in E\}$$

mit $\quad 1 \leq v \leq l_i \ , \ 1 \leq w \leq s_i$

4. *Rufe Sat* $(\mathcal{C}_{ij} \mid M\mathcal{C}_i)$ *für alle Constraintmengen* $\mathcal{C}_{ij} \in \mathcal{C}_i$ *auf und setze jeweils das Ergebnis gleich:*

$$V_{ij} := Sat\ (\mathcal{C}_{ij} \mid M\mathcal{C}_i)$$

5. *Wenn für ein* $i \in \{1, 2, \ldots, k\}$ *und ein* $j \in \{1, 2, \ldots, m_i\}$ $V_{ij} = false$ *dann ende mit Ausgabe: false, sonst führe folgende Schritte aus:*

   (a) V_{ij} *ist nun ein Booleanvektor, der auch leer sein kann. Für alle* $i \in \{1, 2, \ldots, k\}$ *mit* $V_{ij} \neq \vec{\emptyset}$ *berechne den Booleanvektor* V_i *folgendermaßen:*

   - *Für alle* $v \in \{1, 2, \ldots, l_i\}$ *setze:*

$$V_i[v] := (OR\ \ V_{i1}[v]\ V_{i2}[v]\ \ \ldots V_{im_i}[v])$$

   - *Für alle* $w \in \{1, \ldots, s_i\}$ *setze:*

$$V_i[w] := (AND\ \ V_{i1}[w]\ V_{i2}[w]\ \ \ldots V_{im_i}[w])$$

   (b) *Ersetze für alle* $i \in \{1, 2, \ldots, k\}$ *und alle* $v \in \{1, 2, \ldots,\ l_i\}$ *den Ausdruck* $\exists R_i.C'_{iv} \in E$ *durch den Wert* $V_i[v]$ *und für alle* $w \in \{1, 2, \ldots,\ s_i\}$ *ersetze den Ausdruck* $\forall R_i.D'_{iw} \in E$ *durch den Wert* $V_i[w]$ *und erhalte den Vektor* V*.*

   (c) V *ist nun ein Vektor, dessen Elemente Boole'sche Ausdrücke über* $\{0, 1\}$ *sind. Werte alle Elemente von* V *aus und ende mit Ausgabe des ausgewerteten Vektors.*

Man kann den Gesamtverlauf von Algorithmus 5.4 folgendermaßen zusammenfassen:

Erst werden alle möglichen $\sqcap$- und $\sqcup$-Regeln auf $d_0 : C_0$ angewendet. Wenn ein Widerspruch gefunden wird, terminiert der Algorithmus mit *false* (d.h. C_0 ist unerfüllbar), sonst hat man nur die $\exists$-Regel im Zusammenhang mit der $\forall$-Regel auf das aktuelle Constraintsystem S anzuwenden. Dies geschieht in der Prozedur **Some-Prozess**. In dieser Prozedur wird zuerst überprüft, ob die aktuelle Merkmalsliste leer ist. Ist das der Fall, so werden die restlichen Constraints auf Erfüllbarkeit getestet. Wenn sich kein Widerspruch ergibt, wird der leere Booleanvektor an die aufrufende Prozedur zurückgegeben. Ist die aktuelle Merkmalsliste beim Some-Prozess-Aufruf nicht leer, so wird die Prozedur **Continue** aufgerufen, welche erst durch Aufruf der Funktion **Eval** die aktuelle Merkmalsliste M so weit wie möglich zu einer neuen Liste E auswertet. Dann wird parallel zum Erfüllbarkeitstest festgestellt, ob das zugehörige Individuum die Eigenschaft $m \in E$ besitzen kann. Diese geschieht durch rekursive Zerlegung von Merkmalen in Teilmerkmale

und Auswertung durch rekursive Aufrufe der **Sat**-Prozedur. Es kann sein, daß bei einigen von diesen rekursiven Aufrufen die zugehörige Merkmalsmenge leer ist. Ein solcher Aufruf liefert den leeren Booleanvektor, falls das aktuelle Constraintsystem erfüllbar ist. Eine leere Merkmalsmenge wird erzeugt genau dann, wenn alle Rollennamen, die in der aktuellen Merkmalsliste vorkommen, in dem aktuellen Constraintsystem nicht vorkommen. Somit werden keine neuen Teileigenschaften aus der aktuellen Merkmalsmenge abgeleitet (siehe Schritt 2). Im Falle der Erfüllbarkeit wird auf jeden Fall ein Booleanvektor geliefert. Nun muß gezeigt werden, daß dieser Vektor das Gewünschte leistet.

Lemma 5.9
Der Algorithmus 5.4 ist korrekt und vollständig.

Beweis
⋆ Die Vollständigkeit und Korrektheit (bis auf Korrektheit der Informationen, die der erzeugte Booleanvektor beinhaltet) folgen unmittelbar aus Korrektheit und Vollständigkeit des *funktionalen* Algorithmus[4]. Die Hinzunahme einer Merkmalsmenge als zusätzlichen Parameter hat nach Konstruktion keine Auswirkung auf den Ablauf des *funktionalen* Algorithmus.
⋆ Es muß also nur gezeigt werden, daß im Falle der Erfüllbarkeit des anfänglichen Konzeptterms C_0 das erzeugte Gegenbeispiel $V = (b_1, b_2, \ldots, b_n)$ die folgenden Anforderungen erfüllt: V ist ein Booleanvektor über $\{0, 1\}$ und beinhaltet korrekte und vollständige Informationen. Dies ist die Aussage des Satzes 5.10. □

Daß V ein Booleanvektor über $\{0, 1\}$ ist, kann leicht durch Induktion über den Termaufbau gezeigt werden. Intuitiv ist es klar, denn die Regeln werden rekursiv so lange angewendet, bis die Constraintmengen nur aus Literalen bestehen. Dann werden alle Eigenschaften der jeweiligen Merkmalslisten durch 0 oder 1 ersetzt. Diese Werte werden dann rekursiv in die vorherigen, größeren Merkmalsmengen eingesetzt und somit wird die ursprüngliche Merkmalsmenge, deren Elemente Boole'sche Kombinationen ausgewerteter Ausdrücke sind, vollständig ausgewertet.

Satz 5.10
Die Informationen, die V beinhaltet sind korrekt und vollständig. Das heißt für eine Merkmalsmenge $M = \{E_1, E_2, \ldots, E_n\}$ und einen Konzeptterm C_0 gilt für

$$\mathit{false} \neq V := \ \mathit{Sat}\ (\{C_0\} \mid M)$$

die Äquivalenz

$$\forall\ u \in \{1, 2, \ldots, n\}\ :\ V[u] = b_u = 1 \Leftrightarrow d_0 \in E_u^{\mathcal{I}}$$

mit $d_0 \in C_0^{\mathcal{I}}$, wobei $\mathcal{I}$ das kanonische Modell ist, das durch den Algorithmus erzeugt wird (siehe Definition 2.3 und Satz 2.4).

[4] Die Vollständigkeit und Korrektheit des *funktionalen* Algorithmus ist in [19] bewiesen.

Beweis
Mit der Auswertung der Merkmalsmenge M wird erst begonnen, wenn alle möglichen $\sqcup$- und $\sqcap$-Regeln auf das Constraint $d_0 : C_0$ angewendet worden sind. Dann erhält man die folgende Constraintmenge:

$$S = \left\{ \begin{array}{l} L_1,\ L_2,\ \ldots,\ L_q, \\ \exists R_1.C_{11},\ \exists R_1.C_{12},\ \ldots,\ \exists R_1.C_{1m_1},\ \forall R_1.D_{11},\ \ldots,\ \forall R_1.D_{1n_1}, \\ \vdots \\ \exists R_k.C_{k1},\ \exists R_k.C_{k2},\ \ldots,\ \exists R_k.C_{km_k},\ \forall R_k.D_{k1},\ \ldots,\ \forall R_k.D_{kn_k}. \end{array} \right.$$

Nun kann die Behauptung durch Induktion über den Aufbau von $\mathcal{ALC}$-Konzepttermen bewiesen werden:

- Induktionsanfang:
  Sei E_u ein Literal. Nach Konstruktion des kanonischen Modells (siehe Definition 2.3 und Satz 2.4) gilt für ein positives Literal $E_u = A$ gilt

  $$d_0 \in A^{\mathcal{I}} = E_u^{\mathcal{I}} \Leftrightarrow A \in \{L_1, L_2, \ldots, L_q\}$$

  und für einen negativen Literal $E_u = \neg A$ gilt

  $$d_0 \in (\neg A)^{\mathcal{I}} = E_u^{\mathcal{I}} \Leftrightarrow A \notin \{L_1, L_2, \ldots, L_q\}$$

  Daraus folgt unmittelbar: $b_u = 1 \Leftrightarrow d_0 \in E_u^{\mathcal{I}}$

- Induktionsschluß:
  Sei $E_u = \forall R_i.D'_{iw}$ für ein $w \in \{1, \ldots, s_i\}$, dann betrachten wir zwei Fälle:

  I. $R_i \in \{R_1, \ldots, R_k\}$.
  Der Algorithmus ruft für alle Constraintmengen

  $$\begin{array}{l} \mathcal{C}_{i1} = \{C_{i1}, D_{i1}, \ldots, D_{in_i}\} \\ \vdots \\ \mathcal{C}_{ih} = \{C_{ih}, D_{i1}, \ldots, D_{in_i}\} \\ \vdots \\ \mathcal{C}_{im_i} = \{C_{im_i}, D_{i1}, \ldots, D_{in_i}\} \end{array}$$

  die Prozedur **Sat** jeweils mit der Merkmalsmenge:

  $$M\mathcal{C}_i = \{C'_{i1}, \ldots,\ \ldots,\ C'_{il_i},\ D'_{i1}, \ldots,\ D'_{iw},\ \ldots,\ D'_{is_i}\}.$$

  Da die Erfüllbarkeit der Fall und $M\mathcal{C}_i$ nicht leer ist liefern die Aufrufe Sat $(\mathcal{C}_{ih} \mid M\mathcal{C}_i)$ jeweils einen nicht leeren Booleanvektor V_{ih} für alle $h \in \{1, \ldots, m_i\}$:

  $$V_{ih} = (b_{i1}^{(h)}, \ldots,\ \ldots,\ b_{il_i}^{(h)}, b_{i1}^{\prime(h)}, \ldots\ b_{iw}^{\prime(h)}, \ldots b_{is_i}^{\prime(h)}) \in \{0,1\}^{(l_i+s_i)}$$

Jeder Vektor V_{ih} entspricht also einem Individuum $d_{ih} \in dom(\mathcal{I})$, wobei :

$$d_{ih} \in C_{ih}^{\mathcal{I}}, d_{ih} \in D_{i1}^{\mathcal{I}}, \ldots, d_{ih} \in D_{in_i}^{\mathcal{I}} \text{ und } (d_0, d_{ih}) \in R_i^{\mathcal{I}} \tag{5.1}$$

Das heißt also, d_{ih} ist ein R_i-Nachfolger von d_0 für alle $h \in \{1, \ldots, m_i\}$. Da alle Constraintmengen $\mathcal{C}_{ih}$ erfüllbar sind (sonst hätte der Alg. mit false terminiert!), gilt nach Induktionsannahme :

$$b_{iw}^{\prime(h)} = 1 \Leftrightarrow d_{ih} \in D_{iw}^{\prime} \text{ für alle } h \in \{1, \ldots, m_i\} \tag{5.2}$$

Nach Konstruktion wird b_u als Wert von E_u genau dann gleich 1 gesetzt, wenn $D_{iw}^{\prime}$ in allen Aufrufen für $\mathcal{C}_{i1}, \ldots, \mathcal{C}_{im_i}$ zusammen mit $M\mathcal{C}_i$ als 1 ausgewertet wurde. Es gilt also

$$b_u = 1 \Leftrightarrow \forall\ h \in \{1, \ldots, m_i\}\ :\ b_{iw}^{\prime(h)} = 1 \tag{5.3}$$

Aus (5.2) und (5.3) folgt

$$b_u = 1 \Leftrightarrow \forall\ h \in \{1, \ldots, m_i\}\ :\ d_{ih} \in {D_{iw}^{\prime}}^{\mathcal{I}} \tag{5.4}$$

Andererseits besteht die Menge aller R_i-Nachfolger von d_0 aus allen $d_{ih},\ 1 \leq h \leq m_i$ und diese liegen nach 5.4 in ${D_{iw}^{\prime}}^{\mathcal{I}}$. Außerdem liegen alle d_{ih} in Interpretationsmengen von $D_{i1}, \ldots, D_{in_i}$. Es folgt also:

$$b_u = 1 \Leftrightarrow b_{iw}^{\prime(h)} = 1 \Leftrightarrow d_{ih} \in {D_{iw}^{\prime}}^{\mathcal{I}} \Leftrightarrow d_0 \in E_u^{\mathcal{I}}$$

II. $R_i \notin \{R_1, \ldots, R_k\}$

Nach Konstruktion wird $E_u = \forall R_i.D_{iw}^{\prime}$ sofort mit $b_u = 1$ ausgewertet. Es gibt nach Konstruktion des kanonischen Modells (siehe Definition 2.3) keinen R_i-Nachfolger von d_0. Also folgt sofort $d_0 \in E_u^{\mathcal{I}}$.

- Sei $E_u = \exists R_i.C_{iv}^{\prime}$ für ein $v \in \{1,\ \ldots,\ l_i\}$.

I. $R_i \in \{R_1, \ldots, R_k\}$

Der Beweis verläuft fast analog wie bei Werterestriktionen mit dem Unterschied, daß $C_{iv}^{\prime}$ für mindestens einen Aufruf von Sat $(\mathcal{C}_{ih} \mid M\mathcal{C}_i)$ als 1 ausgewertet werden muß, falls b_u den Wert 1 hat.
Es gilt nach Induktionsannahme wie vorher die Äquivalenz (5.2). Nach Konstruktion gilt außerdem:

$$b_u = 1 \Leftrightarrow \text{ existiert ein } h \in \{1, \ldots, m_i\} \text{ mit } b_{iv}^{(h)} = 1$$

Daraus folgt zusammen mit (5.2)

$$b_u = 1 \Leftrightarrow \text{ existiert ein } d_{ih} \in \text{dom}(\mathcal{I}) \text{ mit } d_{ih} \in {C_{iv}^{\prime}}^{\mathcal{I}}$$

Andererseits sind nach Beziehung (5.1) alle Individuen d_{ih} ein R_i-Nachfolger von d_0. Also folgt: $b_u = 1 \Leftrightarrow d_0 \in E_u^{\mathcal{I}}$.

II. $R_i \notin \{R_1, \ldots, R_k\}$

E_u wird nach Konstruktion sofort mit $b_u = 0$ ausgewertet. Dies ist korrekt, da keine R_i-Nachfolger von d_0 erzeugt werden. Insbesondere gibt es dann ja kein $d_1 \in {C'_{iv}}^{\mathcal{I}}$ mit $(d_0, d_1) \in R_i^{\mathcal{I}}$. Es folgt also $d_0 \notin E_u^{\mathcal{I}}$.

- E_u ist eine Boole'sche Kombination über Literale und Restriktionen. Die Behauptung folgt nun direkt aus den vorherigen Induktionsschritten zusammen mit der Definition der Semantik der $\mathcal{ALC}$-Konzepte:

  - $E_u = C_1 \sqcap C_2$. Es gilt:
  $$\begin{array}{c} b_u = 1 \\ \Leftrightarrow \\ b_u \in C_1^{\mathcal{I}}, b_u \in C_2^{\mathcal{I}} \\ \Leftrightarrow \\ d_0 \in C_1^{\mathcal{I}}, d_0 \in C_2^{\mathcal{I}} \\ \Leftrightarrow \\ d_0 \in (C_1 \sqcap C_2)^{\mathcal{I}} = E_u^{\mathcal{I}} \end{array}$$

  - $E_u = C_1 \sqcup C_2$. Es gilt:
  $$\begin{array}{c} b_u = 1 \\ \Leftrightarrow \\ b_u \in C_1^{\mathcal{I}} \text{ oder } b_u \in C_2^{\mathcal{I}} \\ \Leftrightarrow \\ d_0 \in C_1^{\mathcal{I}} \text{ oder } d_0 \in C_2^{\mathcal{I}} \\ \Leftrightarrow \\ d_0 \in (C_1 \sqcup C_2)^{\mathcal{I}} = E_u^{\mathcal{I}} \end{array}$$

  - $E_u = \neg C$
  $$b_u = 1 \Leftrightarrow b_u \notin C^{\mathcal{I}} \Leftrightarrow d_0 \notin C^{\mathcal{I}} \Leftrightarrow d_0 \in E_u^{\mathcal{I}}$$

□

Daß der Experte in PSPACE (in der Größe der Tbox) liegt, kann ähnlich wie bei dem funktionalen Algorithmus begründet werden (siehe Bemerkung 2.11). Es ist nur noch zu beachten, daß expandierte Konzeptterme und somit auch expandierte Merkmale (in der Tbox definierte Konzeptnamen) expontiell (in der Größe der Tbox) wachsen können. Dieses exponentielle Hindernis kann dadurch überwunden werden, daß man nur bei Bedarf expandiert. Man versucht, die Widersprüche möglichst in höheren Stufen des Constraintbaumes zu entdecken. D.h. man geht nicht von einer vollständig expandierten Tbox und Eingabe aus, sondern es wird nur bei Bedarf aufgefaltet und ebenso nur bei Bedarf die Negation eingeschoben. Auch der zweite Eingabeparameter (Merkmalsliste) wird bei jedem Aufruf um einen Schritt aufgefaltet. Somit bleibt man immer in PSPACE. Daß es möglich ist, nur bei Bedarf zu expandieren, soll hier an einem Beispiel verdeutlicht werden.

Wir betrachten die Familien-Terminologie beschrieben in 3.1, wobei noch neue Konzeptnamen folgendermaßen definiert werden:

$$\begin{array}{lcl} \text{Hat-einen-Sohn} & = & \exists \text{hat-kind.Männlich} \\ \text{Mutter} & = & \exists \text{hat-kind.Mensch} \\ \text{Mutter-ohne-Sohn} & = & \text{Mutter} \sqcap \forall \text{hat-kind.}\neg\text{Männlich} \end{array}$$

Es werden dann ausgehend von der Eingabe {Hat-einen-Sohn $\sqcap$ Mutter-ohne-Sohn} folgende Schritte durchgeführt:

$$\begin{array}{ll} \rightarrow_{\sqcap} & \{\text{Hat-einen-Sohn, Mutter-ohne-Sohn}\} \\ \rightarrow_{\exists} & \{\exists \text{hat-kind.Männlich, Mutter} \sqcap \forall \text{ hat-kind.}\neg \text{ Männlich}\} \\ \rightarrow_{\sqcap} & \{\exists \text{hat-kind.Männlich, } \forall \text{ hat-kind.}\neg \text{ Männlich, Mutter}\} \\ \rightarrow_{\sqcap,\exists,\forall} & \{\text{Männlich, } \neg \text{ Männlich, Mutter}\} \\ \rightarrow & \textit{Clash} \end{array}$$

Wir sehen, daß hier ein *Clash* entdeckt wurde, ohne daß die definierten Konzeptnamen Mutter und Männlich expandiert wurden. Diese Optimierung bringt in der Praxis, wo die Konzeptdefinitionen tief geschachtelt sind, sehr viel. Außerdem kann man die schon ganz oder teilweise expandierten Konzeptnamen in einer lokalen Variable zwischenspeichern, um sie bei Bedarf wieder zu verwenden. Solche Optimierungsmethoden sind in [21] diskutiert.

Kapitel 6

Optimierungen

Bezüglich der Implementierung des Experten sind schon einige Optimierungen erwähnt worden. Zum Beispiel gehören die Expandierung der Konzeptterme und die Einschiebung der Negation nur bei Bedarf zu diesen Optimierungen. Trotzdem bleibt die Aufgabe des Experten ein hartes Problem. Ein Expertenaufruf ist also sehr teuer. Eine wichtige Optimierung liegt in der Minimierung der Anzahl dieser Aufrufe. Während der Erzeugung der Minimalbasis durch den Algorithmus 4.45 werden u.a. Implikationen erzeugt, die sich später als überflüssig erweisen. Das heißt, sie leisten nichts für die Erzeugung der gesuchten Basis. Die erzeugten Implikationen kann man in drei Klassen aufteilen:

1. Die trivialen Implikationen. Sie sind von der Form $B \to \emptyset$ und sind somit in jedem Kontext erfüllt. Diese Implikationen sind, einzeln betrachtet, harmlos, da sie keinen expliziten Expertenaufruf verursachen. Sie veranlassen aber die größte Schwierigkeit während der Laufzeit des Algorithmus 4.45, weil sie viel zu oft erzeugt werden. Eine Implikation $A \to B$ gehört genau dann zu dieser Klasse, wenn ihre Prämisse ein Begriffsinhalt des aktuellen Kontextes ist (d.h. $A = A''$). Deshalb leisten solche Implikationen nichts zu der Vollständigkeit der gesuchten Basis (die Basis besteht nur aus Implikationen, deren Prämisse ein Pseudoinhalt des unendlichen Kontextes ist). Außerdem erweitern solche Implikationen auch den Kontext nicht, da sie immer gelten. Ob man diese Art von Implikationen vermeiden oder mindestens minimieren kann ist ein offenes Problem von großer Bedeutung. Dieses Problem werden wir in Kapitel 7 genauer diskutieren.

2. Die Implikationen $P \to P'' \setminus P$, deren Prämissen P Pseudoinhalte des endgültigen Kontextes sind. Sie sind genau die gesuchten Implikationen, die die Minimalbasis bestimmen.

3. Die Implikationen $B \to B'' \setminus B$, deren Prämissen B zwar Pseudoinhalte des aktuellen Kontextes K_i sind, aber kein Pseudoinhalt des endgültigen Kon-

textes K. Sie sind in K nicht erfüllt und erweitern den aktuellen Kontext um ein neues Objekt (Gegenbeispiel).

Die zweite und dritte Art von Implikationen sind teuer, weil für sie der Experte aufgerufen wird. Man fragt sich nun, inwiefern sind sie notwendig? Mit anderen Worten, gibt es auch nichttriviale Implikationen, die bzgl. des Erzeugungsprozesses der Minimalbasis gar nicht relevant sind?
Die zweite Art von Implikationen sind notwendig. Sie bestimmen die gesuchte Basis und müssen früher oder später erzeugt werden. Auch ein Teil der dritten Implikationsklasse ist notwendig, da er Objekte erzeugt, die die Bestimmung der nächsten Pseudoinhalte ermöglichen. Der andere Teil dieser Klasse ist eigentlich irrelevant, da der erschlossene Kontext nicht unbedingt reduziert sein muß (auf die reduzierten Kontexte wird genauer eingegangen). Man stellt sich nun die Frage: Wie ist es möglich, diese Art von Implikationen zu vermeiden? Das heißt, wie kann man dafür sorgen, daß genau die notwendigen Objekte erzeugt werden?
Um diese Frage zu beantworten, muß man zuerst solche irrelevanten Objekte bzw. Implikationen charakterisieren.

6.1 Charakterisierung von irrelevanten Objekten

Definition 6.1
Ein Kontext ist **bereinigt**, *wenn es im Kontext keine Gegenstände mit gleichem Inhalt und keine Merkmale mit gleichem Umfang gibt. Formal:*

$$\forall m_1, m_2 \in M \text{ und } g_1, g_2 \in O : g_1' \neq g_2' \text{ und } m_1' \neq m_2'.$$

Definition 6.2
Sei $K = (M, O, \rho)$ ein Kontext. Ein Gegenstand $g \in O$ heißt **reduzibel**, *wenn es eine Teilmenge $H \subseteq G$ gibt so, daß*

$$\forall h \in H : \ g' \subset h' \ \text{ und außerdem } g' = \bigcap_{h \in H} h'$$

Entsprechend heißt ein Merkmal $m \in M$ reduzibel, wenn es eine Teilmenge $N \subseteq M$ gibt so, daß

$$\forall n \in N : \ m' \subset n' \ \text{ und außerdem } \ m' = \bigcap_{n \in N} n'.$$

Ein Kontext heißt dann reduziert, wenn er bereinigt ist und keine reduziblen Gegenstände oder Merkmale enthält.

Bemerkung 6.3
Direkt aus der Definition folgt, daß ein $g \in O$ genau dann reduzibel ist, wenn sein Inhalt g' sich als Durchschnitt mindestens zwei anderer Zeileninhalte darstellen läßt. D.h. :

$$g \in O \text{ reduzibel}$$
$$\Leftrightarrow$$
$$\text{es ex. } g_1,\ g_2,\ \ldots,\ g_r \in O,\ 2 \leq r,\ g_i \neq g_j \text{ für } i \neq j \text{ und } g' = g'_1 \cap g'_2 \ldots \cap g'_r$$

.
Entsprechend ist ein Merkmal genau dann reduzibel, wenn sich sein Spaltenumfang als Durchschnitt mindestens zwei anderer Spaltenumfänge darstellen läßt.

Definition 6.4
Sei $(V, \leq)$ ein vollständiger Verband. Ein $v \in V$ heißt dann $\bigvee$-irreduzibel[1] *,wenn sich v nicht als Supremum echt kleinerer Elemente darstellen läßt, d.h. $v_\downarrow = \bigvee\{x \in V \mid x < v\} \neq v$. Entsprechend heißt v $\bigwedge$-irreduzibel*[2] *, wenn sich v nicht als Infimum echt größerer Elemente darstellen läßt. d.h. $v^\uparrow = \bigwedge\{x \in V \mid v < x\} \neq v$. Die Menge aller $\bigvee$-irreduziblen Elemente wird mit $J(V)$ und aller $\bigwedge$-irreduziblen Elemente mit $M(V)$ bezeichnet.*
Eine Menge $X \subseteq V$ heißt **supremum-dicht** *in V, wenn jedes Element von V als Supremum einer Teilmenge von X dargestellt werden kann, und dual* **infimum-dicht**, *falls für alle $v \in V$ gilt: $v = \bigwedge\{x \in X \mid v \leq x\}$.*

Satz 6.5
Ein Element v eines endlichen Verbandes ist genau dann $\bigvee$-irreduzibel, wenn es genau einen unteren Nachbarn (siehe Definition 4.2) hat und genau dann $\bigwedge$-irreduzibel, wenn es genau einen oberen Nachbarn hat. Jede supremum-dichte Teimenge von V enthält alle $\bigvee$-irreduziblen Elemente und jede infimum-dichte Teilmenge enthält alle $\bigwedge$-irreduziblen Elemente. In einem endlichen Verband ist umgekehrt die Menge $J(V)$ supremum-dicht und die Menge $M(V)$ infimum-dicht.

Beweis
Nach Definition ist v genau dann $\bigvee$-irreduzibel, wenn $v \neq v_\downarrow$ ist. Dies wiederum ist gleichbedeutend damit, daß $v_\downarrow$ das größte unter allen Elementen ist, die kleiner als v sind. Daraus folgt insbesondere, daß $v_\downarrow$ der einzige untere Nachbar von v ist. Analog beweist man die Aussage für $\bigwedge$-irreduzible Elemente. Die zweite Aussage des Satzes ist trivial, die dritte beweist man induktiv: Jedes Element v, welches nicht selbst $\bigvee$-irreduzibel ist, ist Supremum echt kleinerer Elemente. Sind diese Suprema $\bigvee$-irreduzible Elemente, so auch v. □

Es lassen sich leicht unendliche Verbände angeben, in denen jedes Element reduzibel ist. Zum Beispiel ist das reelle Intervall [0,1] mit der natürlichen Ordnung ein

[1] lies Supremum-irreduzibel (join irreducible)
[2] lies Infimum-irreduzibel (meet irreducible)

solcher Verband, denn jedes Element hat sicher mehr als einen unteren bzw. oberen Nachbarn und somit kann es nach Satz 6.5 kein irreduzibles Element sein. Man kann also nicht ohne weiteres simultan alle reduziblen Elemente eines Verbandes weglassen, ohne die Struktur des Verbandes zu ändern. Dies ist aber für endliche Verbände kein Problem, da bei endlichen Verbänden jedes Element Supremum einer Menge von $\bigvee$-irreduziblen und Infimum einer Menge von $\bigwedge$-irreduziblen Elementen ist. Denn nach Satz 6.5 sind die Mengen aller $\bigvee$-irreduziblen ($J(V)$) bzw. aller $\bigwedge$-irreduziblen ($J(V)$) Elemente eines endlichen vollständigen Verbandes (V) supremum-dicht bzw. infimum-dicht.

Lemma 6.6
Sei $K = (O, M, \rho)$ ein Kontext. Ein Gegenstand $g \in O$ ist genau dann reduzibel, wenn der Gegenstandsbegriff $\gamma(g)$ im Begriffsverband von K $\bigvee$-reduzibel ist. Entsprechend ist ein Merkmal $m \in M$ genau dann reduzibel, wenn der Merkmalsbegriff $\mu(m)$ im Begriffsverband des Kontextes K $\bigwedge$-reduzibel ist.

Beweis
Nach Definition 6.4 gilt:

$$\gamma(g) = (g'', g') \text{ ist } \bigvee\text{-reduzibel}$$
$$\Leftrightarrow$$
$$(g'', g') = \bigvee\{(A_i'', A_i') \mid A_i \subseteq O,\ (A_i'', A_i') < (g'', g')\} = \bigvee X.$$

Die Menge aller zugehörigen Indizes bezeichnen wir als I.

- „$\Leftarrow$“
  Sei $\gamma(g)$ reduzibel im Begriffsverband von K. Nach Hauptsatz 4.23 und obiger Äquivalenz folgt dann $g' = \bigcap A_i'$ und $g'' = (\bigcup A_i'')''$. Es gilt nach Hilfssatz 4.51 die Gleichung $\bigcap A_i' = (\bigcup A_i)'$. Man setze nun $(\bigcup A_i) = H \subseteq G$. Somit ergibt sich $g' = \bigcap\limits_{h \in H} h'$. Andererseits ist nach Definition 4.22 g' echt in allen A_i' enthalten. Also folgt $g' \subset h'$ für alle $h \in H$. Somit ist g in K nach Definition 6.2 reduzibel.

- „$\Rightarrow$“
  Sei g reduzibel in K. Nach Hauptsatz 4.23 gilt

  $$\bigvee X = \left((\bigcup_{i \in I} A_i'')'' \ , \ \bigcap_{i \in I} A_i' \right),$$

  d.h. das Supremum von X ist der kleinste gemeinsame Oberbegriff, dessen Merkmale genau die gemeinsamen Merkmale aller Begriffe aus X sind.
  Da g reduzibel in K ist, gibt es nach Definition 6.2 ein $H \subseteq O$ mit $g' = \bigcap\limits_{h \in H} h'$ und $g' \subset h'$ für alle $h \in H$. Somit liegen nach Definition 4.22 die Begriffe (h'', h') für alle $h \in H$ in X. Dies bedeutet $\bigcap\limits_{i \in I} A_i' \subseteq \bigcap\limits_{h \in H} h' = g'$.

Es folgt dann nach Definition 4.22 $(g'', g') \leq \bigvee X$, was sofort $(g'', g') = \bigvee X$ impliziert. Daraus und aus der obigen Äquivalenz folgt, daß $\gamma(g)$ im Begriffsverband von K reduzibel ist. Der Beweis für reduzible Merkmale ist analog.

□

Lemma 6.7
Die Entfernung von reduziblen Elemente eines Kontextes mit endlicher Merkmalsmenge hat keinen Einfluß auf die Struktur des Begriffsverbandes des Kontextes.

Beweis
Sei $K = (O, M, \rho)$ ein endlicher Kontext und $g \in O$ ein reduzibles Element. Nach Definition 6.2 gibt es eine Menge $H \subseteq O$ von Gegenständen mit $g \notin H$, aber $g' = H'$. Andererseits ist Nach Lemma 6.6 g in dem Kontext K genau dann reduzibel, wenn $\gamma(g)$ $\bigvee$-reduzibel im Begriffsverband des Kontextes ist. Somit ist der Gegenstandsbegriff $\gamma(g)$ das Infimum der Gegenstandsbegriffe $\gamma(h),\ h \in H$ und wegen der Endlichkeit des Begriffsverbandes von K ist die Menge $\gamma(O \setminus \{g\})$ nach Satz 6.5 supremum-dicht im Begriffsverband $\mathcal{V}(O, M, \rho)$ des Kontextes K. Somit genügt die Abbildung γ den Bedingungen, die im Hauptsatz 4.24 gefordert sind. Also folgt nach diesem Satz:

$$\mathcal{V}(O, M, \rho) \cong \mathcal{V}(O \setminus \{g\}, M, \rho \cap ((O \setminus \{g\}) \times M)).$$

□

Diese Aussage bedeutet, daß man bei einem Kontext mit endlich vielen Merkmalen die reduziblen Elemente entfernen kann, ohne die Struktur des Begriffsverbandes zu zerstören. Ganter und Wille haben Methoden angegeben (vgl. [28] Seiten 24-32), mit deren Hilfe man reduzible Elemente eines vorliegenden Kontextes eliminieren kann. Diese Methoden sind für uns aber weniger interessant, denn wir möchten die Erzeugung solcher Elemente von Anfang an vermeiden.

Lemma 6.8
Der Kontext K, der während der Laufzeit des Algorithmus 4.45 durch den Experten 5.4 erschlossen wird, ist zeilen-bereinigt, d.h. es existiert keine verschiedenen Gegenstände mit demselben Inhalt.

Beweis
Für $g_1, g_2 \in O$ würde $g_1' = g_2'$ bedeuten, daß Gegenbeispiele mit wiederholten Inhalt erzeugt werden. Dies kann aber nicht der Fall sein, denn jede erzeugte Implikation die Form $B \to B'' \setminus B$ hat und ist somit vom aktuellen Kontext K_i erfüllt. Wenn nun eine Implikation einmal erzeugt wurde und zum Gegenbeispiel g_{i+1} geführt hat, kann sie nicht mehr von einem $K_j,\ j > i$ erfüllt werden, also wird sie nicht wieder erzeugt. □

Es ist zu beachten, daß der erschlossene Kontext nicht unbedingt spalten-bereinigt ist. Es kann nämlich sein, daß in der Tbox Konzeptnamen A_1, A_2 definiert sind, die zwar verschieden sind aber gleich interpretiert werden. Dieser Kontext ist ebenfalls nicht unbedingt merkmalsreduziert, denn es kann sein, daß in der Tbox Konzeptnamen $A, A_1, \ldots A_r$ definiert sind, so daß A und $A_1 \sqcap \ldots \sqcap A_r$ gleich interpretiert werden (logisch äquivalente Konzeptnamen), d.h. der Spaltenumfang zu A läßt sich als Schnitt bestimmter Spaltenumfänge darstellen. Man könnte einen spalten-bereinigten bzw. spalten-reduzierten Kontext erzielen, wenn man solche logischen Äquivalenzen ausschließt. Dies erfordert aber einen möglicherweise erheblichen Aufwand. Eine solche, wahrscheinlich teure, Maßnahme ist nicht sehr interessant, da der teure Experte nicht für die Erzeugung von Merkmalen, sondern von Gegenständen aufgerufen wird. Eine wichtige Optimierung des Verfahrens liegt also darin, die Erzeugung von reduziblen Gegenbeispielen zu verhindern. Man fragt sich nun: Gibt es auch unter irreduziblen Elementen welche, die irrelevant sind?
Man benötigt auf jeden Fall eine minimale Anzahl von Gegenbeispielen, um alle Pseudoinhalte bestimmen zu können. Kann diese Minimalität genau durch die Menge aller irreduziblen Gegenstände charakterisiert werden? Der Satz 6.9 aus [28] und das Lemma 6.10 bejahen diese Frage.

Satz 6.9
Zu einem endlichen Verband $(V, \leq)$ gibt es bis auf Isomorphie genau einen reduzierten Kontext, dessen Begriffsverband isomorph zu V ist, nämlich:

$$\mathcal{K}(V) = (J(V), M(V), \leq)$$

Lemma 6.10
Die Entfernung eines irreduziblen Elementes eines Kontextes mit endlicher Merkmalsmenge ändert die Struktur seines Begriffsverbandes.

Beweis
Sei $K = (O, M, \rho)$ ein Kontext mit endlicher Merkmalsmenge und dem Begriffsverband $\mathcal{B} = \mathcal{V}(O, M, \rho)$. Sei weiter $g \in O$ ein irreduzibles Element. Nach Lemma 6.6 ist dann $\gamma(g) \in J(\mathcal{B})$ (d.h. der Gegenstandsbegriff $\gamma(g)$ ist im Begriffsverband von K irreduzibel). Sei $\mathcal{B}' = (\mathcal{V}(O \setminus \{g\}, M, \rho')$ mit $\rho' = \rho \cap O \setminus \{g\} \times M$. Angenommen es gilt $\mathcal{B} \cong \mathcal{B}'$. Wegen $\gamma(g) \in J(\mathcal{B})$ folgt dann $J(\mathcal{B}') = J(\mathcal{B}) \setminus \{\gamma(g)\}$. Nach Satz 6.7 sind dann die Verbände $\mathcal{V}(K(\mathcal{B})$ und $\mathcal{V}(K(\mathcal{B}')$ isomorph. Es gilt wegen der Annahme und Satz 6.9 die Isomorphie $\mathcal{V}(K(\mathcal{B}) \cong \mathcal{B}' \cong \mathcal{B}$. Somit sind die Kontexte $K(\mathcal{B})$ und $K(\mathcal{B}')$ zwei reduzierte Kontexte, deren Verbände isomorph zu dem vollständigen Verband $\mathcal{B}$ sind. Sie müssen dann nach Satz 6.9 isomorphe Kontexte sein. Das ist aber nicht möglich, da sie endliche, aber nicht gleich große Objektsmengen besitzen. □

Zusammen mit Lemma 6.7 folgt aus dieser Aussage, daß die Struktur des Begriffsverbandes eines Kontextes genau durch die Menge aller irreduziblen Elemente dieses Kontextes bestimmt wird. Man wünscht sich also, daß während

der Laufzeit des Algorithmus 4.45 genau die irreduziblen Gegenbeispiele erzeugt werden[3].
Es ist möglich, dies zu erreichen, indem man den Experten dazu zwingt, nur maximale Gegenbeispiele zu erzeugen. Diese sind Gegenbeispiele, die um kein weiteres Merkmal erweitert werden können. Das heißt, diejenigen Gegenstände, deren Gegenstandsinhalte in keinem anderen Gegenstandsinhalt enthalten sind. Nach Bemerkung 6.3 ist dann klar, daß ein solches Gegenbeispiel irreduzibel ist, denn es kann nie Durchschnitt anderer Zeileninhalte sein. Formal:

Definition 6.11
Das Gegenbeispiel $V = (b_1, \ldots, b_n)$, das durch den Experten zur Implikation $A \to B$ erzeugt wird, heißt maximal, falls es unter allen möglichen Gegenbeispielen zur Implikation $A \to B$ die maximale Anzahl von True-Werten enthält.

Die Erzeugung von maximalen Gegenständen führt also zur Reduzierung der Anzahl der Expertenaufrufe. Diese Optimierung muß natürlich so sein, daß sich dadurch der Zeitaufwand für die einzelnen Expertenaufrufe nicht zu stark vergrößert.
Es ist möglich, maximale Gegenbeispiele zu erzeugen. Man kann nämlich den Experten so modifizieren, daß zu einer gegebenen Implikation immer ein maximales Gegenbeispiel erzeugt wird. Aber das ist leider mit einem erheblichen Mehraufwand verbunden.

6.1.1 Ein Vorschlag zur Erzeugung von maximalen Gegenbeispielen

Man kann maximale Gegenbeispiele folgendermaßen erzeugen:
Sei C der Konzeptterm, dessen Erfüllbarkeit zu testen ist. Nach Algorithmus 5.4 setzt man alle primitiven Konzeptnamen, die nicht in der Constraintmenge $\{C\}$ vorkommen, auf *false*. Ebenso werden die Ausdrücke von der Form $\exists R.D$ auf *false* gesetzt, wenn der Rollenname R in keinem existentiellen Term im Constraintsystem vorkommt. Dagegen werden Ausdrücke von der Form $\forall R.D$ auf *true* gesetzt, wenn R nicht in einem existentiellen Ausdruck im Constraintsystem vorkommt. Solche Ausdrücke müssen aber nicht unbedingt mit diesen Werten belegt werden. Zum Beispiel betrachten wir den folgenden Aufruf:

$$Sat(\{\exists R.C, \forall R.D\} \mid [A, \ldots, \exists R'.C'_1, \forall R'.D'_1, \ldots, \exists R'.C'_2, \forall R'.D'_2, \ldots])$$

wobei der primitive Name A nicht im aktuellen Constraintsystem liegt und $R \neq R'$ ist. Es ist möglich, die Ausdrücke $A, \exists R'.C'_1, \exists R'.C'_2$ nicht wie früher auf *false*, sondern auf *true* und ebenso die Ausdrücke $\forall R'.D'_1, \forall R'.D'_2$ nicht auf *true*, sondern auf *false* zu setzen. Hierbei muß man zwei Probleme in Betracht ziehen.

[3]Wie man aus der Tabelle 7.4 entnehmen kann, führt die Wahl der irreduziblen Gegenbeispiele in der Tat zur Reduzierung der Anzahl von Expertenaufrufen.

Erstens muß festgestellt werden, welche Auswirkungen diese Belegung auf die oberen Knoten des Baumes hat, d.h. welche Belegung dieser Ausdrücke eine maximale Anzahl von True-Werten in der gesamten ursprünglichen Merkmalsmenge erzeugt. Dieses Problem kann man durch Einführung von Boole'schen Variablen für Teileigenschaften, deren Werte frei wählbar sind, lösen. Somit erhält man als Ausgabe eine Liste von Boole'schen Ausdrücken, die aus Boole'schen Werten und aussagenlogischen Variablen bestehen. Dann findet man eine Belegung, die eine Liste mit einer maximalen Anzahl von True-Werten liefert. Das zweite Problem, das zu beachten ist, ist, daß die Constraintmengen

$$\{C_1', D_1', D_2'\} \text{ und } \{C_2', D_1', D_2'\}$$

erfüllbar sein müssen, wenn man die Ausdrücke auf *true* setzen möchte. Also muß man zusätzliche Erfüllbarkeitstests durchführen. Die Anzahl dieser Tests kann aber offensichtlich exponentiell groß werden. Also ist diese Lösung zwar eine exakte Lösung, aber keine gute. Ein Vorschlag wäre dann, sich alle Constraintmengen, deren Erfüllbarkeit während früherer **Sat**-Aufrufe getestet wurden, zu merken. Dies erfordert natürlich zusätzlichen Speicher, was dem PSPACE-Charakter des Algorithmus zuwider läuft. Die Güte dieses Verfahrens muß durch genauere Überlegung und Implementierung mit statistischer Auswertung festgestellt werden.
Eine andere Möglichkeit ist, Heuristiken einzusetzen, die weniger aufwendig sind, aber nicht unbedingt maximale Beispiele erzeugen. Auf dieses Thema wurde in dieser Arbeit nicht weiter eingegangen, denn bei der statistischen Auswertung mittels eines implementierten Programms hat sich ein wesentlich größeres Problem herausgestellt, dessen Lösung (falls es überhaupt lösbar ist) im Vordergrund steht. Dies führt zum Problem der trivialen Implikationen, welches wir in den nächsten Abschnitten diskutieren.

6.2 Minimierung der Anzahl der Iterationsdurchläufe

Wie die experimentellen Ergebnisse zeigen (siehe Abschnitt 7.2) ist, die Anzahl der Itterationsdurchläufe[4] so groß, daß der Zeitaufwand bei etwas größeren Tboxen nicht mehr vertretbar ist. Dafür ist die große Anzahl der Begriffe verantwortlich. Dies führt dazu, daß während der Laufzeit eine enorme Anzahl von trivialen Implikationen erzeugt wird, welche weder den Kontext, noch die Basis erweitern und somit den Aufbau der Basis verzögern. Folgende Überlegungen zeigen, daß die Erzeugung der maximalen Gegenbeispiele, wie es vorher diskutiert wurde, die einzige Optimierungsmöglichkeit bzgl. der Reduzierung der Anzahl der Durchläufe ist. Allerdings bringt diese Optimierung nicht sehr viel, denn die

[4] Hier ist mit einem Durchlauf die Ausführung des Schrittes (2) bei Algorithmus 4.45 gemeint.

Anzahl von erzeugten trivialen Implikationen kann dadurch nicht beeinflußt werden und genau diese Implikationen machen den größten Anteil der Anzahl von Durchläufen aus.

Lemma 6.12
Sei $K = (O, M, \rho)$ ein Kontext und $B \subseteq M$ ein Begriffsinhalt von K. Wird nun K um einen neuen Gegenstand g erweitert, so ist B ein Begriffsinhalt des erweiterten Kontextes.

Beweis
Nach Lemma 4.16 ist B ein Begriffsinhalt von K genau dann, wenn $B = B''$ in K gilt. Ist $B \not\subseteq g'$, so bleibt B'' im neuen Kontext unverändert, also gilt immer noch $B = B''$ im erweiterten Kontext und somit ist B ein Begriffsinhalt. Ist $B \subseteq g'$, so ist die neue Menge B'' gleich der Schnittmenge zwischen dem alten B'' und g', also folgt: neu-$B'' = B'' \cap g' = B \cap g' = B$. Also ist B auf jeden Fall ein Begriffsinhalt des erweiterten Kontextes. □

Lemma 6.13
Die Anzahl der trivialen Implikationen, die während der Laufzeit des Algorithmus 4.45 erzeugt werden, ist genau gleich der Anzahl der Begriffsinhalte, die der endgültige Kontext K besitzt.

Beweis
Alle erzeugten Implikationen haben die Form $B_i \to B_i'' \setminus B_i$, also ist eine Implikation genau dann trivial, wenn bzgl. des aktuellen Kontextes K_i die Gleichung $B_i = B_i''$ gilt, das heißt, wenn B_i ein Begriffsinhalt von K_i ist. Ist B_i ein Begriffsinhalt des aktuellen Kontextes K_i, so ist B_i nach Lemma 6.12 auch ein Begriffsinhalt des endgültigen Kontextes K. Also hat man mit jeder trivialen Implikation einen Begriffsinhalt des endgültigen Kontextes berechnet. Andererseits sei $A \subseteq M$ ein Begriffsinhalt von K. Da der Algorithmus alle Begriffs- und Pseudoinhalte des endgültigen Kontextes in lektischer Ordnung berechnet, muß A als linke Seite einer Implikation $A \to A_i \setminus A$ erzeugt worden sein, wobei $A_i = A''$ im aktuellen Kontext K_i gilt. Nach Konstruktion bleibt A die linke Seite der nächsten Implikationen, so lange diese vom Experten verneint werden. Man kommt also von A erst weg, wenn eine in K gültige Implikation $A \to A_j \setminus A,\ j \geq i$ erzeugt wird. Diese Implikation ist trivial, denn wäre dies nicht der Fall, so wäre A ein Pseudoinhalt von K und könnte somit kein Begriffsinhalt sein. □

Aus diesem Lemma folgt sofort:

$$\#\mathit{Iterationsdurchläufe} = \#\mathit{Gegenbeispiele} + \mathit{Basislänge} + \#\mathit{Begriffe}$$

Demzufolge ist, was die Reduzierung der Durchläufe betrifft, die Minimierung der erzeugten Gegenbeispiele die einzige mögliche Optimierung. Denn die *Basislänge* und *#Begriffe* sind durch die eindeutige Menge aller Pseudo- bzw. Begriffsinhalte

des Kontextes festgelegt.
Wie mehrere Experimente zeigen (siehe 7.2), haben die Kontexte sehr viele Begriffe, was einen enormen Zeitaufwand zur Folge hat.
Dieses Problem wäre gelöst, wenn man den Algorithmus so modifizieren könnte, daß die Pseudoinhalte mit einem vertretbaren Aufwand direkt, d.h. nicht über Begriffsinhalte, berechnet werden. Dies wäre eine erhebliche Modifikation und derzeit ist nicht klar, ob eine derartige, modifizierte Version überhaupt mit einem vertretbaren Aufwand zu realisieren ist.

Kapitel 7

Implementierung

Im wesentlichen wurden die diskutierten Algorithmen, insbesondere Algorithmus 4.45 und 5.4 implementiert. Dabei wurden eine Reihe von Sortier- bzw. Suchfunktionen geschrieben, die als Hilfsfunktionen dienen. Für die Implementierung wurde die Programmiersprache *Lisp* gewählt, weil sie zum einen wegen ihrer funktionalen Eigenschaft für Implementierung komplizierter, mathematisch orientierter Aufgaben hervorragend geeignet ist und zum anderen wegen automatischer Garbage-Collection die Behandlung dynamischer Datenstrukturen wesentlich erleichtert. Es wurden logisch zusammenhängende Funktionen in einer separaten Lisp-Datei geschrieben und als sogenannte ,,USER-PACKAGE" gebunden. Mit dem Laden der Datei ,,Boot.lisp" werden dann alle notwendigen Funktionen geladen. Bei der Implementierung wurde versucht, die Funktionen möglichst *endrekursiv* zu implementieren, was Speicher spart und die Ausführungsgeschwindigkeit steigert. Im folgenden werden die wichtigsten Datenstrukturen und Funktionen kurz beschrieben und anschließend werden die experimentellen Werte, die durch mehrere Versuche mittels des implementierten Programms ermittelt worden sind, diskutiert.

7.1 Programmbeschreibung

$\mathcal{ALC}-$Ausdrücke: Ein solcher Ausdruck wurde als eine Liste dargestellt, deren Kopf aus einem Konstruktor (**some, all, und, oder, nicht**) besteht. Die Restriktionen entsprechen drei-elementigen Listen, deren Kopf dem Konstruktor some bzw. all, das zweite Element einem Rollennamen und das dritte Element einem syntaktisch korrekten $\mathcal{ALC}-$Ausdruck entspricht, z.B. (**some** hat-kind Mutter) oder (**all** hat-geschwister Männlich). Entsprechend wurden $\sqcap$- bzw. $\sqcup$-Ausdrücke als beliebig lange Listen mit dem Kopf **und** bzw. **oder** und dem Rest als eine Folge von syntaktisch korrekten Ausdrücken dargestellt, z.B. (**und** Weiblich Alleinerziehend Berufstätig). Negations-Ausdrücke entsprechen einer zwei-elementigen Liste mit dem Kopf

nicht und einem syntaktisch korrekten Ausdruck als zweitem Element der Liste, z.B. (**nicht** Vater).

Tbox: Ein Axiom entspricht einer zwei-elementigen Liste. Der Kopf ist ein definierter Konzeptname und das zweite Element ist ein syntaktisch korrekter $\mathcal{ALC}$-Ausdruck, z.B. (Elternteil (**oder** Mutter Vater)). Eine Tbox ist dann eine endliche Liste von Axiomen.

Wissensbasis: Entspricht einer drei-elementigen Liste. Der Kopf ist eine Tbox, das zweite Element ist eine Liste von primitiven Namen und das dritte Element eine Liste von Rollennamen.

Im folgenden werden die wichtigsten Funktionen kurz beschrieben:

- (**Subsumtionshierarchie** wb): Diese Funktion erhält als Eingabe eine Liste als Wissensbasis (wb) und bestimmt die Initialparameter für den Aufruf der Funktion (**dgb** akt-inhalt akt-basis akt-kontext). Die Fkt. **dgb** ist eine rekursive Funktion, die bis auf bestimmte Optimierungsmaßnahmen, genau nach Algorithmus 4.45 implementiert ist. Sie bestimmt die aktuelle Implikation mit Hilfe der Funktion **hull-operator** und ruft damit die Funktion **expert** auf. Anschließend wird die Basis oder der Kontext erweitert, je nach dem, was der Experte liefert. Im Falle einer gültigen Implikation wird dann der nächste Inhalt durch die Funktion **next-closed** berechnet. Dabei wird, außer dem nächsten Begriff- oder Pseudoinhalt, der Wert eines Zählers zurückgegeben, der signalisiert, ob der berechnete Inhalt der lektisch größte Inhalt ist. Wenn dies der Fall ist, terminiert die Rekursion mit der Ausgabe der aktuellen Basis, sonst wird die Rekursion mit neuen Parametern fortgesetzt.

- (**expert** impl): Die Implementation des Experten entspricht, bis auf bestimmte Optimierungspunkte, dem Algorithmus 5.4. Es wird versucht einen *Clash* möglichst in höheren Baumstufen zu entdecken, deshalb wird nur bei Bedarf und schrittweise aufgefaltet. Ebenso geschieht der Übergang zur **NNF** schrittweise. Aufgefaltete Namen werden in einer lokalen Liste gespeichert, wodurch sich der Auffaltungsprozess beschleunigt. Die Funktion **expert** erhält als Eingabe eine Implikation (C, D). Falls die rechte Seite leer ist (dies entspricht der Subsumtion $C \sqsubseteq \top$), so liefert sie sofort **nil**, was dem Booleanwert *false* entspricht. Dies heißt, daß die Subsumtion gilt und demzufolge gibt es kein Gegenbeispiel für die Eingabe. Falls die Implikation nichttrivial ist, wird intern der Ausdruck $C \sqcap \neg D$ gebildet und als Parameter der Funktion **gilt** übergeben. Sie bildet daraus ein anfängliches Constraintsystem $\mathcal{S}_0 = \{C \sqcap \neg D\}$ und ruft die rekursive Funktion (**Sat** $\mathcal{S}_0$ ML) auf, wobei ML eine Liste aller definierten Konzeptnamen

ist. Die nichtrekursive Funktion **gilt** scheint überflüssig zu sein, da man das anfängliche Constraintsystem unmittelbar bilden und die Funktion **Sat** direkt von der Funktion **expert** aus aufrufen könnte. Aber es ist sinnvoll, wenn man die Expertenzeit messen will, sonst müßte man die Startzeit als zusätzlichen Parameter allen rekursiven **Sat**-Aufrufen übergeben. Der Aufruf (**Sat** $\mathcal{S}_0$ ML) liefert einen Booleanvektor, falls das Constraintsystem $\mathcal{S}_0$ widerspruchsfrei ist (d.h. $C \sqcap \neg D$ erfüllbar ist) und sonst **nil**, was das Scheitern des Versuches, ein Gegenbeispiel zu finden, bedeutet. **Sat** ruft sich rekursiv mit kleineren Eingabeparametern auf und wertet parallel die aktuelle Merkmalsmenge aus. Die Rekursion wird so lange fortgesetzt, bis entweder ein *Clash* entdeckt wird oder die Erfüllbarkeit entschieden ist, was die Ausgabe eines Booleanvektors zur Folge hat, der genauso lang ist wie die anfängliche Merkmalsliste. Dieser Vektor erfüllt bestimmte Eigenschaften, die für den weiteren Ablauf des Programms notwendig sind (siehe Algorithmus 5.4). **Sat** ruft mehrere Hilfsfunktionen auf. Zuerst wird durch den Aufruf der Funktion **clash** nach einem Widerspruch im aktuellen Constraintssystem gesucht. Wird kein *Clash* gefunden, so wird für den ersten gefundenen $\sqcap$-Ausdruck ein entsprechendes Constraintssystem gebildet und anschließend **Sat** aufgerufen. Wird kein $\sqcap$-Ausdruck gefunden, so wird dasselbe für $\sqcup$-Ausdrücke gemacht. Wenn kein *Clash* gefunden wurde und keine $\sqcap$- bzw. $\sqcup$-Ausdrücke mehr vorhanden sind, wird die Prozedur (**some-prozess** $\mathcal{S}$ ML) aufgerufen. Erst mit diesem Aufruf wird mit der Bearbeitung der Merkmalsliste begonnen. Die Funktion **some-prozess** faltet die Merkmalsliste um einen Schritt auf und wertet sie, so weit wie möglich, bzgl. des aktuellen Constraintsystems aus. Die Auswertung geschieht durch den Aufruf (**eval** $\mathcal{S}$ ML). Nach diesem Aufruf befinden sich in der Merkmalsliste keine Literale (Konzeptnamen oder ihre Negation). Die Merkmalsliste besteht dann aus Boole'schen Werten und Boole'schen Kombinationen von $\mathcal{ALC}$-Termen. Im weiteren nennen wir diese Liste *taml*. Die teilweise ausgewertete Merkmalsliste, *taml*, wird dann durch den Aufruf (**reform-ml** taml) für weitere Verarbeitungen umgeformt. Die umgeformte Liste besteht dann aus Paaren von der Form:

$$L_i = (\ [(\textbf{some}\ R_i\ C_{i1}) \ldots (\textbf{some}\ R_i\ C_{in_i})]\ [(\textbf{all}\ R_i\ D_{i1}) \ldots (\textbf{all}\ R_i\ D_{im_i})]\ )$$

Das i-te Element (L_i) der umgeformten Merkmalsliste ist mit dem eindeutigen Rollennamen R_i spezifiziert. Die erste Komponente jedes Paares besteht aus **some**-Ausdrücken mit dem gleichen Rollennamen und die zweite Komponente aus **all**-Ausdrücken mit demselben Rollennamen. Dabei kann eine Komponente fehlen. Anschließend wird die Prozedur (**Sat-Prozess** $\mathcal{S}$ uml) von **some-prozess** aufgerufen, wobei $\mathcal{S}$ das aktuelle Constraintsystem und uml die umgeformte Merkmalsliste sind. Diese Funktion führt für jedes Element L_i der umgeformten Liste folgende Schritte aus:

$\star$ Der Aufruf (**aml-R** L_i) liefert eine zwei-elementige Liste der Form:

$$\text{aml-}R_i \;=\; [n_i\ (C_{i1} \ldots C_{in_i}\ D_{i1} \ldots D_{im_i})]$$

Das zweite Element dieser Liste ist eine Liste von Teileigenschaften und das erste Element ist eine natürlicher Zahl n_i, die angibt, daß die ersten n_i Teilmerkmale aus **some**-Ausdrücke stammen. Diese Festlegung ist für die Auswertung der Merkmalsliste notwendig.

$\star$ Der Aufruf (**R-sat-prozess** $R_i\ \mathcal{S}$ aml-R_i) bestimmt zu R_i zuerst die Liste $\mathcal{D}_i \;=\; (D'_{i1} \ldots D'_{ik})$, wobei die Liste

$$[\ (\textbf{some}\ R_i\ C'_{i1}) \ldots (\textbf{some}\ R_i\ C'_{is})\ (\textbf{all}\ R_i\ D'_{i1}) \ldots (\textbf{all}\ R_i\ D'_{ik})\ ]$$

eine Liste aller **some-all**-Ausdrücke mit Rollennamen R_i aus dem aktuellen Constraintsystem $\mathcal{S}$ ist. Anschließend erfolgt für alle Constraintmengen

$$\mathcal{C}_{ij} \;=\; \{C'_{ij}\} \cup \mathcal{D}_i \text{ mit } \; 0 \leq j \leq s$$

der Aufruf (**Sat** $\mathcal{C}_{ij}$ aml-R_i). Sobald einer dieser Aufrufe **nil** liefert, wird der Wert **nil** über die Funktionen **R-sat-prozess, Sat-Prozess** und **some-prozess** an die **Sat**aufrufende Funktion zurückgegeben. Somit liefert die aufrufende Funktion **expert** den Wert **nil**. Es ist zu beachten, daß in diesem Falle die **Sat**-Aufrufe für nächste Constraintmengen nicht erfolgen, da die Unerfüllbarkeit der Subsumtion folgt. Ansonsten werden die Teilergebnisse aller Aufrufe (diese sind Booleanvektoren V_{ij} zu Teilmerkmalen aus aml-R_i und bzgl. der Constraintmengen $\mathcal{C}_{ij}$ mit $0 \leq j \leq k$) in einer Liste $L_{ik} \;=\; (V_{i1} \ldots V_{ik})$ gesammelt und zurückgegeben.

$\star$ Nun erfolgt der Aufruf (**werte-L-aus** $n_i\ L_{ik}$), mit der Steuerzahl n_i und der Vektorenliste L_{ik}. Diese Funktion verknüpft die ersten Teilvektoren der Länge n_i mit logischem **or** und die Restvektoren mit logischem **and**. Somit entsteht ein Booleanvektor, der genau so lang ist wie die Liste aml-R_i. Dieser Vektor enthält die Information über alle **some-all**-Ausdrücke mit Rollennamen R_i, die sich in der am Anfang teilweise ausgewerteten Merkmalsliste *taml* befinden. Der Prozess wird sukzessive für alle Rollennamen wiederholt. Am Ende hat man dann alle Ausdrücke aus *taml* ausgewertet. Nun ersetzt man in dieser Liste alle Ausdrücke durch ihre Booleanwerte und erhält dann eine Liste aus variablenfreien aussagenlogischen Formeln. Die Funktion **final-eval** nimmt diese Liste als Eingabe und liefert den endgültigen Booleanvektor zur ursprünglichen Merkmalsliste.

Die oben beschriebenen Prozeduren teilen ihre Aufgaben in kleinere Aufgaben, die durch mehrere Hilfsfunktionen erledigt werden. Diese Funktionen sind im

wesentlichen Sortier-, Such- und Ersetzungsfunktionen. Sie sind in den Dateien *extern.lisp, include.lisp, shift.lisp, pread.lisp, such.lisp, eval.lisp, expert.lisp und dg-base.lisp* beschrieben.

7.2 Statistische Auswertung

Nach der Implementation hat es sich herausgestellt, daß sich der Zeitverbrauch für Tboxen, die mehr als 20 definierte Konzeptnamen enthalten, in einem unvertretbaren Rahmen bewegt. Wie schon erwähnt wurde, ist dafür die enorme Anzahl von trivialen Implikationen verantwortlich. Um bestimmten empirischen Annahmen und Testzwecken nachgehen zu können, wurden eine Reihe von Testfunktionen geschrieben, durch die insbesondere Zeitmessungen vorgenommen werden können. Diese Funktionen sind in der Datei *experiment.lisp* abgespeichert. Es wurden zufällige Tboxen und Kontexte mit verschiedenen Eigenschaften wie Größe, Abhängigkeit zwischen Merkmalen usw. erzeugt und untersucht.

7.2.1 Experimente mit zufällig erzeugten Tboxen

Die Funktion **creat-wb** erzeugt eine zufällige Wissensbasis. Als Eingabe erhält sie verschiedene Steuerparameter, durch die Tboxen verschiedener Länge, maximaler Anzahl von primitiven Konzepten bzw. Rollennamen, Abhängigkeitsgrad, maximaler Schachtelungstiefe von Konzeptdefinitionen und von Konstruktoren bei einzelnen Ausdrücken, Wahrscheinlichkeit für die Erzeugung bestimmter Ausdrücke (z.B. ist die Anzahl von **und**-Ausdrücken in realistischen Tboxen größer als die anderen Ausdrücken) usw. vorgegeben werden können. Um die Erzeugung von zyklischen Tboxen zu verhindern, wurde bei Erzeugung des $(i+1)$-ten Konzeptnamens A_{i+1} nur die Verwendung von vorher definierten Namen $\{A_1, A_2, \ldots, A_i\}$ zugelassen.

Die wichtigsten Funktionen, wie **expert, gilt, dgb, hull-operator** und **next-closed**, wurden so modifiziert, daß man die durch sie verbrauchte Zeit messen konnte. Anschließend wurde mit Tboxen verschiedener Größe experimentiert. Dabei wurde das Verhalten des Programmes bzgl. verschiedener Gesichtspunkte untersucht, indem verschiedene Steuerparameter eingegeben wurden. Die Mittelwerte zu diesen Experimenten sind in der Tabelle 7.1 eingetragen. Die Einträge in dieser Tabelle sind Mittelwerte die durch ≥ 50 Versuche für die jeweilige Tbox-Größe ermittelt worden sind.

Da der Zeitaufwand ab der Tbox-Größe ≥ 30 kritisch wurde, mußte eine Zeitschranke eingebaut werden, durch die mindestens Teilergebnisse zu ermitteln waren. Die Abkürzungen *ZS, Erf, Abg* stehen für die Zeitschranke (in sec), die Anzahl der erfolgreichen Versuche und Anzahl der Versuche, bei denen die Zeit-

Tabelle 7.1:

Mittelwerte für zufällig erzeugte Tboxen								
		# Expert-calls	*Expert-zeit (sec)*	*Dgb-zeit (sec)*	*# der Basis*	*# Gegen-bsp.*	*# trivial-impl.*	*Zeit funkt. Alg.*
N|Anz|ZS	Erf.	= -Werte						
	Abg.	$\geq$ -Werte						
10|150|50	150	74.55	1.7	0.15	4.74	69.80	197.10	0.74
	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
15|100|300	98	472.29	26.18	0.15	7.59	464.70	2772.59	10.9
	2	227.0	317.99	1.56	9.5	217.5	1033.1	142.67
20|100|300	70	854.94	57.03	46.11	13.35	841.58	13491.14	25.91
	30	1536.7	248.11	64.75	10.63	1526.06	15429.63	92.81
30|100|300	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
	100	1653.02	169.71	135.05	10.42	1642.62	18295.83	73.68
40|100|300	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
	100	2020.90	218.81	183.34	8.96	2011.97	15267.73	84.15
50|50|600	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
	50	1939.86	325.38	278.80	9.3	1930.86	16212.23	120.5
60|50|1000	0	-.-	-.-	-.-	-.-	.-	-.-	-.-
	50	2553.06	599.92	413.28	9.33	2543.73	14523.72	230.73
70|50|1500	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
	50	3362.21	627.28	873.36	9.29	3352.92	21018.32	241.43
80|50|2200	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
	50	3321.8	1184.66	1040.15	9.61	3312.19	16974.14	438.76
90|50|3000	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
	50	4115.54	1728.22	1277.81	9.94	4105.6	17373.58	617.22
100|50|3600	0	-.-	-.-	-.-	-.-	-.-	-.-	-.-
	50	4251.5	1818.02	1804.01	11.87	4239.6	18582.06	673.34

N = Tbox-Größe, Anz = Anzahl der Experimente, ZS = Zeitschranke (Sec).

schranke überschritten war und deswegen das Verfahren mit Ausgabe bisdahin errechneter Werte abgebrochen wurde. Bei *Abg.* sind die Mittelwerte also als $\geq$-Werte zu interpretieren. Man sieht, daß ab der Tbox-Größe 30 alle Versuche erfolglos sind.

In der Spalte mit der Überschrift *# der Basis* sind die Größe der bis dahin berechneten Basis, also die Anzahl der bis dahin erzeugten, gültigen und nichttrivialen Implikationen angegeben. Man sieht, daß diese Größe im Vergleich mit der Anzahl der erzeugten Gegenbeispiele und insbesondere der Anzahl der Begriffsinhalte sehr klein ist. Das scheint eine Besonderheit unserer speziellen Kontexte (siehe Definition 5.1) zu sein, da die errechneten Teilbasen bei zufällig erzeugten Kontexten wesentlich größer sind (siehe Tabelle 7.4).

Man kann zwar auf Anhieb die unvergleichbar größere Anzahl von trivialen Implikationen sehen, aber man kann ihre zeitraubende Eigenschaft erst feststellen, wenn man die Zeitwerte vergleicht. In der Spalte mit der Überschrift *Expertenzeit* sind die Mittelwerte für die Zeit eingetragen, die allein der Experte verbraucht hat. Dem gegenüber sind in der Spalte mit der Überschrift *dgb-zeit* die Mittelwerte für die Zeit eingetragen, die der ,,Gantersche" Teil des Verfahrens, d.h. Gesamtlaufzeit ohne Expertenzeit, verbraucht hat. Ein kleiner Anteil dieser Zeit ist natürlich sinnvoll verbraucht, nämlich die Zeit, die durch Durchläufe verbraucht wurde, bei denen entweder ein Gegenbeispiel erzeugt oder eine Implikation zu der Basis hinzugefügt worden ist. Die durch triviale Implikationen vergeudete Zeit kann man also nach der Formel:

$$\boxed{\frac{dgbzeit * (1 - \frac{\#Basis\ +\ \#obj}{\#Basis\ +\ \#obj\ +\ \#trivimpls})}{dgbzeit + expertenzeit} * 100} \tag{7.1}$$

prozentual berechnen. Nach dieser Formel liegt die vergeudete Zeit bei mindestens **41**%. Dieser Prozentsatz ist ganz unabhängig davon, wieviel Zeit der Experte verbraucht, d.h. auch wenn die Expertenzeit durch z.B. eine effizientere Implementierung oder Erzeugung von guten Gegenbeispielen optimiert wird, muß man damit rechnen, daß mindestens 41% der Gesamtlaufzeit für etwas verbraucht wird, was nicht direkt zur Erzeugung der gesuchten Basis beiträgt. Demzufolge verbraucht der ,,Gantersche" Anteil des Verfahrens fast so viel Zeit wie der Experte.

Tboxen mit verschiedenem Abhängigkeitsgrad

Es liegt nahe zu erwarten, daß bei Kontexten, in denen es eine stärkere Abhängigkeit zwischen Merkmalen gibt, weniger Begriffsinhalte existieren. Im Zusammenhang mit Tboxen bedeutet dies, Tboxen mit einer größeren Anzahl von definierten Namen auf den rechten Seiten der Axiome zu betrachten. Für den Abhängigkeitsgrad wurde eine Skala von 1 bis 10 definiert, wobei die Wahl der Zufallszahl

Tabelle 7.2:

Mittelwerte für zufällig erzeugte Tboxen mit verschiedenen Abhängigkeitsgrad						
Abh.-keits-grad	*# Exp.-Aufruf*	*Exp.-zeit (sec)*	*Dgb-zeit (sec)*	*# der Basis*	*# Gegen-bsp.*	*# trivial-impl.*
1	161.13	1.27	0.29	2.55	159.53	442.83
3	116.28	1.45	0.23	3.45	112.76	329.78
5	87.9	3.1	0.18	3.94	83.95	246.48
7	69.4	2.1	0.13	4.95	64.37	172.9
9	46.9	3.2	0.09	6.02	40.8	115.18
10	39.4	3.4	0.07	6.01	33.4	78.12

Tbox-Größe = 10, Anzahl der Experimente jeweils 150.

$n \in \{1, \ldots, 10\}$ bedeutet, daß bei der Erzeugung eines Ausdrucks mit der Wahrscheinlichkeit $(9n)\%$ definierte Konzeptnamen statt primitive als Grundbaustein genommen werden. Es wurden Experimente mit verschiedenen Abhängigkeitsgraden durchgeführt. Die Mittelwerte hierzu sind in der Tabelle 7.2 eingetragen.

Wie die Werte aus dieser Tabelle zeigen, hat sich die Vermutung, daß ein höherer Abhängigkeitsgrad eine kleinere Anzahl von trivialen Implikationen zur Folge hat, in der Tat bestätigt. Man kann sich diese Tatsache vielleicht folgenderweise erklären:
Die höhere Abhängigkeit zwischen den definierten Namen bedeutet, daß im Kontext mehr Implikationen gelten. Man kann sich überlegen, daß in einer Tbox, in der kein Name einen anderen Namen direkt oder indirekt verwendet, viel weniger Implikation gelten. Diese Tatsache ist an der kleineren Anzahl der erzeugten Gegenbeispiele bei Tboxen mit größerem Abhängigkeitsgrad erkennbar. Mehr gültige Implikationen haben entsprechend größere Basen zur Folge. Wenn man sich nun den Hüllenoperator $\mathcal{L}^*(X)$ betrachtet, der zu der gegebenen Menge X den lektisch nächsten Inhalt liefert, merkt man, daß je größer die Basis $\mathcal{L}$ ist, desto wahrscheinlicher ist es, daß ein größerer Inhalt $B = \mathcal{L}^*(X)$ berechnet wird. Wenn nun B größer ist, unterscheidet B sich immer weniger von B'' und somit erhöht sich die Wahrscheinlichkeit, daß $B = B''$ ist. Man kann vielleicht diese empirische Vermutung durch eine geeignete wahrscheinlichkeitstheoretische Modellierung bestätigen. Wenn man aber die Nebeneffekte dieser Maßnahme in Betracht zieht, sieht man ein, daß sie insgesamt nicht als eine Optimierungsmaßnahme bewertet werden kann. Einerseits gibt es weniger triviale Implikationen und weniger Gegenbeispiele (da mehr Implikationen im Kontext gelten). Andererseits nimmt die Anzahl der trivialen Implikationen nur linear ab, was im Vergleich

zur exponentiellen Größe dieser Implikationen nicht viel bringt und auch weniger Gegenbeispiele bedeuten nicht weniger Expertenaufrufe, da es genau so viele oder mehr gültige Implikationen gibt, für die auch der teure Experte aufgerufen werden muß. Es gibt noch den großen Nachteil, daß sich die Expertenzeit für einzelne Aufrufe erhöht. Außerdem kann man in der realen Welt den Abhängigkeitsgrad von Tboxen nicht beliebig setzen, da Tboxen entsprechend der Anwendung definiert werden müssen.

In weiteren Versuchen wurde immer der Abhängigkeitsgrad 5 gewählt. Diese Wahl ist unter der Annahme getroffen, daß diese Größe den realistischen Tboxen entspricht.

Mehraufwand des Experten

Es wurde der Frage nachgegangen, wie teuer ein Expertenaufruf im Vergleich zu einem normalen Subsumtionsaufruf mittels des funktionalen Algorithmus ist. Die Funktion **Sat** wurde so modifiziert, daß bei der Eingabe der leeren Merkmalsliste der einfache funktionale Algorithmus ausgeführt wird. Es wurden während der Merkmalexploration zu jeder erzeugten Tbox alle erzeugten Implikationen in einer lokalen Liste zwischengespeichert. Anschließend wurde der funktionale Algorithmus für sie aufgerufen. Dabei wurde die Zeit gemessen. Mittelwerte hierzu sind in der letzten Spalte der Tabelle 7.1 eingetragen.
Man sieht, daß die Erweiterung des funktionalen Algorithmus zum Experten den Zeitaufwand durchschnittlich um den konstanten Faktor **2.53** erhöht hat.

Klassifikation der expandierten Tboxen

Wir haben erwähnt, daß die Berechnung einer erweiterten Subsumtionshierarchie ein Spezialfall der Klassifikation ist. Man kann nämlich für jede Konjunktion zwischen definierten Namen einen neuen Namen definieren und anschließend die auf diese Weise expandierte Tbox klassifizieren. Die zu einer Tbox mit n definierten Namen expandierte Tbox hat genau $2^n - 1$ Konzeptnamen. Die expandierte Tbox ist zwar exponentiell groß, aber es gibt KR-Systeme, die in Folge einer mehrjährigen Überarbeitung und Optimierung sehr effektiv die Klassifikation auch für größere Tboxen durchführen können. Es ist wünschenswert, die Anzahl der Subsumtionsaufrufe, die ein solches System für expandierte Tboxen durchführen muß, zu berechnen, um diese Anzahl mit der entsprechenden Anzahl bei der Berechnung der erweiterten Subsumtionshierarchie mittels Merkmalexploration zu vergleichen.
Eines dieser optimierten Systemen heißt **CRACK**[1]. Es wurden mehrere zufällige Tboxen in CRACK-Syntax erzeugt. Anschließend wurden sie klassifiziert. Dabei

[1] CRACK 1.0, beta Version, 1995,1996,1997 Enrico Franconi.

Tabelle 7.3:

Experimente mit CRACK-System			
Größe der Tbox	*# der Exper.*	*# Aufr. Merkmal-expl.*	*# Aufr. CRACK Sys.*
3	100	5.21	13.74
5	100	13.60	139.71
8	100	33.86	2908.37
10	50	74.55	19701.26
12	50	147.18	33546.43

wurde die Anzahl der echten[2] Subsumtionsaufrufe für jeweilige Tbox berechnet. Leider mußte man sich mit Tbox-Größen ≤ 12 begnügen, da expandierte Tboxen sehr groß werden[3]. Die Mittelwerte hierzu sind in der Tabelle 7.3 eingetragen. Wenn man von einem *Over-Head* von 300% für einen Expertenaufruf im Vergleich mit einem Aufruf des funktionalen Algorithmus ausgeht, entspricht ein Expertenaufruf 3 Aufrufen des funktionalen Algorithmus. Aus den Werten in der Tabelle 7.3 ist dann zu entnehmen, daß man die Anzahl der teuren Subsumtionsaufrufe, für Tbox-Größe $N \geq 8$ um fast **99**% minimieren kann, wenn man die erweiterte Subsumtionshierarchie durch die Merkmalexploration berechnet. Für die Größe $5 \leq N < 8$ erzielt man eine Minimierung von **90**% und für Größe $N < 5$ eine Minimierung von **30**%.

7.2.2 Experimente mit zufällig erzeugten Kontexten

Es stellt sich die Frage, ob das Problem der trivialen Implikationen ein spezifisches Problem für unsere speziellen Kontexte (siehe Definition 5.1) ist. Um dieser Frage nachzugehen, wurden Experimente mit zufällig erzeugten Kontexten durchgeführt. Die Funktion **creat-kontext** erzeugt zufällige Kontexte mit vorgegebenen Größen für die Objekt- sowie Merkmalsmenge, wobei die Erzeugung von wiederholten Gegenständen verhindert wird. Diese Kontexte wurden während der Merkmalexploration als Experte eingesetzt (im folgenden nennen wir sie *Kontext-experte*). Der Experte erhält die Implikation $A \rightarrow B$ und entscheidet ihre Gültigkeit, indem er alle Zeilen des Kontextes durchläuft und überprüft, ob die Implikation von allen Zeileninhalten erfüllt ist. Wenn ein Objekt gefunden wird, dessen Inhalt die Implikation nicht erfüllt, ist es ein Gegenbeispiel. Es wurden für Merkmalsmengen verschiedener Größe mehrere zufällige Kontexte er-

[2]Hier sind die Subsumtionen gemeint, für die der Subsumtionsalgorithmus aufgerufen wird. Z.B. muß der Algorithmus für die triviale Subsumtion $A \sqsubseteq \top$ nicht aufgerufen werden.

[3]Für die Große $N = 15$ bzw. $N = 20$ hat man fast 32000 bzw. 1000000 definierte Namen.

zeugt und anschließend wurde eine Merkmalexploration durchgeführt.

Außerdem wollte man gerne wissen, inwiefern die Wahl der guten Gegenbeispiele die Anzahl der Expertenaufrufe reduziert. Deshalb wurde bei diesen Experimenten bzgl. jedes Kontextes einmal das beste vorhandene Gegenbeispiel gewählt und einmal das erste gefundene Gegenbeispiel. Die Mittelwerte zu diesen Experimenten sind in der Tabelle 7.4 eingetragen.
Die Abkürzungen ,,N" bzw. ,,Anz" stehen für die Anzahl der Merkmale bzw. Anzahl der Experimente bzgl. einer festen Größe der Merkmalsmenge. Die Abkürzung *#Kon.-expert* steht für die Größe des Kontextes, die als Experte eingesetzt wurde. Unter *dgb-zeit* soll man die Gesamtlaufzeit ohne Expertenzeit verstehen. Das entspricht dem Zeitverbrauch des ,,Ganterschen" Teils des Algorithmus 4.45. Bei diesen Experimenten wurden für Kontexte, die als Experte eingesetzt sind, dieselben Größen gewählt, die der Objektanzahl in der Tabelle 7.1 entsprechen. D.h. die Größe von Kontexten, die während der Merkmalexploration mit Tboxen erzeugt wurden. Entsprechend wurden dieselben Zeitschranken (*ZS*) und der Abhängigkeitsgrad 5 gewählt. Aus der Tabelle 7.4 kann man einige Schlüsse ziehen:

⋆ Daß bei der Merkmalexploration sehr viele Begriffsinhalte erzeugt werden, was sehr viele triviale Implikationen zur Folge hat, ist keine Besonderheit unserer speziellen Kontexte. Wäre dies der Fall, dann müßten die Ergebnisse der Experimente mit zufällig erzeugten Kontexten anderes aussehen, als die bei den zufällig erzeugten Tboxen. Wenn man die Mittelwerte aus der Tabelle 7.4 mit den Werten aus der Tabelle 7.1 vergleicht, kann man als einzige Besonderheit unserer Spezialkontexte eine kleinere Anzahl von in diesen Kontexten gültigen Implikationen feststellen. Das hat aber keine wesentliche Auswirkung auf die Zeit, denn es werden zwar weniger Gegenbeispiele erzeugt, dafür aber genau so viel oder noch mehr gültige Implikationen, für die auch der Experte aufgerufen werden muß. Alle Versuche ab $N \geq 30$ sind wieder gescheitert, weil sehr viele Begriffsinhalte berechnet werden mußten.

⋆ Die Anzahl der berechneten Begriffsinhalte ist in den meisten Fällen größer. Ein Grund kann sein, daß der Experte hier innerhalb derselben Zeit weniger Zeit verbraucht hat (er ruft ja keinen zeitaufwendigen Subsumtionsalgorithmus auf), als der echte Experte bei zufällig erzeugten Tboxen. Dies trifft inbesondere für die größeren Tboxen zu.

⋆ Dadurch, daß der Experte hier logischerweise wesentlich weniger Zeit verbraucht als der echte Experte, erhöht sich die durch die trivialen Implikationen vergeudete Zeit. Dieser Prozentsatz, berechnet nach der Formel 7.1 auf Seite 86, liegt bzgl. zufälliger Kontexte bei **86**%. Dies zeigt, daß auch eine wesentliche Optimierung des Experten das Problem nicht lösen kann.

Tabelle 7.4:

Zufällig erzeugte Kontexte als Experte							
	# kon.-expert	*# Expert-calls*	*Expert-zeit (sec)*	*Dgb-zeit (sec)*	*# der Basis*	*# Gegen-bsp.*	*# trivial-impl.*
N|Anz|ZS	mit erstem gefundenen Gegenbeispiel						
	mit bestem gefundenen gegenbeispielen						
10|1000|300	100	121.1	0.03	0.06	60.8	60.2	473.57
	100	110.5	0.04	0.61	60.8	49.64	473.57
15|100|300	200	1098.78	0.09	266.67	950.47	148.3	5057.68
	200	1094.63	1.01	266.47	950.48	144.15	5057.68
20|100|300	1500	1739.02	7.58	292.54	1309.41	428.94	16890.69
	1500	1707.03	8.01	292.14	1309.41	397.68	16890.69
30|100|300	1700	1652.95	13.49	286.61	1142.15	510.69	17753.92
	1700	1612.26	14.54	285.55	1141.15	471.00	17749.92
40|100|300	2000	1501.87	16.17	283.89	974.12	527.66	15750.38
	2000	1463.47	17.84	282.25	975.20	488.2	15765.72
50|100|600	2000	2029.69	25.81	574.31	1430.57	598.97	19047.05
	2000	2000.21	28.38	571.75	1433.74	566.36	19077.73
60|50|1000	2500	2363.12	41.79	958.40	1660.92	702.12	22730.66
	2500	2329.44	46.25	953.88	1660.84	668.44	22733.38
70|50|1500	3500	2509.38	65.74	1434.40	1709.71	799.65	26374.22
	3500	2472.26	73.42	1426.72	1710.18	761.97	26381.34
80|50|1500	3500	2632.84	75.36	2124.85	1827.37	805.44	29823.42
	3500	2061.06	101.42	2115.59	1829.75	771.13	29850.28
90|40|3000	4000	2868.62	101.42	2898.94	1976.47	895.05	31703.52
	4000	2831.87	114.21	2886.09	1975.77	856.03	31713.12
100|40|3600	4000	2986.97	112.90	3487.45	2076.35	910.57	32107.47
	4000	2951.0	126.98	3473.25	2077.55	873.40	32125.20

N = Merkmalsanzahl, Anz = Anzahl der Experimente, ZS = Zeitschranke (Sec).

Ab $N \geq 30$ alle Versuche erfolglos.

⋆ Man sieht, daß die Wahl von guten Gegenbeispielen in der Tat die Anzahl der Expertenaufrufe verringert hat. Diese Reduzierung liegt bei weniger als **3**%, was keine wirkliche Verbesserung bedeutet. Aber man muß beachten, daß diese guten Beispiele nicht die besten Beispiele sind, denn als ein gutes Beispiel wurde unter allen im Kontext vorhandenen Beispielen eines gewählt, das am meistens True-Werte besaß. Es kann sein, daß ein solches Gegenbeispiel noch mehr Merkmale besitzen und dadurch besser werden kann. Man darf vielleicht hoffen, daß bei Merkmalexploration für Tboxen mehr erreicht wird, wenn man die besten möglichen (oder annähernd guten) Gegenbeispiele erzwingt.

⋆ Bei der Betrachtung von einzelnen Experimenten hat sich, wie erwartet, herausgestellt, daß die Anzahl der Begriffsinhalte jeweils bei guten und schlechten (oder weniger guten) Gegenbeispielen nicht nur bzgl. eingetragener Mittelwerte, sondern auch in allen Einzelfällen genau gleich ist. D.h. die Erzeugung von trivialen Implikationen läßt sich von der Art der erzeugten Objekten nicht beeinflussen, denn alle Begriffsinhalte müssen während der Merkmalexploration erzeugt werden.

Kapitel 8

Zusammenfassung und Ausblick

In der Softwareentwicklung läßt sich seit einiger Zeit die Tendenz beobachten, die konventionelle Datenverarbeitung durch die Wissensverarbeitung zu ergänzen oder gar zu ersetzen. So werden zum Beispiel wissensbasierte bzw. Expertensysteme entwickelt. Konstituierende Teile eines Expertensystems sind die **Inferenzkomponente** und **Wissensbasis**. Die Grundlage für diese Komponenten bildet ein geeigneter Formalismus zur **Wissensrepräsentation**. Wissensbasierte Systeme verwenden dazu neue Ansätze in Form von logikorientierten, regelbasierten und objektorientierten Methoden mit prädikatenlogischen Formeln, Produktionsregeln, Frames und semantischen Netzen. Der Mangel an einer richtigen Semantik anderer Formalismen hat insbesondere die Rolle der logikorientierten Wissensdarstellung hervorgehoben.

In diesem Zusammenhang stellen die **terminologischen** KR-Systeme wegen ihrer klaren Syntax und Semantik und ihrer Fähigkeit, das Wissen strukturiert darzustellen, eine wichtige Klasse dar. In terminologischen KR-Systemen werden die relevanten Merkmale eines Wissensgebietes als **Konzeptnamen** in sogenannten terminologischen Boxen (Tbox) definiert. Objekte, die diese Eigenschaften besitzen, werden als Elemente eines Interpretationsbereiches, sogenannte **Individuen**, dargestellt. Die Objekt-Eigenschaft-Beziehungen werden durch **Rollennamen** ausgedrückt. Kombinierte Eigenschaften werden durch **Konzeptterme** dargestellt, die durch Anwendung bestimmter Operatoren auf einfache Konzeptnamen und Rollennamen entstehen. In dieser Arbeit wurde die terminologische Sprache $\mathcal{ALC}$ zugrundegelegt.

Als Schlußfolgerungsproblem wird in terminologischen KR-Systemen häufig die **Subsumtion** betrachtet, die sich für $\mathcal{ALC}$ auf Erfüllbarkeit von Konzepttermen reduzieren läßt. Probleme wie Erfüllbarkeit und ihre Komplexität sind Gegenstand neuester Forschungen in diesem Gebiet. Die Ergebnisse dieser Forschungen haben gezeigt, daß das Erfüllbarkeitsproblem bei terminologischen KR-Systemen ein hartes Problem ist. Dieses motiviert den Versuch, die entwickelten Algorith-

men so weit wie möglich zu optimieren, um praktisch anwendbare Methoden zu gewinnen. Aus diesen Bemühungen haben sich viele optimierte Algorithmen ergeben, die meist in den heutigen KR-Systemen als ein Systemdienst implementiert sind. Einer dieser Systemdienste ist die **Klassifikation** von Tboxen, welche eine **Subsumtionshierarchie** zwischen den in einer Tbox definierten Konzeptnamen berechnet. Die Klassifikation vereinfacht das Schlußfolgerungsproblem während der Laufzeit. Die Informationen, die aus dieser einfachen Hierarchie gewonnen werden können, sind aber sehr beschränkt, da diese Hierarchie keine Auskunft über die Subsumtionsrelationen zwischen den Konjunktionen von Konzeptnamen gibt. Solche Konjunktionen stellen einen großen Teil der zusammengesetzten Eigenschaften dar. Es ist wünschenswert, eine **erweiterte Subsumtionshierarchie** zu berechnen, die alle Konjunktionen zwischen den definierten Konzeptnamen repräsentiert. Zur Berechnung einer solchen Hierarchie möchte man natürlich so wenig wie möglich den teuren Subsumtionsalgorithmus aufrufen müssen.

Das Ziel dieser Arbeit war, diese Aufgabe zu lösen. Hierzu wurden die Resultate aus der Welt der **Formalen Begriffsanalyse** herangezogen. In diesem Gebiet werden Objekte, Eigenschaften (Merkmale) und die Objekt-Eigenschafts-Beziehung durch **formale Kontexte** und **formale Begriffe** dargestellt. Begriffsanalytische Methoden, die von Ganter und Wille entwickelt wurden, ermöglichen die Berechnung der Begriffshierarchie eines gegebenen Kontextes, die als ein vollständiger Verband dargestellt werden kann.

Durch die Definition eines geeigneten Kontextes zu einer gegebenen Tbox wurde der Zusammenhang zwischen der Begriffsanalyse und terminologischen Tboxen hergestellt, in der Hoffnung, daß dadurch die Berechnung einer erweiterten Subsumtionshierarchie im oben definierten Sinne ermöglicht wird. In diesem Kontext entsprechen die Merkmale den in der Tbox definierten Konzeptnamen. Entsprechend ist jeder Begriffsinhalt eine Konjunktion zwischen Konzeptnamen.

Ganter und Wille haben eine Methode angegeben (**Merkmalexploration**), durch die man eine minimale Implikationsbasis für einen gegebenen Kontext berechnen kann, auch wenn der Kontext unendlich ist. Diese Basis repräsentiert dann den ganzen Begriffsverband des Kontextes. Die durch die Merkmalexploration berechnete Basis $\mathcal{L}$ besteht aus einer minimalen Anzahl von Implikationen, die im Kontext gelten. Die Menge aller im Kontext gültigen Implikationen ist dann genau die Menge aller Implikationen, die aus $\mathcal{L}$ folgen. Man berechnet diese Basis für einen gegebenen Kontext, indem man alle sogenannten **Pseudoinhalte** des Kontextes mit Hilfe einer auf der Potenzmenge der Menge aller Merkmale definierten Ordnung (**lektische Ordnung**) berechnet.

In dem zu einer Tbox geeignet definierten Kontext entspricht diese Basis einer minimalen Repräsentation aller Subsumtionen zwischen den Konjunktionen der

definierten Konzeptnamen. Um diese Basis zu berechnen, hat der ,,Gantersche“ Algorithmus einen menschlichen Experten vorausgesetzt, der sich in dem Anwendungsgebiet auskennt und zu jeder Implikation zwischen Merkmalen entscheiden kann, ob sie im Kontext gilt oder nicht. Im Zusammenhang mit Tboxen wird diese Rolle von einem Subsumtionsalgorithmus übernommen. Hierzu wurde ein bekannter, optimierter und regelbasierter Erfüllbarkeitsalgorithmus (**funktionaler Algorithmus**) zu dem gewünschten Experten erweitert, der eingesetzt im ,,Ganterschen“ Algorithmus die Berechnung der gewünschten Implikationsbasis ermöglicht.

Während der Merkmalexploration werden außerdem **redundante** Objekte erzeugt. Sie sind diejenigen vom Experten erzeugten Gegenbeispiele, die nichts zur Erzeugung der Basis beitragen. Das heißt, die Entfernung solcher Objekte aus dem Kontext beeinflußt die Struktur des Begriffsverbandes nicht. Man möchte die Erzeugung solcher Objekte vermeiden, denn für sie wird der teure Experte unnötig aufgerufen. In der Arbeit wurden diese Objekte mathematisch charakterisiert. Man kann während der Merkmalexploration die Erzeugung solcher Objekte vehindern. Das erfordet aber einen Mehraufwand, der möglicherweise durch die Entwicklung und die Implementierung einer geeigneten Heuristik reduziert werden kann.

Die oben erwähnten Algorithmen und Methoden wurden in der Arbeit ausführlich diskutiert. Um die Güte dieser Methoden zu beurteilen und ihre Anwendbarkeit zu testen, wurden sie implementiert. Anschließend wurde das implementierte Programm für verschiedene Eingabedaten getestet und statistisch ausgewertet.

Viele Experimente, die mit zufällig erzeugten Tboxen und Kontexten ausgeführt wurden, haben einerseits gezeigt, daß man für die Berechnung dieser Basis und somit der erweiterten Subsumtionshierarchie bis auf 99% weniger oft den teuren Subsumtionsalgorithmus aufrufen muß, als die Anzahl der Aufrufe bei der Berechnung der erweiterten Subsumtionshierarchie mittels der Klassifikation der expandierten Tbox.
Andererseits wurde festgestellt, daß für Tboxen, die mehr als 20 definierte Konzeptnamen haben, zu viele triviale Implikationen erzeugt werden, die die Zeit für den Aufbau der gesuchten Basis drastisch verzögern. Der Grund liegt in den exponentiell vielen Begriffsinhalten, die während der Merkmalexploration erzeugt werden müssen.

Die ,,Gantersche“ Methode, in der angegebenen Form, kann in der Praxis für die Berechnung einer erweiterten Subsumtionshierarchie nicht eingesetzt werden, da sich die Zeitkomplexität in einem unvertretbaren Rahmen bewegt. Wenn man die begriffsanalytischen Methoden für diesen Zweck einsetzen will, muß man versuchen, die Menge aller Pseudoinhalte eines Kontextes nicht über die Berechnung

aller Begriffsinhalte zu erzeugen, sondern sie direkt zu berechnen. Man könnte z.B. versuchen, es zu untersuchen, ob die Menge aller Pseudoinhalte eines Kontextes ein Hüllensystem ist und, ob die Hüllen (d.h. Pseudoinhalte) dieses Systems durch den zugehörigen Hüllenoperator mit einem vertretbaren Aufwand zu berechnen sind.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meiner Diplomarbeit unterstützt haben.

Insbesondere möchte ich mich bei Herrn Prof. Baader für die hilfreichen Diskussionen und die hervorragende Betreuung bedanken.

Ich danke außerdem Jörn Richts für seine Hilfe bei programmiertechnischen Problemen, Can Adam Albayrak für seine Hilfe bei TEXnischen Problemen und Ulrike Sattler für ihre Hilfe bei Experimenten mit dem CRACK-System. Jörg Dölfer und Felix Wolf danke ich für ihre Hilfe bei der Suche geeigneter Formulierungen und beim Korrekturlesen meiner Arbeit. Mein besonderer Dank gilt Ralf Molitor, für die anregende Diskussionen, die wir während der gemeinsamen Diplomandenzeit führen konnten.

Zum Schluß danke ich allen Freunden, die mich auch in den anstrengendsten Phasen meines Studiums unterstützt und ermutigt haben.

Literaturverzeichnis

[1] Struß, P.: Gibt es Expertensysteme?, in *Computer Magazin Data*, 5.1986, S. 294 ff.

[2] Habel, C. (ed.): Künstliche Intelligence, Frühjahrsschule Dassel; Inform. Fachb. 93; Springer, Berlin, Heidelberg, New York, Tokyo, 1985.

[3] Laubsch, J.: Techniken der Wissensdarstellung; in [2] S. 48 ff.

[4] Hayes, P.J.: The logic of Frames; in: Mething, D. (ed.): Frame Conception and Text Understanding; de Gruyter, Berlin 1980, S. 46.

[5] Minsky, M.: A Framework for Representation Knowledge: in: Winston (ed.): The Psychology of Computer Vision; McGraw Hill, New York, 1975.

[6] Richter, L. und Stucky, W. (eds.): Artifical Intelligence - Eine Einführung: Teuber, Stuttgart, 1978

[7] Trost, H.: Wissensrepräsentation in der AI am Beispiel Semantischer Netze; in [6], S. 48 ff.

[8] Shortlife, E.H., Davis, R., Axline, S.G., Buchanan, B.G., Cordell Green, C., Cohen, S.N. : Computer-Based Consultation in Clinical Therapeutics: Explanation and Rule Acquisition Capabilities of MYCIN System; *Comp. Biomed. Res., 8, Pages* 303-320, 1975

[9] F. Baader. Computing a Minimal Representation of the Subsumption Lattice of all Conjunctions of Concepts Defined In a Terminologie. In *Proceedings of the International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency, KRUSE 95,* Santa Cruz, USA, 1995.

[10] B. Nebel: Reasoning and Revision in Hybrid Representation Systems. In: *Lecture Notes in Computer Science 442.* Springer Verlag, Berlin, 1990.

[11] K. Schild. A correspondence theory for teminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Pages 466-471, Sydney, Australia, 1991.

[12] A. Borgida. On the relationship between description logic and predicate logic queries. Resarch Report LCSR-TR-295-A, Department of Computer Science, Rutgers University, Piscataway, NJ, USA, 1993.

[13] M. Mortimer. On languages with two variables, *Zeitschr. f. math. Logik und Grundlagen d. math.,* 21:135-140, 1975.

[14] M. Schmidt-Schauß und G. Smolka, Attributive concept descriptions with complements, *Artificial intelligens* **48** (1)(1991) 1-26.

[15] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languagees. In J.A. Allen, R. Frikes, and E. Sandewall, editors *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference, Page* 151-162, Cambridge, MA, 1991. Morgan Kaufmann.

[16] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235-249,1990.

[17] M.R. Garey and D.S. Johnson, A Guide to the Theory of NP-Completeness, *Computers and Intractability*, Freeman, San Francisco, CA, 1997

[18] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In M. Richter and H. Boley, editors, *International Workshop on Processing Declarative Knowledge*, volume 567 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1991.

[19] Franz Baader and Hans-Jürgen Bürkert and Bernhard Hollunder and Werner Nutt and Jörg H.Siekmann. Concept Logics. *Computational Logics, Symposium Proceedings*, 1990, Springer Verlag.

[20] Schmidt-Schauß. Subsumption in *KL-ONE* is Undecidable, *KR-89,* Pages 421-431, 1989, Boston (USA).

[21] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological systems. *J. Applied Intelligence,* 4:109-132, 1994.

[22] T. Lipkis. A KL-ONE classifier. In J.G. Schmolze and R.J. Brachman, editors, *Proceedings of the 1981 KL-ONE Workshop*, page 128-145, Cambridge, MA, 1982. The proceedings have been published as BBN Report No. 4842 and Fairchild Technical Report No. 618.

[23] R. MacGregor. A deductive pattern matcher. In *Proceedings of the 7th National Conference of th American Association for Artificial Intelligence, pages 403-408*, Saint Paul, MI, Aug. 1988.

[24] C. Peterson, A. Schmidel, C. Kindermann, and J. Quantz. The BACK system revisited. KIT Report 75, Department of Computer Science, Technische Universität Berlin, Berlin, Germany, Sept. 1989.

[25] W.A. Woods. Understanding Subsumtion and taxonomy: A framework for progress. In J.F. Sowa, editor, *Principles of Semantic Networks*, pages 45-94. Morgan Kaufmann, San Mateo, CA, 1991.

[26] R. Wille. Bedeutungen von Begriffsverbänden. In: em Beiträge zur Begriffsanalyse, pages 161-211. B. I. Wissenschaftsverlag, Mannheim, 1987.

[27] R.Wille. Concept lattices and conceptual knowledge systems. Computer Math. Applic. Vol. 23, No. 6-9, pp. 493-515, 1992.

[28] B. Ganter und R. Wille *Formale Begriffsanalyse*, 1996. Springer Verlag, Berlin.

[29] R. Wille. Liniendiagramme hierarchischer Begriffssysteme. In: H. H. Bock (Hrsg.): *Anwendungen der Klassifikation: Datenanalyse und numerische Klassifikation,* pages 32-51. INDEKS-Verlag, Frankfurt, 1984.

[30] M. Skorsky. *Endliche Verbände -Diagramme und Eigenschaften.* Dissertation, TH Darmstadt, 1992. Verlag Shaker.

[31] B. Ganter und K. Reuter. *Finding all closed sets: a general approach.* Order 8 (1991), 283-290.

[32] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Logic Programming* 3:267-284, 1984.

[33] J. -L. Guigues and Vincent Duquenne. Familles minimales d'implication informatives resultant d'un tableau de donn'ees binaires. *Math. Sci. Humaines* 95 (1986), 5-18. ($\rightarrow$ 83, 95).

[34] R. Cleaveland and B. Steffen. A linear-time-model-checking algorithm for the alternation-free model mu-calculus. In *Proceeding of the 3rd International Workshop on Computer Aided Verification, CAV'91*, pages 48-58, volume 575 of *Lecture Notes in Computer Science*. Springer-Verlag,1992.