Überprüfung der Erfüllbarkeit im Endlichen:

Ein Verfahren und seine Anwendung

Dissertation im Fach Informatik
an der Fakultät für Mathematik und Informatik
der Ludwig-Maximilians-Universität München
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften

von

Sunna Torge

Tag der Einreichung: 23.10.1998

Überprüfung der Erfüllbarkeit im Endlichen:

Ein Verfahren und seine Anwendung

Dissertation im Fach Informatik
an der Fakultät für Mathematik und Informatik
der Ludwig-Maximilians-Universität München
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften

von

Sunna Torge

Berichterstatter: Prof. Dr. François Bry
Prof. Dr. Rainer Manthey

eingereicht am: 23.10.1998
Tag der mündlichen Prüfung:

Jeden Tag erfuhr ich etwas Neues über den Planeten, über die Abreise und über die Fahrt. Das ergab sich ganz sachte im Laufe meiner Überlegungen.

Antoine de Saint-Exupéry, „Der Kleine Prinz“
(Ins Deutsche übertragen von Grete und Josef Leitgeb)

Abstract

For many applications of automated reasoning, tableaux-methods are very convenient, because they not only detect unsatisfiability but also generate models. However, for most applications these methods suffer from the following major drawback: They sometimes initiate the construction of infinite models, even if finite ones exist.

In this work, the Extended-Positive-Tableaux-Method is introduced, which is based on range restriction and resolution. Instead of relying on Skolemization, the Extended-Positive-Tableaux-Method processes existential quantifiers in a special manner which makes it complete not only for unsatisfiability, but also for finite satisfiability. These results are established in this work.

In addition to the formal introduction of the Extended-Positive-Tableaux-Method, an implementation in Prolog is also presented.

Furthermore, the system SIC (Satisfiable Integrity Constraints) is introduced, which is an interactive prototype to assist in the design of finitely satisfiable integrity constraints. SIC combines two systems, a reasoning component and an interactive visual interface. The reasoning component is an implementation of the Extended-Positive-Tableaux-Method.

Applications of the Extended-Positive-Tableaux-Method in diagnosis and computer linguistics are also discussed in this work.

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlegende Begriffe und Notationen **7**
2.1 Die Prädikatenlogik der ersten Stufe 7
2.1.1 Syntax 7
2.1.2 Prädikatenlogische Formeln in Klausel- und Implikationsform 10
2.1.3 Substitution und Unifikation 12
2.1.4 Semantik 13
2.2 Herbrand-Semantik 16
2.3 Term-Semantik 18
2.4 Graphen und Bäume 21
2.5 Grundlegende Begriffe aus dem Automatischen Beweisen 22
2.5.1 Die Tableaux-Methode 22
2.5.2 Resolution und Hyperresolution 24
2.6 Festlegungen und Notationen 25

3 Die PUHR-Tableaux-Methode **27**
3.1 Bereichsbeschränkte Klauseln 27
3.2 Positive-Unit-Hyperresolution-Tableaux 30
3.3 Implementierung in Prolog: SATCHMO 38
3.4 Erweiterungen der PUHR-Tableaux-Methode 41
3.4.1 Complement-Splitting 42
3.4.2 Beschränkte Suche 45

4 Die EP-Tableaux-Methode **49**
4.1 Motivation für die Erweiterung der Klauselsyntax 50
4.2 Eine erste Erweiterung der Klauselsyntax 53
4.3 Positive Formeln mit restringierter Quantifikation 58
4.4 Extended-Positive-Tableaux 67
4.5 Widerlegungskorrektheit und -vollständigkeit 71
4.6 Die Vollständigkeit für die Erfüllbarkeit im Endlichen 75

5 Implementierung der EP-Tableaux-Methode 87
5.1 Das Programm FINFIMO . . . 87
5.2 Skizze einer alternativen Definition der PRQ-Syntax . . . 94

6 Anwendungen der EP-Tableaux-Methode 99
6.1 Diagnose . . . 99
6.1.1 Konsistenzbasierte Diagnose . . . 100
6.1.2 Abduktive Diagnose . . . 102
6.1.3 Diagnose und EP-Tableaux-Methode . . . 102
6.2 Datenbankschemaentwurf . . . 103
6.3 Natürliche Sprache . . . 104
6.3.1 Die EP-Tableaux-Methode und die kontrollierte Sprache ACE . . . 104
6.3.2 Die EP-Tableaux-Methode in der Linguistik . . . 106

7 Korrektheit von Integritätsbedingungen 111
7.1 Eine Anwendung der EP-Tableaux-Methode . . . 111
7.2 Beispiele von problematischen Integritätsbedingungen . . . 114
7.3 Die Implementierung: SIC . . . 115

8 Abschließende Bemerkungen 123
8.1 Zusammenfassung . . . 123
8.2 Verwandte Arbeiten . . . 124
8.3 Ausblick . . . 126
8.3.1 Weitere Anwendungen . . . 126
8.3.2 Erweiterungen der EP-Tableaux-Methode . . . 128

A CS-SATCHMO 133

B MM-SATCHMO 135

Literaturverzeichnis 135

Abbildungsverzeichnis 145

Index 148

Kapitel 1

Einleitung

Unter den verschiedenen Deduktionsmethoden, die im Bereich der *Automatischen Deduktion* untersucht werden, haben Tableaux-Verfahren etwa seit Anfang der 1980er Jahre wieder zunehmend an Interesse gewonnen. Unabhängig voneinander entwickelten Beth [Bet55], Hintikka [Hin55] und Schütte [Sch56] bereits Mitte der fünfziger Jahre „tableau-ähnliche" logische Kalküle, die die Grundlage für spätere Tableaux-Verfahren [Smu68, Fit90] bilden. Der Name „Tableaux" geht auf Beth zurück [Bet55]. Er stellte die Bäume, die bei der Anwendung eines Tableaux-Verfahren entstehen, als Tabellen (französisch: „Tableaux") dar.

Nachdem Robinson [Rob65b] das Resolutionsprinzip und die Unifikation gefunden hatte, traten die Tableaux-Verfahren in den Hintergrund. Die entscheidende Neuerung des Resolutionsprinzips bestand darin, daß Herleitungen mittels der Unifikation auf der „allgemeinsten Ebene" durchgeführt werden konnten. Damit wurde der Nachteil früherer Kalküle vermieden, den gewaltigen Suchraum, der durch eine Instantiierungsregel erzeugt wird, weitgehend uninformiert durchlaufen zu müssen. Diesen Nachteil hatten insbesondere die Ansätze, die auf dem Herbrand-Theorem basieren [Gil60, Pra60, Pra69, Dav63, CDHM64, Lov68, Lov72, Cha72a, Cha72b], aber auch die bis dahin bekannten Varianten des Tableaux-Verfahrens mit ihrer γ-Regel [Smu68]. Gérard Huet beschrieb das Problem in seiner Habilitationsschrift [Hue76] besonders deutlich: „Les premiers programmes réalisés visaient à l'implémentation du théorème d'Herbrand, par génération de tronçons initiaux de l'univers d'Herbrand d'une proposition. Une telle recherche exhaustive était vouée à l'échec, et cette approche fut abandonnée."[1]

Trotzdem erstarb das Interesse an Tableaux-Verfahren nicht vollständig. Dies liegt zum einen daran, daß Tableaux-Beweise leichter vom Menschen zu verstehen sind. Zum anderen können Tableaux-Verfahren relativ einfach an vorgegebene logische Sprachen angepaßt werden. Dies gilt insbesondere auch für nicht-klassische Logiken wie Temporal- oder Modallogik, was besonders im Hinblick auf Anwendungen von Bedeutung ist.

Als es schließlich in den 1980er Jahren gelang, das Modelleliminationsverfahren [Lov68],

[1] Die ersten entwickelten Programme hatten das Ziel, den Satz von Herbrand durch die Generierung einer Anfangssequenz des Herbranduniversums einer Aussage zu implementieren. Eine solche erschöpfende Suche war zum Scheitern verurteilt, und dieser Ansatz wurde aufgegeben.

die SL-Resolution [KK71] und andere lineare Resolutionsverfahren in einem „Tableaux-Rahmen“ darzustellen [Let98, BF98], wurden Tableaux-Verfahren auch für die Automatische Deduktion wieder interessanter.

Motiviert durch Datenbankanwendungen wurde etwa zeitgleich die *Modellgenerierung* als neues Paradigma in die Automatische Deduktion eingeführt [MB88]. Anstatt wie bisher die Unerfüllbarkeit einer Menge von Formeln der ersten Stufe zu zeigen, war jetzt der Nachweis der Erfüllbarkeit der Formelmenge das Ziel. Dies wurde durch Verfahren zur Generierung einer erfüllenden Herbrandinterpretation, also eines Modells, erreicht. Damit bringt die Anwendung eines Modellgenerierungsverfahrens auf eine Formelmenge nicht nur den Nachweis der Erfüllbarkeit, sondern gleichzeitig auch ein Modell der Formelmenge. Dies ist für die verschiedensten Anwendungen nützlich. Die Modelle von Datenbankintegritätsbedingungen können etwa als „zulässige“ Datenbankzustände betrachtet werden. Modelle der Formalisierung des Diagnoseproblems dienen als Diagnose und Modelle der Formalisierung eines Planungsproblems repräsentieren einen möglichen Ablauf einer Folge von Aktionen.

In der Arbeit von [MB88] wird eine besondere Form der Hyperresolution mit der β- oder „Splitting“-Regel, wie sie aus dem klassischen Tableau [Smu68, Fit90] bekannt ist, kombiniert. In einer Formalisierung dieses Ansatzes, der Positive-Unit-Hyper-Resolution-Tableaux-Methode (kurz PUHR-Tableaux-Methode) [BY96, BFN96], wird diese Verbindung von Resolutions- und Tableaux-Anteilen besonders deutlich. Genau wie klassische Tableaux-Verfahren ist die PUHR-Tableaux-Methode nicht nur vollständig für die Unerfüllbarkeit, sondern sie dient auch zur Modellgenerierung. Das Problem der γ-Regel, das im klassischen Tableaux-Verfahren auftritt, wird dadurch umgangen, daß, wie bei Resolutionsverfahren, mittels der Unifikation die Instantiierung in die „Modus Ponens“-Regel integriert wird.

Die PUHR-Tableaux-Methode verarbeitet bereichsbeschränkte Klauseln in Implikationsform. Aufgrund der syntaktischen Forderung der Bereichsbeschränkung werden Variablen während der Verarbeitung einer Formel „zielgerichtet“ instantiiert und die Unifikation reduziert sich so auf einfaches „matching“ oder „Halbunifikation“.

Genauso wie es im Datenbankbereich und in der Künstlichen Intelligenz üblich ist, repräsentiert auch ein PUHR-Tableau nur den „positiven Teil“ einer Interpretation. Die „negative“ Information bleibt implizit. Auf diese Weise wird der Suchraum reduziert, aber trotzdem repräsentiert die explizite Information eine vollständige Herbrandinterpretation, also gegebenenfalls ein Modell der gegebenen Formelmenge.

Erfüllbarkeit im Endlichen

Die Entwicklung der PUHR-Tableaux-Methode war motiviert durch Anwendungen im Datenbankbereich. So können etwa die erzeugten Modelle einer Menge von Datenbankintegritätsbedingungen als zulässige Datenbankzustände angesehen werden. Die erzeugten Modelle erlauben dann Rückschlüsse darüber, ob die Datenbankintegritätsbedingungen die Gegebenheiten des Anwendungsbereichs adäquat beschreiben. Allerdings sind realisierbare Datenbankzustände im allgemeinen endlich, d.h. in diesem Falle sind nur endliche Modelle

von Interesse oder überhaupt sinnvoll. Dies gilt in ähnlicher Weise auch für verschiedene andere Anwendungen wie etwa die Diagnose.

Deshalb sollte ein für diese Anwendungen entworfenes Verfahren die Frage beantworten können, ob eine gegebene Menge von Formeln ein endliches Modell besitzt, d.h. erfüllbar im Endlichen [EFT92] ist oder nicht. Diese Frage ist aber nicht entscheidbar [Tra50]. Jedoch ist die Erfüllbarkeit im Endlichen semi-entscheidbar, so daß zumindest ein Verfahren denkbar ist, das vollständig für die Erfüllbarkeit im Endlichen ist, d.h. daß für eine gegebene Formelmenge, die ein endliches Modell besitzt, diese Tatsache auch nach endlicher Zeit nachgewiesen wird.

Die Hoffnung, daß es sich bei der PUHR-Tableaux-Methode um ein solches Verfahren handelt, zerschlägt sich aber bei genauerer Betrachtung. Die PUHR-Tableaux-Methode behandelt bereichsbeschränkte Klauseln in Implikationsform. Dies bedeutet insbesondere, daß Existenzquantoren mittels der weithin verbreiteten Methode der Skolemisierung behandelt werden. Anstelle eines Existenzquantors wird ein Funktionssymbol vor der Bearbeitung der betreffenden Formel eingefügt. Zum Beispiel wird statt der Formel $\forall x\ (p(x) \rightarrow \exists y\ q(y))$ die Formel $\forall x\ (p(x) \rightarrow q(f(x)))$ behandelt.

Funktionssymbole in einer Formelmenge führen aber dazu, daß die Anwendung der PUHR-Tableaux-Methode auf die Formelmenge unter Umständen ausschließlich zu unendlichen Modellen führt, obgleich die Formelmenge ein endliches Modell besitzt, also endlich erfüllbar ist. Diese Schwierigkeiten bringen insbesondere auch die durch die Skolemisierung eingeführten Funktionssymbole mit sich. Das heißt aber, daß die PUHR-Tableaux-Methode für die Anwendungen, die ihren Entwurf motivierten, wie etwa die Überprüfung der Korrektheit von Datenbankintegritätsbedingungen, eigentlich nicht geeignet ist.

Diese Beobachtungen bilden den Ausgangspunkt der vorliegenden Arbeit. Die Arbeit entwickelt eine Erweiterung der PUHR-Tableaux-Methode, die sogenannte Extended-Positive-Tableaux-Methode (kurz EP-Tableaux-Methode). Das Ziel beim Entwurf der EP-Tableaux-Methode war es, ein Verfahren zur Verfügung zu stellen, das sowohl vollständig für die Unerfüllbarkeit (wie es auch die PUHR-Tableaux-Methode ist) als auch vollständig für die Erfüllbarkeit im Endlichen ist (was für die PUHR-Tableaux-Methode nicht gilt).

Dabei sollten alle „schönen“ Eigenschaften der PUHR-Tableaux-Methode, also die Vorteile der Bereichsbeschränkung und der Hyperresolution, erhalten bleiben. Die Schwierigkeiten hinsichtlich der Generierung endlicher Modelle, die von den Funktionssymbolen verursacht werden, vermeidet die EP-Tableaux-Methode durch eine explizite Behandlung der Existenzquantoren, d.h. die Skolemisierung wird umgangen. Sonstige Funktionssymbole werden in der zugrundeliegenden Sprache nicht zugelassen. Dies bedeutet aber keine wesentliche Einschränkung, da für einige Anwendungen, wie z.B. der Überprüfung von Datenbankintegritätsbedingungen, Funktionssymbole nicht notwendig sind.

Bereits im klassischen Tableaux-Verfahren [Smu68, Fit90] werden Existenzquantoren explizit behandelt. Dies geschieht durch die sogenannte δ-Regel, die bei jedem Auftreten eines Existenzquantors eine neue Konstante einführt, was aber zu ähnlichen Schwierigkeiten wie die Skolemisierung führt. Deshalb werden Existenzquantoren in der EP-Tableaux-Me-

thode mittels einer erweiterten δ-Regel, der sogenannten δ^*- oder $\exists$-Regel, verarbeitet.

$$\frac{\exists x F(x)}{F(c_1) \mid \ldots \mid F(c_k) \mid F(c_{neu})}$$

wobei $\{c_1, \ldots, c_k\}$ die Menge aller bisherigen Konstanten und c_{neu} eine neue Konstante ist. Die Einführung der Konstante c_{neu} entspricht der klassischen δ-Regel. Allerdings wird gleichzeitig versucht, die existentiell quantifizierte Variable durch bereits vorhandene Konstanten zu ersetzen, um so ein möglichst „kleines“, insbesondere auch endliches Modell zu erhalten. Diese Methode, die Existenzquantoren zu behandeln, wurde bereits in [BM87] in im Datenbankkontext für die Auswertung von zusammengesetzten Termen, u.a. von Skolemtermen, vorgeschlagen. In [Hin88] und [Lor94] wurde die $\exists$-Regel in das klassische Tableaux-Verfahren integriert.

Die Arbeit

Wie oben bereits gesagt wurde, ist die Extended-Positive-Tableaux-Methode (kurz EP-Tableaux-Methode), die in dieser Arbeit vorgestellt wird, eine Erweiterung der PUHR-Tableaux-Methode [BY96].

Die der EP-Tableaux-Methode zugrundeliegende Sprache, die sogenannte PRQ-Syntax, ist ein weitreichendes Fragment einer Sprache der ersten Stufe ohne Funktionssymbole. Insbesondere werden Existenzquantoren explizit zugelassen. Damit steht eine wesentlich flexiblere Sprache zur Verfügung als bei der PUHR-Tableaux-Methode, die eine Klauselsyntax fordert. Besonders im Hinblick auf Anwendungen ist das eine deutliche Verbesserung gegenüber der PUHR-Tableaux-Methode. Die Arbeit diskutiert und begründet die PRQ-Syntax und zeigt, daß die damit definierte Sprache gleichmächtig wie eine Sprache der ersten Stufe ohne Funktionssymbole ist.

Die Arbeit weist die relevanten Eigenschaften der EP-Tableaux-Methode nach. Die Methode ist korrekt und vollständig für die Unerfüllbarkeit sowie modellkorrekt. Das Hauptergebnis dieser Arbeit ist jedoch der Beweis der Vollständigkeit für die Erfüllbarkeit im Endlichen. Diese Eigenschaft macht die EP-Tableaux-Methode besonders interessant für verschiedene Anwendungen.

Ein weiterer Bestandteil der Arbeit ist die Implementierung FINFIMO (FINd FInite MOdells) der EP-Tableaux-Methode in Prolog. Die Implementierung entspricht im Stil der Implementierung SATCHMO [MB88] der PUHR-Tableaux-Methode und ist deshalb sehr kurz. Der Unterschied besteht darin, daß durch die erweiterte Syntax notwendigerweise mehr Fälle betrachtet werden müssen. Ansonsten ist auch die EP-Tableaux-Methode mit Tiefensuche implementiert, wobei wie auch bei SATCHMO die Prolog-eigene Tiefensuche ausgenutzt wird, was zu diesem knappen Programm führt.

Der Entwurf der EP-Tableaux-Methode war durch mögliche Anwendungen motiviert. In dieser Arbeit wird diskutiert, wie die EP-Tableaux-Methode in der Diagnose und in der Linguistik eingesetzt werden kann. Zudem wird die Möglichkeit einer „natürlichsprachlichen“ Schnittstelle für FINFIMO erörtert. Dabei geht es darum, die Eingabe für FINFIMO

in der kontrollierten englischen Sprache ACE [FS96] zu ermöglichen. Darüberhinaus ist die EP-Tableaux-Methode auch zur Konsistenzprüfung von Integritätsbedingungen in Datenbanken einsetzbar. Bevor eine Datenbank bevölkert wird, ist es sinnvoll zu testen, ob die Integritätsbedingungen überhaupt erfüllbar sind, insbesondere ob sie endlich erfüllbar sind. Ist dies nicht der Fall, d.h. wenn es entweder kein Modell oder nur unendliche Modelle für die Integritätsbedingungen gibt, müssen die Integritätsbedingungen geändert werden. Um diesen Prozeß des Entwurfs von Integritätsbedingungen zu unterstützen, wurde das System SIC (Satisfiable Integrity Constraints) entwickelt, das ebenfalls in dieser Arbeit vorgestellt wird.

Der Aufbau der Arbeit ist folgendermaßen:

Nach Bereitstellung der allgemeinen logischen Grundlagen in Kapitel 2 stellt Kapitel 3 die PUHR-Tableaux-Methode sowie ihre Prolog-Implementierung SATCHMO vor. Damit sind die Voraussetzungen für die neuen Beiträge der Arbeit eingeführt.

Kapitel 4 ist der EP-Tableaux-Methode gewidmet. Hier wird die PRQ-Syntax motiviert und eingeführt, das Verfahren definiert und seine Eigenschaften bewiesen: Korrektheit und Vollständigkeit für die Unerfüllbarkeit, Modellkorrektheit und schließlich die Vollständigkeit für die Erfüllbarkeit im Endlichen.

Die Implementierung FINFIMO der EP-Tableaux-Methode in Prolog wird in Kapitel 5 vorgestellt. Zudem wird an dieser Stelle eine Variante der PRQ-Syntax diskutiert, die durch die Implementierung motiviert wurde.

Die Kapitel 6 und 7 behandeln Anwendungen der EP-Tableaux-Methode. In Kapitel 6 wird aufgezeigt, wie die EP-Tableaux-Methode in der Diagnose, beim Entwurf von Datenbankintegritätsbedingungen und im Zusammenhang mit der natürlichen Sprache eingesetzt werden kann. In Kapitel 7 wird dann das System SIC zur Überprüfung der Korrektheit von Integritätsbedingungen vorgestellt.

Kapitel 8 rundet die Arbeit mit einer Reihe von möglichen Anwendungen und Erweiterungen der EP-Tableaux-Methode ab, deren detaillierte Untersuchung im Rahmen dieser Arbeit nicht mehr möglich war.

Kapitel 2

Grundlegende Begriffe und Notationen

In diesem Kapitel werden die für diese Arbeit grundlegenden Begriffe und Notationen eingeführt. Wir beginnen mit der Prädikatenlogik der ersten Stufe, also derjenigen Sprache, die das Fundament dieser Arbeit bildet. Als nächstes wird die Herbrand-Semantik und anschließend eine Variante der Herbrand-Semantik, die Term-Semantik, eingeführt. Sie erweist sich im Hinblick auf die Generierung endlicher Modelle als besonders geeignet. Das Ende dieses Kapitels bildet ein Abschnitt über grundlegende Begriffe des Automatischen Beweisens. Insbesondere wird dort die klassische Tableaux-Methode, die Resolution und die Hyperresolution vorgestellt.

2.1 Die Prädikatenlogik der ersten Stufe

Um die Prädikatenlogik der ersten Stufe einzuführen, wird zunächst die Syntax, also die zugelassenen Zeichen festgelegt. In einem zweiten Schritt wird dann die Semantik, also die Bedeutung der Sprache, eingeführt. Die Definitionen stützen sich weitgehend auf [EFT92] und [BA93]. Für eine ausführlichere Darstellung sei ebenfalls auf [EFT92] verwiesen.

2.1.1 Syntax

Definition 2.1 Das *Alphabet* einer Sprache der ersten Stufe besteht aus folgenden paarweise disjunkten Mengen:

- $\mathcal{R}$, einer (eventuell leeren) Menge von *Relationssymbolen*, die mit $p, q, r, \ldots$ sowie D und Worten, wie es in der Automatischen Deduktion üblich ist, bezeichnet werden. Jedem Relationssymbol ist dabei eine Stelligkeit $n \geq 1$ zugeordnet.

- $\mathcal{F}$, einer (eventuell leeren) Menge von *Funktionssymbolen*, die mit $f, g, h, \ldots$ bezeichnet werden. Jedem Funktionssymbol ist dabei eine Stelligkeit $n \geq 1$ zugeordnet.

- $\mathcal{K}$, einer (eventuell leeren) Menge von *Konstanten*, die mit $a, b, c, \ldots$ bezeichnet werden.
- $\mathcal{V}$, einer abzählbar unendlichen Menge von *Variablen*, die mit $x, y, z, \ldots$, mit oder ohme Index, bezeichnet werden.
- einer Menge von *logischen Junktoren*: $\top$ („wahre Formel“ oder „top“) und $\bot$ („falsche Formel“ oder „bottom“) (nullstellige Verknüpfungen), $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (wenn dann), $\leftrightarrow$ (genau dann wenn).
- den Quantoren $\forall$ (universeller Quantor oder Allquantor) und $\exists$ (existentieller Quantor).
- Klammersymbolen „(“ und „)“ sowie dem Komma „,“.

Die Menge der Relationssymbole, Funktionssymbole und Konstanten bilden die *Signatur* ($\Sigma = \mathcal{R} \cup \mathcal{F} \cup \mathcal{K}$). □

Der Einfachheit halber werden in Definition 2.1 nullstellige Relationssymbole ausgeschloßen. Die Hinzunahme nullstelliger Relationssymbole bringt aber sicherlich keine Schwierigkeiten mit sich. Anstelle von einer Menge von Konstanten werden in Definition 2.1 häufig nullstellige Funktionen zugelassen. Die nullstelligen Funktionen werden dann als Konstanten bezeichnet. Da im folgenden aber meistens Signaturen mit Konstanten, aber ohne Funktionssymbole betrachtet werden, ist die Unterscheidung zwischen Konstanten und Funktionssymbolen sinnvoll.

Sei im folgenden eine Signatur Σ und eine Menge von Variablen $\mathcal{V}$ gegeben. Dann können *Terme* und *Formeln* definiert werden.

Definition 2.2 Die Menge $\mathcal{T}(\Sigma, \mathcal{V})$ der *Terme* ist die kleinste Menge, für die gilt:

T1 Ist x eine Variable aus $\mathcal{V}$, so ist $x \in \mathcal{T}(\Sigma, \mathcal{V})$.

T2 Ist c eine Konstante aus $\mathcal{K}$, so ist $c \in \mathcal{T}(\Sigma, \mathcal{V})$.

T3 Sind $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$ und ist f ein n-stelliges Funktionssymbol, so ist auch $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$. Die Terme $t_1, \ldots, t_n$ im Term $f(t_1, \ldots, t_n)$ werden als *Argumente* von f bezeichnet.

□

Definition 2.3 Die Menge $\mathcal{F}(\Sigma, \mathcal{V})$ der *Formeln* ist die kleinste Menge, für die gilt:

F1 $\top \in \mathcal{F}(\Sigma, \mathcal{V})$ und $\bot \in \mathcal{F}(\Sigma, \mathcal{V})$.

F2 Sind $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$ und ist p ein n-stelliges Relationssymbol, so ist auch $p(t_1, \ldots, t_n) \in \mathcal{F}(\Sigma, \mathcal{V})$. Die Terme $t_1, \ldots, t_n$ in der Formel $p(t_1, \ldots, t_n)$ werden als *Argumente* von p bezeichnet.

F3 Sind $F, G \in \mathcal{F}(\Sigma, \mathcal{V})$, so sind auch $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$ und $(F \leftrightarrow G) \in \mathcal{F}(\Sigma, \mathcal{V})$.

F4 Ist $F \in \mathcal{F}(\Sigma, \mathcal{V})$ und $x \in \mathcal{V}$, so sind auch $\forall x\ F$ und $\exists x\ F\ \in \mathcal{F}(\Sigma, \mathcal{V})$.

□

Formeln, die mit F2 gewonnen werden können, heißen *Atome* oder *atomare* Formeln, die anderen Formeln außer $\top$ und $\bot$ heißen *zusammengesetzte* Formeln. Eine Formel, die keine Quantoren enthält, heißt *quantorenfrei*. Formeln der Gestalt $\forall x_1 \ldots \forall x_n F$ mit quantorenfreiem F werden als *universelle* Formeln bezeichnet. Die Negation $\neg F$ einer Formel F ist eine Abkürzung für die Formel $F \rightarrow \bot$.

Um der Lesbarkeit willen werden wie üblich Klammern in den Formeln durch Präzedenz- und Assoziativregeln eingespart: Quantoren und die Negation binden enger als die zweistelligen logischen Junktoren. Unter diesen bindet die Konjunktion am stärksten. Es folgen die Disjunktion, die Implikation und schließlich die Äquivalenz.

Um auch Teile einer Formel benennen zu können, wird oft der Begriff einer *Teilformel* verwendet.

Definition 2.4 Der Begriff *Teilformel* wird induktiv über den Aufbau einer Formel definiert:

- Für ein Atom F, heißt F selbst *Teilformel von* F.
- Für eine Formel $F * G$ mit $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ heißen sowohl F als auch G *Teilformel von* $F * G$.
- Für eine Formel $\exists x\ F$ heißt F *Teilformel von* $\exists x\ F$.
- Für eine Formel $\forall x\ F$ heißt F *Teilformel von* $\forall x\ F$.

□

Definition 2.5 Die Menge $\text{frei}(t)$ bzw. $\text{frei}(F)$ der in einem Term bzw. in einer Formel *freien Variablen* wird induktiv über den Aufbau eines Terms bzw. einer Formel definiert:

- $\text{frei}(x) := \{x\}$ für eine Variable x.
- $\text{frei}(c) := \emptyset$ für eine Konstante c.
- $\text{frei}(\top) := \emptyset$.
- $\text{frei}(\bot) := \emptyset$.
- $\text{frei}(f(t_1, \ldots, t_n)) := \text{frei}(t_1) \cup \ldots \cup \text{frei}(t_n)$ für ein n-stelliges Funktionssymbol f.
- $\text{frei}(p(t_1, \ldots, t_n)) := \text{frei}(t_1) \cup \ldots \cup \text{frei}(t_n)$ für ein n-stelliges Relationssymbol p.

- $\text{frei}(F * G) := \text{frei}(F) \cup \text{frei}(G)$ für Formeln F und G und $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- $\text{frei}(\forall x\ F) := \text{frei}(F) \setminus \{x\}$ für eine Formel F und eine Variable x.
- $\text{frei}(\exists x\ F) := \text{frei}(F) \setminus \{x\}$ für eine Formel F und eine Variable x.

Formeln ohne freie Variablen heißen *geschlossene Formeln* oder *Sätze*. Atome bzw. Terme ohne freie Variablen heißen *Grundatome* bzw. *Grundterme*. □

2.1.2 Prädikatenlogische Formeln in Klausel- und Implikationsform

Um die systematische Bearbeitung von prädikatenlogischen Formeln zu erleichtern, werden in der Automatischen Deduktion oft Normalformen verwendet. Für diese Arbeit sind in diesem Zusammenhang die *Klauselform* und die *Implikationsform* relevant, die im folgenden eingeführt werden.

Definition 2.6 Ein *Literal* ist eine atomare Formel F (*positives Literal*) oder eine negierte atomare Formel $\neg F$ (*negatives Literal*).

Eine *Klausel* ist eine Formel der Form

$$\forall x_1\ \ldots \forall x_n\ (L_1 \vee \ldots \vee L_k),$$

wobei $L_1, \ldots, L_k$ Literale und $x_1, \ldots, x_n$ alle in $L_1, \ldots, L_k$ auftretenden Variablen sind.

Klauseln werden oft entweder in folgender Klauselnotation

$$\forall x_1\ \ldots \forall x_n\ (\neg A_1 \vee \ldots \vee \neg A_k \vee B_1 \vee \ldots \vee B_m),$$

oder in der sogenannten *Implikationsform* dargestellt:

$$\forall x_1\ \ldots \forall x_n\ (A_1 \wedge \ldots \wedge A_k \rightarrow B_1 \vee \ldots \vee B_m).$$

In beiden Fällen sind $A_1, \ldots, A_k, B_1, \ldots, B_m$ Atome und $x_1, \ldots, x_n$ alle in diesen Atomen auftretenden Variablen. Die Konjunktion $A_1 \wedge \ldots \wedge A_k$ wird als *Rumpf* der Klausel, die Disjunktion $B_1 \vee \ldots \vee B_m$ als *Kopf* der Klausel bezeichnet. Die Formel $\top$ wird als Konjunktion von null Atomen aufgefaßt und kann daher als Rumpf einer Klausel auftreten. Die Formel $\bot$ wird als Disjunktion von null Atomen aufgefaßt und kann daher als Kopf einer Klausel auftreten.

Oft werden Klauseln ohne explizite Allquantoren notiert. Eine Formel der Form

$$(A_1 \wedge \ldots \wedge A_k \rightarrow B_1 \vee \ldots \vee B_m),$$

wobei $A_1, \ldots, A_k$ und $B_1, \ldots, B_m$ Atome sind, wird dann als implizit universell quantifiziert angenommen und dient damit als Kurzschreibweise für die Klausel in Implikationsform

$$\forall x_1\ \ldots \forall x_n\ (A_1 \wedge \ldots \wedge A_k \rightarrow B_1 \vee \ldots \vee B_m),$$

wobei $x_1, \ldots, x_n$ alle in diesen Atomen auftretenden Variablen sind, d.h. für einen Satz.

□

Genauso wie für Klauseln wird im folgenden auch für Formeln eine *Implikationsform* eingeführt. Dazu werden einige Definitionen benötigt.

Definition 2.7 Eine Formel, die eine Konjunktion von Disjunktionen von Literalen ist, heißt *Formel in konjunktiver Normalform.*

Eine Formel der Form

$$Q_1 x_1 \dots Q_n x_n F,$$

wobei $Q_i = \exists$ oder $Q_i = \forall$ für $1 \leq i \leq n$ und F quantorenfrei ist, heißt *Formel in pränexer Normalform.*

Eine Formel in pränexer Normalform heißt *Formel in Implikationsform*, wenn der quantorenfreie Teil F aus einer Konjunktion von Klauseln in Implikationsform besteht. Die gesamte Formel hat dann also die Gestalt

$$\begin{aligned} Q_1 x_1 \dots Q_n x_n \; (A_{11} \wedge \dots \wedge A_{1k_1} \rightarrow B_{11} \vee \dots \vee B_{1m_1}) \wedge \dots \\ \wedge (A_{n1} \wedge \dots \wedge A_{nk_n} \rightarrow B_{n1} \vee \dots \vee B_{nm_n}) \end{aligned}$$

hat. Hierbei sind $Q_i = \exists$ oder $Q_i = \forall$ für $1 \leq i \leq n$ und $A_{11}, \dots, A_{nk_n}$ und $B_{11}, \dots, B_{nm_n}$ Atome. □

Bemerkung 2.8 Eine Formel in Implikationsform kann nicht als Konjunktion von Klauseln aufgefaßt werden, da Existenzquantoren erlaubt sind. Klauseln sind im Gegensatz zu Formeln in Implikationsform universelle Formeln.

An dieser Stelle seien noch einige Ergebnisse über Normalformen zitiert.

Satz 2.9 *Jede quantorenfreie Formel ist logisch äquivalent*[1] *zu einer Formel in konjunktiver Normalform.*

Satz 2.10 *Zu jeder Formel F läßt sich eine logisch äquivalente Formel G in pränexer Normalform mit* $\text{frei}(F) = \text{frei}(G)$ *angeben.*

Hieraus ergibt sich direkt:

Satz 2.11 *Zu jeder Formel F läßt sich eine logisch äquivalente Formel G in Implikationsform mit* $\text{frei}(F) = \text{frei}(G)$ *angeben.*

[1] Die formale Definition von „logisch äquivalent“ findet sich in Abschnitt 2.1.4 Definition 2.22

2.1.3 Substitution und Unifikation

Dieser Abschnitt ist der Ersetzung von Variablen durch Terme gewidmet.

Definition 2.12 Eine *Substitution* ist eine Abbildung $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ von der Menge der Variablen in die Menge der Terme, die nur an endlich vielen Stellen von der Identität abweicht. Entgegen der üblichen Schreibweise für Abbildungen wird bei Substitutionen die Postfix-Schreibweise verwendet. Eine Substitution σ wird als Menge von Paaren $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ dargestellt, falls $t_1, \ldots, t_n$ Terme sind, so daß die Variablen x_i paarweise verschieden sind und für alle x_i mit $1 \leq i \leq n$ die Gleichung $x_i\sigma = t_i$ und $t_i \neq x_i$ gilt. Paare der Form x/x werden also weggelassen. Die Identität, also die „leere“ Substitution, wird mit ϵ bezeichnet. Der Begriff der Substitution wird nun induktiv auf Terme und quantorenfreie Formeln erweitert:

- $c\sigma := c$ für $c \in \mathcal{K}$
- $f(t_1, \ldots, t_n)\sigma := f(t_1\sigma, \ldots, t_n\sigma)$ für $f \in \mathcal{F}$
- $p(t_1, \ldots, t_n)\sigma := p(t_1\sigma, \ldots, t_n\sigma)$ für $p \in \mathcal{R}$
- $\top\sigma := \top$
- $\bot\sigma := \bot$
- $(F * G)\sigma := F\sigma * G\sigma$ für $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$

Sind in einer Substitution $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ alle Terme $t_1, \ldots, t_n$ Grundterme, so wird von einer *Grundsubstitution* gesprochen. □

Definition 2.13 Eine Formel (ein Term) G ist eine *Instanz* einer Formel (eines Terms) F, falls es eine Substitution σ gibt, so daß $G = F\sigma$ ist. Treten in einer Instanz G keine freien Variablen mehr auf, so ist G eine *Grundinstanz* von F. □

Definition 2.14 Ein Atom F *subsumiert* ein Atom G (bzw. eine Disjunktion von Atomen $G_1 \vee \ldots \vee G_n$), falls es eine Substitution σ gibt, so daß $G = F\sigma$ (bzw. $G_i = F\sigma$ für ein $i \in \{1, \ldots, n\}$). □

Insbesondere subsumiert ein Atom alle seine Instanzen.

Definition 2.15 Eine Substitution σ heißt *Matcher* von s nach t für zwei Terme s und t, falls $s\sigma = t$. Eine Substitution σ heißt *Unifikator* von zwei Termen s und t, falls $s\sigma = t\sigma$. Gibt es für s und t einen Unifikator, so heißen s und t *unifizierbar*. Ein Unifikator σ von s und t heißt *allgemeinster Unifikator* von s und t, falls es für jeden Unifikator θ von s und t eine Substitution ρ gibt, so daß $\theta = \sigma\rho$ gilt. □

In [Rob65b] zeigte Robinson, daß es für zwei unifizierbare Objekte immer einen allgemeinsten Unifikator gibt, der bis auf Variablenumbenennung eindeutig ist. Zudem gab er einen Algorithmus zur Berechnung des allgemeinsten Unifikators an. Das Matching (einseitige Unifikation) ist algorithmisch einfacher als die Unifikation. Deshalb ist es vorteilhaft, wenn ein Verfahren nicht die volle Unifikation benötigt, sondern mit Matching auskommt.

2.1.4 Semantik

Nachdem die Syntax der Prädikatenlogik der ersten Stufe und einige in diesem Zusammenhang relevante Begriffe eingeführt wurden, ist dieser Abschnitt nun der Bedeutung der betrachteten Sprache, d.h. der Semantik gewidmet. Dazu wird der Begriff der *Interpretation* eingeführt. Durch eine Interpretation werden Konstanten Objekte, Funktionssymbolen Abbildungen zwischen Objekten und Relationssymbolen Mengen von Objekten zugeordnet.

Definition 2.16 Sei Σ eine Signatur. Eine Σ-*Interpretation* ist ein Paar $\mathcal{I} = (\mathcal{D}, m)$ mit folgenden Eigenschaften:

1. $\mathcal{D}$ ist eine nicht-leere Menge, die sogenannte *Domäne* oder das *Universum* von $\mathcal{I}$.
2. Die *Zuweisungsfunktion* m ist eine auf Σ definierte Abbildung. Für sie gilt:
   (a) Für jedes n-stellige Relationssymbol p aus Σ ist $m(p)$ eine n-stellige Relation über $\mathcal{D}$.
   (b) Für jedes n-stellige Funktionssymbol f aus Σ ist $m(f)$ eine n-stellige Funktion von $\mathcal{D}$ nach $\mathcal{D}$.
   (c) Für jede Konstante c aus Σ ist $m(c)$ ein Element aus $\mathcal{D}$.

□

Ist der Bezug zur Signatur Σ klar oder unwichtig, wird statt von Σ-Interpretationen einfach von Interpretationen gesprochen.

Definition 2.17 Eine *Variablenbelegung* für die Variablenmenge $\mathcal{V}$ bezüglich einer Domäne $\mathcal{D}$ ist eine Abbildung $\beta : \mathcal{V} \to \mathcal{D}$. □

Nachdem der Begriff der Interpretation eingeführt wurde, kann jetzt die *Modellbeziehung* definiert werden. Es sei eine Signatur Σ fest vorgegeben.

Definition 2.18 Sei $\mathcal{I} = (\mathcal{D}, m)$ eine Interpretation und β eine Variablenbelegung. Dann bezeichne $\mathcal{I}_\beta$ folgende Interpretation von Termen:

- $\mathcal{I}_\beta(x) := \beta(x)$ für alle $x \in \mathcal{V}$.
- $\mathcal{I}_\beta(c) := m(c)$ für alle $c \in \mathcal{K}$.
- $\mathcal{I}_\beta(f(t_1, \ldots, t_n)) := m(f)(\mathcal{I}_\beta(t_1), \ldots, \mathcal{I}_\beta(t_n))$ für ein n-stelliges Funktionssymbol $f \in \mathcal{F}$ und Terme $t_1, \ldots, t_n$.

Für eine Belegung β bezüglich einer Domäne $\mathcal{D}$, für $d_i \in \mathcal{D}$ mit $1 \leq i \leq n$ und für Variablen x_i mit $1 \leq i \leq n$ ist $\beta\frac{d_1 \ldots d_n}{x_1 \ldots x_n}$ diejenige Belegung, die x_i auf d_i für $1 \leq i \leq n$ abbildet und für alle von x_i mit $i \in \{1, \ldots, n\}$ verschiedenen Variablen mit β übereinstimmt:

$$\beta\frac{d_1 \ldots d_n}{x_1 \ldots x_n}(y) := \begin{cases} \beta(y) & \text{für } y \neq x_i \text{ für } 1 \leq i \leq n \\ d_i & \text{für } y = x_i \text{ für } i \in \{1, \ldots, n\} \end{cases}$$

Bezeichne $\mathcal{I}_\beta$ die Interpretation $\mathcal{I}$ mit der Belegung β, so sei $\mathcal{I}_{\beta\frac{d_1 \ldots d_n}{x_1 \ldots x_n}} := \mathcal{I}_{\beta\frac{d_1 \ldots d_n}{x_1 \ldots x_n}}$.

Für eine Interpretation $\mathcal{I}_\beta$ und eine Formel F wird die Beziehung $\mathcal{I}_\beta$ *ist Modell von* F ($\mathcal{I}_\beta \models F$) folgendermaßen definiert[2]:

$\mathcal{I}_\beta \models \top$.
$\mathcal{I}_\beta \not\models \bot$, d.h. $\mathcal{I}_\beta \models \bot$ gilt nicht.
$\mathcal{I}_\beta \models p(t_1 \ldots t_n)$:gdw $(\mathcal{I}_\beta(t_1), \ldots, \mathcal{I}_\beta(t_n)) \in m(p)$.
$\mathcal{I}_\beta \models (F \wedge G)$:gdw $\mathcal{I}_\beta \models F$ und $\mathcal{I}_\beta \models G$.
$\mathcal{I}_\beta \models (F \vee G)$:gdw $\mathcal{I}_\beta \models F$ oder $\mathcal{I}_\beta \models G$.
$\mathcal{I}_\beta \models F \to G$:gdw wenn $\mathcal{I}_\beta \models F$, dann $\mathcal{I}_\beta \models G$.
$\mathcal{I}_\beta \models (F \leftrightarrow G)$:gdw $\mathcal{I}_\beta \models F$ genau dann, wenn $\mathcal{I}_\beta \models G$.
$\mathcal{I}_\beta \models \forall x F$:gdw für alle $d \in \mathcal{D}$ gilt $\mathcal{I}_{\beta\frac{d}{x}} \models F$.
$\mathcal{I}_\beta \models \exists x F$:gdw es gibt ein $d \in \mathcal{D}$ mit $\mathcal{I}_{\beta\frac{d}{x}} \models F$.

Statt $\mathcal{I}_\beta$ ist Modell von F werden auch die Sprechweisen $\mathcal{I}_\beta$ *erfüllt* F und F *ist wahr in* $\mathcal{I}_\beta$ verwendet.
Für eine Menge M von Formeln ist eine Interpretation $\mathcal{I}_\beta$ *Modell von* M (in Zeichen $\mathcal{I}_\beta \models M$), falls $\mathcal{I}_\beta \models F$ für jedes $F \in M$ gilt. □

Nach der Definition der Modellbeziehung wird deutlich, daß die intuitive Semantik der Negation $\neg F$ einer Formel F mit der Semantik der Formel $F \to \bot$ übereinstimmt ($\bot$ wird von keiner Interpretation erfüllt). Deshalb ist die Verwendung der Negation $\neg F$ einer Formel F als Abkürzung für die Formel $F \to \bot$ naheliegend.

Bemerkung 2.19 Kommen in einer Formel F keine freien Variablen vor, ist F also ein Satz, so kann bei der Angabe eines Modells von F auf die Variablenbelegung verzichtet werden. Statt $\mathcal{I}_\beta \models F$ kann dann kürzer $\mathcal{I} \models F$ geschrieben werden. Entsprechendes gilt für Mengen von Sätzen. Die gleiche Schreibweise wird benutzt, wenn die Variablenbelegung beliebig ist.

Definition 2.20 Eine Formel F heißt *erfüllbar* genau dann, wenn es eine Interpretation gibt, die ein Modell von F ist. Eine Menge M von Formeln heißt *erfüllbar* genau dann, wenn es eine Interpretation gibt, die ein Modell von jeder Formel $F \in M$ ist.

[2] „gdw“ steht von nun an für „genau dann, wenn“. Der Doppelpunkt vor „gdw“ soll deutlich machen, daß die linke Seite durch die rechte definiert wird.

Eine Formel F heißt *endlich erfüllbar*, falls F ein Modell besitzt, dessen Domäne endlich ist. Eine Menge M von Formeln heißt *endlich erfüllbar*, falls es eine Interpretation mit endlicher Domäne gibt, die ein Modell von jeder Formel $F \in M$ ist.
Eine Formel F (bzw. eine Menge M von Formeln) heißt *unerfüllbar*, falls F (bzw. M) nicht erfüllbar ist.
Eine Formel F (bzw. eine Menge M von Formeln) heißt *allgemeingültig*, falls jede Interpretation ein Modell von F (bzw. M) ist. □

Das folgende Beispiel zeigt, daß die Begriffe „erfüllbar“ und „endlich erfüllbar“ durchaus nicht gleichbedeutend sind.

Beispiel 2.21 Die Formelmenge $F = \{F_1, F_2, F_3, F_4\}$ mit $F_1 = s(a, b)$, $F_2 = \forall xy\ (s(x, y) \rightarrow \exists z\ s(y, z))$, $F_3 = \forall xyz\ (s(x, y) \wedge s(y, z) \rightarrow s(x, z))$ und $F_4 = \forall xy\ (s(x, y) \wedge s(y, x) \rightarrow \bot)$ ist erfüllbar, aber nicht endlich erfüllbar. Jedes Modell von F hat eine unendliche Domäne.

Beispiel 2.21 wird am Ende von Abschnitt 4.6 noch ausführlich behandelt (siehe auch Abbildung 4.3).

Definition 2.22 Eine Formel G *folgt aus* einer Formel F (in Zeichen $(F \models G)$), falls jede Interpretation, die ein Modell von F ist, auch ein Modell von G ist. Eine Formel G *folgt aus* einer Menge M von Formeln (in Zeichen $(M \models G)$), falls jede Interpretation, die ein Modell von M ist, auch ein Modell von G ist.
Zwei Formeln F und G sind *logisch äquivalent*, falls $F \models G$ und $G \models F$. □

Definition 2.23 Zwei Formeln F und G heißen *erfüllbarkeitsäquivalent*, falls F erfüllbarbar ist genau dann, wenn G erfüllbar ist. □

Definition 2.24 Eine Formel F heißt *konsistent*, falls $F \not\models \bot$. Eine Menge M von Formeln heißt *konsistent*, falls $M \not\models \bot$. □

Die Begriffe erfüllbar und konsistent werden teilweise synonym verwandt. Der Grund dafür ist folgendes Lemma.

Lemma 2.25 *Eine Menge M ist konsistent genau dann, wenn M erfüllbar ist.*

Beweis: Angenommen, M ist nicht erfüllbar. Dann gibt es keine Interpretation, die ein Modell von M ist. Also ist jedes Modell von M auch ein Modell von $\bot$, d.h. $M \models \bot$.
Angenommen, M ist erfüllbar. Dann gibt es eine Interpretation $\mathcal{I}$, die ein Modell von M ist. Nach Definition 2.18 ist keine Interpretation ein Modell von $\bot$, insbesondere $\mathcal{I} \not\models \bot$. Also gilt $M \not\models \bot$, d.h. M ist konsistent. ∎

Schließlich wird noch der Begriff des *Redukts* eingeführt. Hierbei handelt es sich um eine Modifikation des bekannten Begriffs einer Subinterpretation [EFT92]. Es wird nicht nur eine Teilmenge der Domäne (wie im Fall einer Subinterpretation [EFT92]) einer Interpretation sondern auch eine Teilmenge der Signatur betrachtet.

Definition 2.26 Seien Σ und Σ' Signaturen mit $\Sigma \subseteq \Sigma'$ und $(\mathcal{D}', m')$ eine Σ'-Interpretation. Dann heißt $(\mathcal{D}, m)$ ein Σ-*Redukt* von $(\mathcal{D}', m')$, falls gilt:

1. $\mathcal{D} \subseteq \mathcal{D}'$ und $\mathcal{D}$ ist nicht leer.
2. Für jede Konstante $c \in \Sigma$ ist
$$m(c) = \begin{cases} m'(c), \text{ falls } m'(c) \in \mathcal{D} \\ d \text{ für ein festes } d \in \mathcal{D}, \text{ sonst} \end{cases}$$
3. Für jedes n-stellige Relationssymbol $p \in \Sigma$ ist
$$m(p) = m'(p) \cap \mathcal{D}^n.$$

□

Direkt aus dieser Definition folgt ein Hilfslemma, welches in Abschnitt 3.1 und Abschnitt 4.3 benötigt wird.

Lemma 2.27 *Sei $(\mathcal{D}', m')$ eine Σ'-Interpretation, $\Sigma \subseteq \Sigma'$ und $(\mathcal{D}, m)$ ein Σ-Redukt von $(\mathcal{D}', m')$. Dann ist $(\mathcal{D}, m)$ eine Σ-Interpretation.*

Beweis:

1. Definition 2.26 des Σ-Redukts ist $\mathcal{D}$ nicht leer.
2. Für jede Konstante c aus Σ ist $m(c)$ ein Element aus der Domäne $\mathcal{D}$.
3. Für jedes n-stellige Relationssymbol $p \in \Sigma$ ist $m(p) = m'(p) \cap \mathcal{D}^n$ eine n-stellige Relation über $\mathcal{D}$.

■

2.2 Herbrand-Semantik

In diesem Abschnitt wird der Begriff der *Herbrandinterpretation* eingeführt. Bei einer Herbrandinterpretation liegt die Interpretation der Konstanten und Funktionssymbole fest. Über die Interpretation von Relationssymbolen wird nichts ausgesagt, sie ist frei wählbar. Im folgenden sei Σ eine festgelegte Signatur, die mindestens eine Konstante enthält. Dies ist notwendig, da die Domäne einer Herbrandinterpretation über Σ die Menge aller Grundterme ist, die aus Symbolen in Σ bildbar sind. Diese Menge wird mit T^Σ bezeichnet. Um eine nichtleere Domäne einer Herbrandinterpretation zu gewährleisten, muß also mindestens eine Konstante in Σ existieren.

Definition 2.28 Eine Σ-Interpretation $\mathcal{H} = (\mathcal{D}, m)$ heißt *Herbrandinterpretation* :gdw

(i) $\mathcal{D} = T^\Sigma$. $\mathcal{D}$ heißt *Herbranduniversum.*

(ii) Für n-stelliges $f \in \Sigma$ und $t_1, \ldots, t_n \in T^\Sigma$ ist $m(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.

(iii) Für $c \in \Sigma$ ist $m(c) = c$.

□

Die Interpretation der Relationssymbole in einer Herbrandinterpretation $\mathcal{H}$ ist frei wählbar. Sie kann eindeutig festgelegt werden durch die Angabe der Menge $\mathcal{G}$ der Grundatome, die in $\mathcal{H}$ wahr sind. Um dies zu verdeutlichen, wird für eine Herbrandinterpretation $\mathcal{H}$, deren Interpretation der Relationssymbole durch die Menge der Grundatome $\mathcal{G}$ festgelegt ist, oft $\mathcal{H}(\mathcal{G})$ geschrieben. Für ein Relationssymbol p und ein Tupel $(t_1, \ldots, t_n) \in \mathcal{D}^n$ ist dann $(t_1, \ldots, t_n) \in m(p)$:gdw $p(t_1, \ldots, t_n) \in \mathcal{G}$. Damit ist mit den bisherigen Definitionen 2.16 und 2.18 für jeden Satz F festgelegt, ob $\mathcal{H}(\mathcal{G}) \models F$ gilt oder nicht.

Satz 2.29 *Sei $\mathcal{G}$ eine Menge von Grundatomen und $\mathcal{H}(\mathcal{G})$ eine Σ-Herbrandinterpretation. Dann gilt für einen Satz F folgendes:*

1. *Ist F ein Grundatom, so ist $\mathcal{H}(\mathcal{G}) \models F$ gdw $F \in \mathcal{G}$.*

2. *Ist F eine universelle Formel $\forall x_1 \ldots \forall x_n H$, so gilt:*
   $\mathcal{H}(\mathcal{G}) \models \forall x_1 \ldots \forall x_n H$ *gdw für alle Grundterme $(t_1, \ldots, t_n) \in (T^\Sigma)^n$ und jede Substitution $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ gilt $\mathcal{H}(\mathcal{G}) \models H\sigma$.*

   Insbesondere ist $\mathcal{H}(\mathcal{G}) \models G$ für jede Grundinstanz G von H.

Beweis:

1. Die Interpretation von Relationen wird über die Menge $\mathcal{G}$ eindeutig festgelegt.

2. $\mathcal{H}(\mathcal{G}) \models \forall x_1 \ldots \forall x_n H$
   gdw
   für eine beliebige Variablenbelegung β und für alle $t_1, \ldots, t_n \in (T^\Sigma)^n$ gilt, daß $\mathcal{H}(\mathcal{G})_{\beta\frac{t_1 \ldots t_n}{x_1 \ldots x_n}} \models H$
   gdw
   für jede Substitution $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ mit $t_i \in T^\Sigma$ für $i = 1, \ldots, n$, also für jede Grundinstanz G von H, gilt, daß $\mathcal{H}(\mathcal{G}) \models F\sigma$.

■

Bemerkung 2.30 Wie bereits im Beweis des obigen Satzes deutlich wurde, kann eine Variablenbelegung zu einer Herbrandinterpretation stets durch eine Grundsubstitution repräsentiert werden.

Herbrandinterpretationen spielen im Bereich der Automatischen Deduktion eine zentrale Rolle, da einerseits die Terme, also syntaktische Objekte, mit den Elementen der Domäne, die die Semantik bestimmen, übereinstimmen. Dadurch wird eine Generierung von Interpretationen allein aufgrund von syntaktischen Informationen möglich.

Zum anderen bedeutet die Betrachtung von Herbrandinterpretationen zumindest für universelle Formeln ohne Gleichheit keinerlei Einschränkung. Es gilt folgender Satz:

Satz 2.31 *Sei M eine erfüllbare Menge universeller und gleichheitsfreier Sätze. Dann hat M ein Herbrandmodell, d.h. ein Modell, das eine Herbrandinterpretation ist.*

Dieser Satz kann jedoch nicht auf endlich erfüllbare Mengen übertragen werden, wie das folgende Beispiel zeigt.

Beispiel 2.32 Wir betrachten eine Signatur Σ mit abzählbar vielen Konstanten und einem einstelligen Relationssymbol p. Die universelle gleichheitsfreie Formel $F = \forall x\ p(x)$ ist endlich erfüllbar, da die Σ-Interpretation $\mathcal{I} = (\mathcal{D}, m)$ mit $\mathcal{D} = \{d\}$ und $m(p) = \{d\}$ ein Modell der Formel F ist. Die Interpretation der anderen Symbole aus Σ ist hier beliebig. Die Formel F hat jedoch kein endliches Σ-Herbrandmodell, da jede Σ-Herbrandinterpretation aufgrund der abzählbar vielen Konstanten in Σ eine unendliche Domäne besitzt.

2.3 Term-Semantik

Da es in dieser Arbeit vornehmlich um endliche Modelle gehen soll, ist der Begriff der Herbrandinterpretation nicht ganz geeignet. Enthält die festgelegte Signatur ein Funktionssymbol oder unendlich viele Konstanten, so ist jede Herbrandinterpretation über dieser Signatur unendlich. Gleichzeitig ist es aber nicht immer unbedingt notwendig, alle Domänenelemente explizit zur Verfügung zu haben. Oft genügt ein (endlicher) Teil des Herbranduniversums.

Beispiel 2.33 Wir betrachten wieder eine Signatur Σ mit abzählbar vielen Konstanten und einem einstelligen Relationssymbol p. Damit hat jede Σ-Herbrandinterpretation eine unendliche Domäne. Sei $F = \exists x\ p(x)$. Die Formel F wird von einer Σ-Herbrandinterpretation $\mathcal{H}(\mathcal{G})$ erfüllt, falls $\mathcal{G}$ ein Grundatom $p(a)$ für eine Konstante a enthält. Insbesondere wird F von der Σ-Herbrandinterpretation $\mathcal{H}(\{p(a)\})$ erfüllt. In diesem Fall ist allein das Domänenelement a von Interesse. Gleichzeitig ist die Domäne von $\mathcal{H}(\{p(a)\})$ aber wegen der abzählbar vielen Konstanten in Σ unendlich.

Diese Überlegungen führen zum Begriff einer *Terminterpretation*. Der wesentliche Unterschied zwischen einer Terminterpretation und einer Herbrandinterpretation besteht in der Definition der Domäne. Daraus resultieren dann auch Unterschiede in der Definition der Interpretation der Konstanten. Im Deutschen werden die Begriffe *Terminterpretation* und *Herbrandinterpretation* manchmal synonym verwandt. Deshalb sei hier ausdrücklich darauf hingewiesen, daß dies hier nicht der Fall ist.

Eine Terminterpretation wird ausschließlich bezüglich Signaturen *ohne* Funktionssymbole definiert. Die Interpretation der Relationssymbole kann wie bei Herbrandinterpretationen durch die Angabe einer Menge von Grundatomen eindeutig festgelegt werden.

Definition 2.34 Sei F eine Formel und c eine Konstante. Dann *tritt c in F auf*, falls F eine Teilformel T besitzt, so daß c ein Argument von T ist. Sei M eine Menge von Formeln und c eine Konstante. Dann *tritt c in M auf*, falls es eine Formel F in M gibt, so daß c in F auftritt. □

Definition 2.35 Sei $\mathcal{G}$ eine Menge von Grundatomen über der Signatur Σ. Weiter gebe es mindestens eine Konstante in Σ, die nicht in $\mathcal{G}$ auftritt.[3] Dann heißt die im folgenden definierte Σ-Interpretation Σ-*Terminterpretation* $\mathcal{T}(\mathcal{G})$:

(i) Die Domäne von $\mathcal{T}(\mathcal{G})$ ist die Menge $\mathcal{D} = \mathcal{K}_\mathcal{G} \cup \{d\}$, wobei $\mathcal{K}_\mathcal{G} := \{c |\ c \text{ tritt in } \mathcal{G} \text{ auf}\}$ sowie $d \in \Sigma$ und $d \neq c$ für alle Konstanten c in $\mathcal{K}_\mathcal{G}$ ist.

(ii) Für $c \in \Sigma$ ist

$$m(c) = \begin{cases} c, & \text{für alle } c \in \mathcal{K}_\mathcal{G} \\ d, & \text{sonst} \end{cases}$$

d.h. eine Konstante $c \in \Sigma$ wird durch c interpretiert, falls c in $\mathcal{G}$ auftritt, sonst durch das Element d.

(iii) Für ein Grundatom F gilt: $\mathcal{T}(\mathcal{G}) \models F$:gdw $F \in \mathcal{G}$.

. □

In Definition 2.35 wird in der Definition der Domäne zwischen zwei Arten von Elementen unterschieden, den Konstanten der Menge der Grundatome, durch die die Terminterpretation repräsentiert wird sowie einem zusätzlichen Element. Das folgende Beispiel zeigt die Notwendigkeit für das zusätzliche Element.

Beispiel 2.36 Die leere Terminterpretation $\mathcal{T}(\emptyset)$ ist ein Termmodell der Formel $p(a) \rightarrow \bot$. Da die Menge der Grundatome, die das Termmodell repräsentiert, leer ist, besteht die Domäne von $\mathcal{T}(\emptyset)$ aus dem einzigen zusätzlichen Element d. Dies ist notwendig, um die Konstante a zu interpretieren.

Dieses zusätzliche Element in der Domäne einer Terminterpretation ist also notwendig. Gleichzeitig ist es aber auch beliebig wählbar. Deshalb ist es sinnvoll, zwei Terminterpretationen $\mathcal{T}(\mathcal{G}_1)$ und $\mathcal{T}(\mathcal{G}_2)$ als gleich anzusehen, wenn $\mathcal{G}_1 = \mathcal{G}_2$ gilt.

Wird eine Terminterpretation $\mathcal{T}(\mathcal{G}_\infty)$ durch eine unendliche Menge $\mathcal{G}_\infty$ von Grundatomen repräsentiert, so ist ihre Domäne unendlich, vorausgesetzt, die zugrundeliegende Signatur enthält nur endlich viele Relationssymbole. Da im folgenden aber meist endliche Mengen von Formeln betrachtet werden, kann dies angenommen werden.

[3] Diese Einschränkung scheint zwar gravierend, ist aber für unsere Belange irrelevant.

Bemerkung 2.37 Die Verwendung des Begriffs Terminterpretation ist nur im Zusammenhang mit bereichsbeschränkten Klauseln oder allgemeineren bereichsbeschränkten Formeln, wie sie in Kapitel 4 definiert werden, sinnvoll. Bereichsbeschränkung ist eine syntaktische Forderung. Bei quantifizierten Formeln wird der „Bereich", über den quantifiziert wird, durch „positive" Teilformeln vorgegeben. Die Bereichsbeschränkung hat jedoch keine Auswirkungen hinsichtlich der Erfüllbarkeit einer Formelmenge. Werden dagegen beliebige Formeln der ersten Stufe betrachtet, so treten bei universell quantifizierten Formeln Schwierigkeiten auf. Eine Formel $\forall x p(x)$ wird von einer Interpretation $\mathcal{I}_\beta$ erfüllt, falls für alle Elemente e der Domäne $\mathcal{I}_{\beta\frac{e}{x}} \models p(x)$ gilt. Für eine Terminterpretation und eine Belegung β muß dann insbesondere $\mathcal{T}(\mathcal{G})_{\beta\frac{d}{x}} \models p(x)$ für das zusätzliche Domänenelement d gelten. Damit ist aber $\mathcal{T}(\mathcal{G})$ ein Modell des Grundatoms $p(d)$. Also ist $p(d) \in \mathcal{G}$ und d ein Element von $\mathcal{K}_\mathcal{G}$. Das steht aber im Widerspruch zur Definition der Domäne einer Terminterpretation. Werden nur bereichsbeschränkte Klauseln oder allgemeinere bereichsbeschränkte Formeln betrachtet, so tritt dieses Problem nicht auf.

Bemerkung 2.38 Genauso wie bei Herbrandinterpretationen kann eine Variablenbelegung zu einer Terminterpretation stets durch eine Grundsubstitution repräsentiert werden, da die Elemente der Domäne Konstanten sind.

Den Abschluß dieses Abschnittes bildet die Einführung eines Minimalitätsbegriffs auf der Menge der Termmodelle einer Formel oder Formelmenge.

Definition 2.39 Durch die Teilmengenbeziehung $\subseteq$ wird eine Ordnung $\leq$ auf Terminterpretationen induziert: $\mathcal{T}(\mathcal{G}_1) \leq \mathcal{T}(\mathcal{G}_2)$:gdw $\mathcal{G}_1 \subseteq \mathcal{G}_2$. Ein Termmodell einer Menge von Formeln heißt *minimal*, wenn es minimal bezüglich $\leq$ ist. □

Der oben definierte Minimalitätsbegriff führt nicht zu einem eindeutigen minimalen Modell. So hat z.B. die Formel $p(a) \vee q(b)$ zwei minimale Termmodelle $\mathcal{T}(\{p(a)\})$ und $\mathcal{T}(\{q(b)\})$.

Bemerkung 2.40 In Definition 2.39 wird keine Aussage über die Domänen der betrachteten Terminterpretationen ausgesagt. Jedoch gilt für zwei Terminterpretationen $\mathcal{T}(\mathcal{G}_1)$ und $\mathcal{T}(\mathcal{G}_2)$ mit den Domänen $\mathcal{D}_1$ und $\mathcal{D}_2$ folgende Beziehung: Gilt $\mathcal{T}(\mathcal{G}_1) \leq \mathcal{T}(\mathcal{G}_2)$, so gilt auch $\mathcal{D}_1 \backslash \{d_1\} \subseteq \mathcal{D}_2 \backslash \{d_2\}$. Hierbei sind d_1 und d_2 die zusätzlichen Domänenelemente. Die Umkehrung dieser Aussage ist jedoch falsch, wie das folgende Beispiel zeigt.

Beispiel 2.41 Die Formel $p(a) \wedge p(b) \ \vee \ q(a)$ hat zwei minimale Termmodelle, nämlich $\mathcal{T}(\{p(a), p(b)\})$ und $\mathcal{T}(\{q(a)\})$. Obwohl die Domäne des zweiten Termmodells eine Teilmenge der Domäne des ersten Modells ist (abgesehen von den beiden Zusatzelementen), so sind die beiden Termmodelle trotzdem nicht bezüglich $\leq$ vergleichbar.

2.4 Graphen und Bäume

In diesem Abschnitt wird der Begriff des *Graphen* und der Begriff des *Baumes*, eines speziellen Graphen, eingeführt. Bäume werden häufig dazu genutzt, um Tableauxverfahren zu repräsentieren. Auch für die in dieser Arbeit vorgestellten Tableauxverfahren sind Bäume als Repräsentationsstrukturen besonders geeignet.

Definition 2.42 $G = (V, E)$ heißt *gerichteter Graph*, falls gilt:

1. V ist eine (möglicherweise unendliche) Menge. Die Elemente von V heißen *Knoten*.
2. $E \subseteq V \times V$ ist eine zweistellige Relation auf V. Die Elemente in E heißen *Kanten*.

□

Definition 2.43 Sei $G = (V, E)$ ein gerichteter Graph und $k = (v_0, \ldots, v_n, \ldots)$ mit $n \geq 1$ eine (möglicherweise unendliche) Folge von Knoten.

1. k heißt *Pfad*, wenn $(v_i, v_{i+1}) \in E$ für $i \geq 0$ und $v_i \neq v_j$ für alle $i, j \geq 0$ mit $i \neq j$.
2. k heißt *Pfad von v* nach w, falls v und w Knoten sind, k ein endlicher Pfad $(v_0, \ldots, v_n)$ ist und $v_0 = v$ und $v_n = w$ gilt.
3. k heißt *Zyklus*, wenn k ein endlicher Pfad $(v_0, \ldots, v_n)$ ist, jedoch $v_0 = v_n$ gilt.
4. Ein Graph ohne Zyklen heißt *zyklenfrei*.

□

Aufbauend auf der Definition eines Graphen können nun *Bäume* definiert werden.

Definition 2.44 Ein gerichteter Graph $B = (V, E)$ heißt *Baum*, falls folgendes gilt:

1. B ist zyklenfrei.
2. Es gibt einen Knoten $r \in V$, so daß für alle $v \in V$ genau ein Pfad von r nach v existiert. r heißt *Wurzel* von B.

□

Definition 2.45 Sei $B = (V, E)$ ein Baum.

- Ein Knoten w heißt *Kindknoten* von v, falls $(v, w) \in E$. Der Knoten v heißt *Elternknoten* von w.
- Für einen Pfad k, der im Knoten v beginnt, heißen alle Knoten des Pfades *Nachfolgerknoten* von v. Der Knoten v ist *Vorgängerknoten* aller Knoten in k.
- Ein Knoten, der keine Nachfolgerknoten besitzt, heißt *Blatt*.

- Ein (möglicherweise unendlicher) Pfad $k = (v_0, \ldots, v_n, \ldots)$ heißt *Zweig*, falls k an der Wurzel r des Baumes beginnt ($v_0 = r$) und falls es für jeden Knoten v des Pfades, der einen Nachfolgerknoten besitzt, auch einen Nachfolgerknoten in k gibt.

Sei $\{v_0, v_1, v_2, \ldots\}$ die Menge der Kindknoten eines Knoten u. Dann wird durch $<$ eine Ordnung $<_k$ auf den Knoten induziert:

$$v_i <_k v_j \text{ :gdw } i < j \text{ für } i, j \geq 0.$$

Für zwei Knoten v und w mit dem gleichen Elternknoten heißt v *links von* w, falls $v <_k w$ gilt. v heißt *rechts von* w, falls $w <_k v$ gilt. Ein Zweig Z_1 heißt *links von* (bzw. *rechts von*) einem Zweig Z_2, falls ein gemeinsamer Knoten u von Z_1 und Z_2[4] zwei Kindknoten v_1 und v_2 besitzt, so daß v_1 ein Element aus Z_1 ist, v_2 ein Element aus Z_2 ist und $v_1 <_k v_2$ (bzw. $v_2 <_k v_1$) gilt. □

2.5 Grundlegende Begriffe aus dem Automatischen Beweisen

Die Grundlage für jedes automatische Beweisen bildet der Vollständigkeitssatz von Gödel [Göd30], der besagt, daß dem semantischen Folgerungsbegriff ein syntaktischer Ableitungsbegriff entspricht. Dieser syntaktische Ableitungsbegriff wird mittels des Begriffs eines *Kalküls* definiert. Ein Kalkül setzt eine formale Sprache voraus, die in dieser Arbeit die Prädikatenlogik der ersten Stufe sein wird, und besorgt syntaktische Operationen, um aus einer Menge von Formeln neue Formeln abzuleiten. Die Grundlage für die Operationen bilden sogenannte Schlußregeln, die folgende Form haben:

$$\frac{\mathcal{F}_1 \ldots \mathcal{F}_n}{\mathcal{F}}$$

Eine Anwendung einer Schlußregel ist möglich, falls die Formeln $\mathcal{F}_1 \ldots \mathcal{F}_n$ entweder gegeben oder bereits abgeleitet sind. Aufgrund der Anwendung der Schlußregel wird zusätzlich die Formel $\mathcal{F}$ abgeleitet.

Im folgenden werden die Kalküle, die für diese Arbeit von Bedeutung sind, vorgestellt. Die Darstellung lehnt sich an [GHR93] an.

2.5.1 Die Tableaux-Methode

Die Tableaux-Methode geht auf Beth [Bet55], Hintikka [Hin55] und Schütte [Sch56] zurück und wurde von Smullyan [Smu68] und Fitting [Fit90] weiterentwickelt. Die Tableaux-Methode und ihre Varianten werden auch *analytische Tableaux* genannt, da die Anwendung der Tableaux-Methode auf eine Formel oder Formelmenge gleichbedeutend ist mit einer systematischen Zerlegung der Formel(n) in ihre Teilformeln.

[4] Dieser existiert immer, da die Wurzel ein gemeinsamer Knoten von allen Zweigen ist

α	α_1	α_2
$\neg\neg F$	F	F
$F \wedge G$	F	G
$\neg(F \vee G)$	$\neg F$	$\neg G$

β	β_1	β_2
$F \vee G$	F	G
$\neg(F \wedge G)$	$\neg F$	$\neg G$

γ	$\gamma(t)$
$\forall x F$	$F\{t/x\}$
$\neg\exists x F$	$\neg F\{t/x\}$

δ	$\delta(c)$
$\exists x F$	$F\{c/x\}$
$\neg\forall x F$	$\neg F\{a/x\}$

Abbildung 2.1: Einteilung in α-, β-, γ- und δ-Formeln

Die Tableaux-Methode wurde als Refutationsmethode formuliert mittels derer die Unerfüllbarkeit einer Formel oder Formelmenge gezeigt werden kann. Die grundlegende Idee ist, mit der zu widerlegenden Formel zu starten und diese solange in ihre Teilformeln (modulo einer Art von Substitution) zu zerlegen, bis eine Formel und ihr Negat erzeugt und damit ein Widerspruch hergeleitet wird. Um diese Zerlegung in Teilformeln einfach beschreiben zu können, werden die Formeln in der Tableaux-Methode in fünf Kategorien – Literale, α-, β-, γ- und δ-Formeln – eingeteilt (siehe Abbildung 2.1). Mit dieser Einteilung, die auf Smullyan zurückgeht, können die Schlußregeln der Tableaux-Methode (auch *Tableaux-Expansionsregeln*) mittels derer ein Tableau nach und nach aufgebaut wird, einfach beschrieben werden.

Definition 2.46 (Tableaux-Regeln)

α-Regel $\quad \dfrac{\alpha}{\alpha_1}$ und $\dfrac{\alpha}{\alpha_2}$

β-Regel $\quad \dfrac{\beta}{\beta_1 \mid \beta_2}$

γ-Regel $\quad \dfrac{\gamma}{\gamma(t)}$, wobei t ein Grundterm ist.

δ-Regel $\quad \dfrac{\delta}{\delta(c)}$, wobei c eine neue Konstante ist. □

Abbildung 2.2 zeigt ein Tableau für die Formel $\forall x\ p(x) \vee \exists y\ (q(a,y) \wedge r(y,b))$, in dem kein Widerspruch hergeleitet wurde. Die Verzweigung von 1. nach 2. entsteht durch eine Anwendung der β-Regel auf die gegebene Formel $\forall x p(x) \vee \exists y(q(a,y) \wedge r(y,b))$. Auf der linken Seite des Tableau wurde von 2. nach 3. und von 2. nach 4. jeweils die γ-Regel auf die Formel $\forall x\ p(x)$ in 2. angewendet. Auf der rechten Seite führt eine Anwendung der δ-Regel von 2. nach 3. wobei die neue Variable c_{neu} eingeführt wird. Eine Anwendung der α-Regel auf die Teilformel $q(a, c_{neu}) \wedge q(c_{neu}, b)$ führt sowohl von 3. nach 4. als auch von 3. nach 5.

Unter gewissen Bedingungen ist es möglich, daß die in einem Zweig eines Tableau auftretenden Grundatome ein Modell der gegebenen Formel oder Formelmenge repräsentieren.

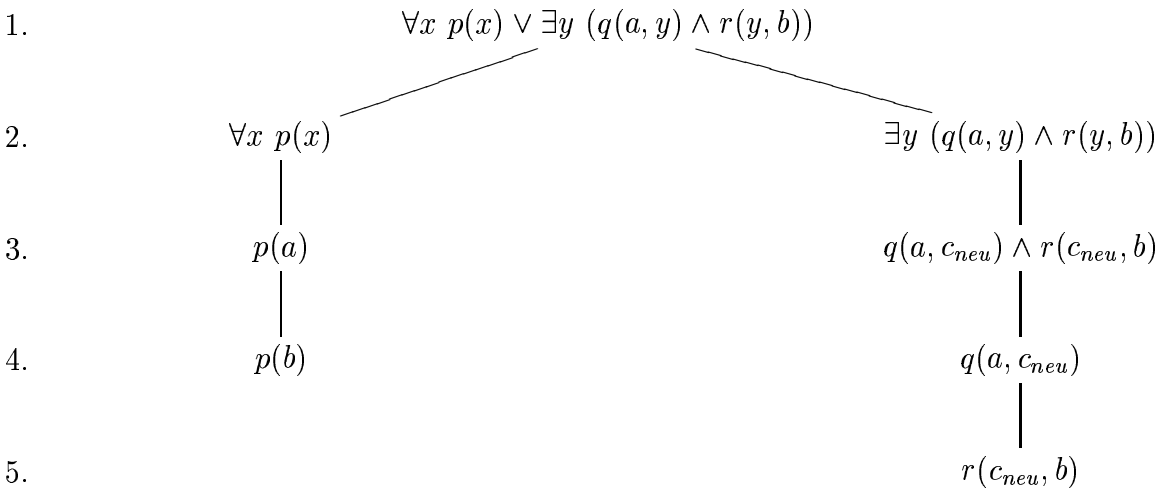


Abbildung 2.2: Ein Tableau für die Formel $\forall x\ p(x) \vee \exists y\ (q(a,y) \wedge r(y,b))$.

Wird allerdings die Erzeugung von Modellen einer Formelmenge angestrebt, so ist die hier beschriebene Tableaux-Methode nicht besonders gut geeignet, da aufgrund der γ-Regel die betreffenden Teilformeln beliebig instantiiert werden können. Dies kann zu unendlichen Zweigen führen, ohne daß ein Modell gefunden wird.

2.5.2 Resolution und Hyperresolution

Die Resolutionsmethode, die auf Robinson [Rob65b] zurückgeht, benutzt die formale Sprache der Klauseln und ist ebenfalls ein Refutationsverfahren. Der Kalkül stellt zwei Schlußregeln zur Verfügung, die Resolutionsregel und die Faktorisierungsregel, mit denen eine Klauselmenge um weitere Klauseln angereichert werden kann. Die ursprüngliche Klauselmenge ist genau dann unerfüllbar, wenn die leere Klausel, d.h. die Disjunktion von null Atomen, ableitbar ist.

Für diese Arbeit spielt die Faktorisierungsregel keine Rolle. Deshalb wird hier nur die Resolutionsregel erläutert.

Definition 2.47 (Resolutionsregel)

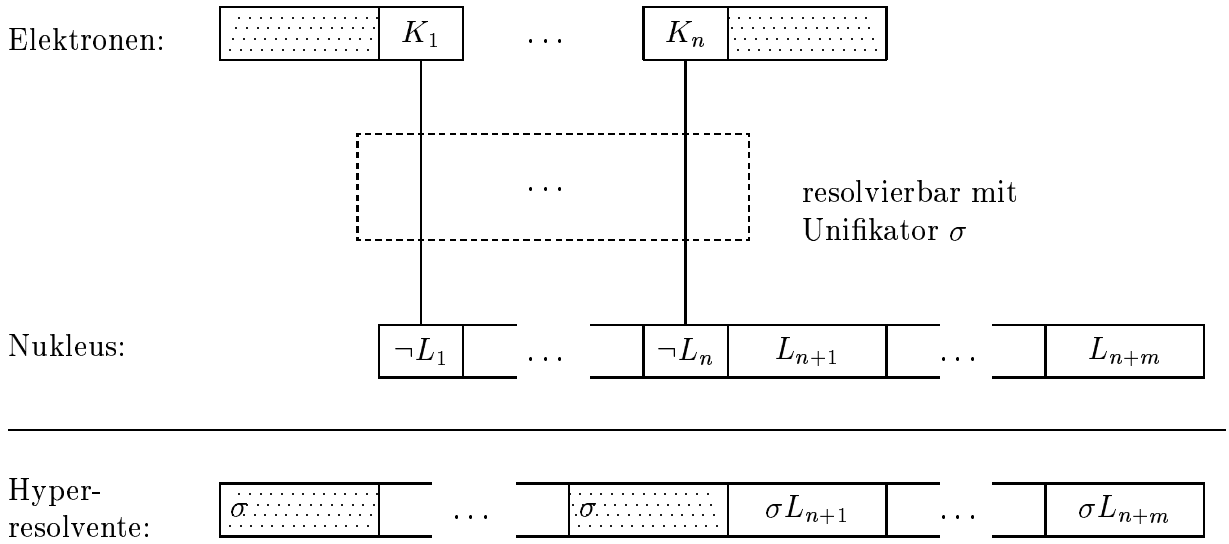
$$\frac{\begin{array}{ll} L \vee K_1 \vee \ldots \vee K_k & \sigma \text{ ist allgemeinster Unifikator} \\ \neg L' \vee M_1 \vee \ldots \vee M_m & \text{von } L \text{ und } L' \end{array}}{K_1\sigma \vee \ldots \vee K_k\sigma \vee M_1\sigma \vee \ldots \vee M_m\sigma}$$

$K_1\sigma, \ldots, K_k\sigma, M_1\sigma, \ldots, M_m\sigma$ wird als *Resolvente* bezeichnet. □

Die folgende Abbildung zeigt eine einfache Anwendung der Resolutionsregel:

$$\frac{\begin{array}{ll} p(a,y) \vee r(y) & \\ \neg p(x,b)) \vee q(x) & \sigma = \{a/x, b/y\} \end{array}}{r(b) \vee q(a)}$$

Um den Suchraum der Resolutionmethode weiter zu verkleinern, wurden eine ganze Reihe von Varianten der Resolutionmethode entwickelt. Eine dieser Varianten ist die Methode der *Positiven Hyperresolution*. Sie wurde ebenfalls von Robinson [Rob65a] eingeführt. Das folgende Schema zeigt die zugrundeliegende Regel. Dabei entsprechen die schraffierten Teile Teilklauseln mit nur positiven Literalen.



Eine Klausel mit mindestens einem negativen Literal dient hier als der sogenante *Nukleus*. Für jedes negative Literal des Nukleus wird ein sogenanntes *Elektron* benötigt. Ein Elektron ist eine Klausel, die nur positive Literale enthält. Der Nukleus wird dann mit allen Elektronen gleichzeitig resolviert, was zu einer Klausel, die nur aus positiven Literalen besteht, führt. Diese kann in einem nächsten Hyperresolutionsschritt als Elektron verwendet werden. Dual zu der hier beschriebenen positiven Hyperresolution kann auch die *Negative Hyperresolution* definiert werden durch Ersetzen der positiven Literale durch negative und umgekehrt. Wird die Positive Hyperresolution dahingehend eingeschränkt, daß die Elektronen nur aus einzelnen positiven Literalen bestehen, so wird von der Positiven Unit Hyperresolution [BY96] gesprochen.

Die folgende Abbildung zeigt ein Beispiel für einen Positiven Hyperresolutionsschritt.

$$\begin{array}{llllllll} q(a,c) & & & & & & & \\ & & p(u,v) & \vee & r(u,v) & & & \\ & & & & & & & \sigma = \{a/x, c/z, a/u, c/v\} \\ \neg q(x,z) & \vee & \neg p(x,z) & \vee & p(x,y) & \vee & p(y,z) & \\ \hline & & r(a,c) & \vee & p(a,y) & \vee & p(y,c) & \end{array}$$

2.6 Festlegungen und Notationen

Nachdem die für diese Arbeit wesentlichen Definitionen eingeführt wurden, werden jetzt noch einige Festlegungen getroffen, die im folgenden gelten sollen.

Die Signatur Σ bestimmt eine Sprache der Prädikatenlogik der ersten Stufe. Für diese Arbeit wird, soweit nicht anders vermerkt, immer eine Signatur *ohne* Funktionssymbole und mit abzählbar unendlich vielen Konstanten zugrundegelegt. Im Hinblick auf die Generierung endlicher Modelle ist der Ausschluß von Funktionssymbolen zunächst sinnvoll. In den nächsten Kapiteln wird dies ausführlich erläutert.

Folgende Notationen und Abkürzungen werden verwendet.

Notation 2.48

- Formeln der Prädikatenlogik der ersten Stufe werden mit $F, G, \ldots$ sowie mit $\varphi, \psi, \ldots$ (mit und ohne Index) bezeichnet.
- Eine endliche Sequenz von Variablen $x_1, \ldots, x_n$ oder Konstanten $c_1, \ldots, c_n$ wird auch kürzer durch $\bar{x}$ bzw. $\bar{c}$ bezeichnet, falls die Länge der Sequenz irrelevant oder aus dem Kontext leicht ersichtlich ist.
- Für ein Atom $p(x_1, \ldots, x_n)$ bzw. $p(c_1, \ldots, c_n)$ wird dann auch kürzer $p(\bar{x})$ bzw. $p(\bar{c})$ geschrieben.
- Für eine Substitution $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ steht abkürzend auch $\sigma = \{\bar{t}/\bar{x}\}$. Für eine Formel F wird dann statt $F\{t_1/x_1, \ldots, t_n/x_n\}$ auch abkürzend $F\{\bar{t}/\bar{x}\}$ geschrieben.
- Statt $\forall x_1 \ldots \forall x_n F$ wird abkürzend $\forall x_1 \ldots x_n F$ oder $\forall \bar{x} F$ benutzt. Hier wird für $\bar{x}$ auch die *leere* Sequenz ϵ zugelassen. Die Formel $\forall \epsilon F$ steht dann für die Formel F.
- Die Bezeichnungen $\forall x_1 \ldots \forall x_n F(x_1, \ldots, x_n)$ und $\forall x_1 \ldots \forall x_n F(\bar{x})$ stehen teilweise anstelle von $\forall x_1 \ldots \forall x_n F$, um zu verdeutlichen, daß die Variablen $x_1, \ldots, x_n$ bzw. $\bar{x}$ in der Formel F auftreten.

Kapitel 3

Die PUHR-Tableaux-Methode

In diesem Kapitel wird die Positive-Unit-Hyperresolution-Tableaux-Methode (kurz: PUHR-Tableaux-Methode) [BY96] vorgestellt. Die PUHR-Tableaux-Methode wurde zur Formalisierung des Theorembeweisers SATCHMO [MB88] eingeführt und ist die Basis der Extended-Positive-Tableaux-Methode, die in Kapitel 4 vorgestellt wird.

Der erste Abschnitt dieses Kapitels ist der Syntax für die PUHR-Tableaux-Methode gewidmet. Abschitt 3.2 beinhaltet dann die PUHR-Tableaux-Methode. Die wichtigsten Ergebnisse werden genannt und die zugehörigen Beweise werden teilweise zitiert [BY96] (Lemma 3.13, Satz 3.15, Satz 3.18, Korollar 3.20) teilweise in überarbeiteter Form gegeben (Satz 3.4, Lemma 3.14, Lemma 3.17, Satz 3.19). Im dritten Abschnitt wird die Implementierung SATCHMO [MB88] vorgestellt. Den Abschluß dieses Kapitels bildet ein Abschnitt über Erweiterungen der PUHR-Tableaux-Methode. Hier wird es um die Generierung ausschließlich minimaler Modelle gehen.

Für das folgende Kapitel wird eine Sprache angenommen, die keine Funktionssymbole enthält.

3.1 Bereichsbeschränkte Klauseln

Die PUHR-Tableaux-Methode ist für *bereichsbeschränkte* Klauseln definiert, wie sie im folgenden eingeführt werden. Es wird sich zeigen, daß die Einschränkung auf bereichsbeschränkte Klauseln nicht schwerwiegend ist, da eine Transformation angegeben werden kann, die jede Klauselmenge in eine Menge bereichsbeschränkter Klauseln überführt und in gewisser Weise modellerhaltend ist.

Definition 3.1 (Bereichsbeschränkte Klausel) Eine Klausel heißt *bereichsbeschränkt*, falls jede Variable, die in einem positiven Literal auftritt, auch in einem negativen Literal auftritt. Ist eine Klausel in Implikationsform, so bedeutet dies, daß jede Variable, die im Kopf der Klausel auftritt, auch in ihrem Rumpf auftritt. □

Beispiel 3.2 Die Klausel $K_1 = \forall x \forall y\ (p(x) \rightarrow q(x,y))$ ist nicht bereichsbeschränkt, da die Variable y in $p(x)$ nicht auftritt. Dagegen ist $K_2 = \forall x \forall y\ (p(x) \wedge r(y) \rightarrow q(x,y))$ bereichsbeschränkt. Sowohl x als auch y kommen im Rumpf von K_2 vor.

Die für diese Arbeit wesentliche Eigenschaft bereichsbeschränkter Klauseln ist die folgende: Eine bereichsbeschränkte Klausel enthält genau dann keine Variablen, wenn ihr Rumpf keine Variablen enthält. Warum diese Eigenschaft von zentraler Bedeutung für die PUHR-Tableaux Methode ist, werden wir im nächsten Abschnitt sehen.

Im folgenden wird eine Transformation angegeben, um aus einer Menge von Klauseln $\mathcal{S}$ eine Menge $BB(\mathcal{S})$ bereichsbeschränkter Klauseln zu erhalten. Diese Transformation ist in einem unten beschriebenen Sinne modellerhaltend.

Definition 3.3 (Transformation in bereichsbeschränkte Klauseln) Sei Σ die Signatur der betrachteten Sprache der ersten Stufe, D ein einstelliges Relationssymbol, das nicht in Σ enthalten ist und Σ' die Vereinigung von Σ und $\{D\}$.

Für jede Klausel $C = A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m$ sei $BB(C)$ folgende Σ'-Klausel:

$$BB(C) := \begin{cases} C, \text{ falls } C \text{ bereichsbeschränkt ist} \\ D(x_1) \wedge \ldots \wedge D(x_l) \wedge A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m, \text{ sonst} \end{cases}$$

wobei $x_1, \ldots, x_l$ die Variablen sind, die in $B_1, \ldots, B_m$ vorkommen, aber nicht in $A_1, \ldots, A_n$.

Sei $\mathcal{S}$ eine Menge von Σ-Klauseln. Dann sei

$$BB(\mathcal{S}) := \{BB(C) \mid C \in \mathcal{S}\} \cup K(\mathcal{S})$$

die zu $\mathcal{S}$ gehörende Menge bereichsbeschränkter Klauseln, wobei

$$K(\mathcal{S}) := \begin{cases} \{\top \rightarrow D(c) \mid c \text{ tritt in } \mathcal{S} \text{ auf}\}, & \text{falls es eine Konstante gibt, die in } \mathcal{S} \text{ auftritt} \\ \{\top \rightarrow D(a) \mid \text{für eine Konstante } a \in \Sigma\} & \text{sonst.} \end{cases}$$

□

Nach Definition 3.3 sind die Klauseln in $BB(\mathcal{S})$ bereichsbeschränkt. Weiter ist für eine endliche Menge $\mathcal{S}$ von Klauseln auch die zugehörige Menge der bereichsbeschränkten Klauseln $BB(\mathcal{S})$ endlich. Wie bereits erwähnt erhält die in Definition 3.3 definierte Transformation auch die Modelle in einem gewissen Sinne. Streng genommen kann nicht von Modellerhaltung gesprochen werden, da die zugrundeliegende Sprache Σ zu Σ' erweitert wird. Wird dieses zusätzliche Relationssymbol aber geschickt interpretiert, etwa durch die ganze Domäne einer Interpretation, so unterscheiden sich die Modelle von der gegebenen Klauselmenge $\mathcal{S}$ und von der zugehörigen Menge bereichsbeschränkter Klauseln $BB(\mathcal{S})$ nicht wesentlich. Der folgende Satz 3.4 zeigt, wie hier in diesem Sinne „Modellerhaltung“ verstanden wird.

Satz 3.4 *Sei Σ die Signatur der betrachteten Sprache der ersten Stufe, D ein einstelliges Relationssymbol, das nicht in Σ enthalten ist und Σ' die Vereinigung von Σ und $\{D\}$. Dann existiert für jede endliche Menge $\mathcal{S}$ von Σ-Klauseln eine endliche Menge bereichsbeschränkter Σ'-Klauseln $BB(\mathcal{S})$, so daß gilt:*

1. *Ist* $(\mathcal{D}, m)$ *ein* Σ*-Modell von* $\mathcal{S}$ *mit der Domäne* $\mathcal{D}$ *und der Zuweisungsfunktion* m*, und ist* m' *folgende über* Σ' *definierte Abbildung:*

$$m'(s) := \begin{cases} m(s), & falls \quad s \neq D \\ \mathcal{D}, & falls \quad s = D \end{cases}$$

dann ist $(\mathcal{D}, m')$ *ein* Σ'*-Modell von* $BB(\mathcal{S})$*.*

2. *Ist* $(\mathcal{D}', m')$ *ein* Σ'*-Modell von* $BB(\mathcal{S})$*, dann existiert ein* Σ*-Redukt* $(\mathcal{D}, m)$ *von* $(\mathcal{D}', m')$*, so daß* $(\mathcal{D}, m)$ *ein* Σ*-Modell von* $\mathcal{S}$ *ist.*

Beweis: Sei $BB(\mathcal{S})$ die zu $\mathcal{S}$ gehörende Menge bereichsbeschränkter Klauseln wie sie in Definition 3.3 definiert wurde.

Zu 1.: Nach Voraussetzung ist $(\mathcal{D}, m)$ ein Σ-Modell von $\mathcal{S}$. Da nach Definition $m'(D) = \mathcal{D}$, ist $(\mathcal{D}, m')$ ein Σ'-Modell von $BB(\mathcal{S})$.

Zu 2.: Zunächst wird ein Σ-Redukt $(\mathcal{D}, m)$ von $(\mathcal{D}', m')$ durch Festlegung der Domäne $\mathcal{D} := m'(D)$ definiert. Insbesondere ist $\mathcal{D} \subseteq \mathcal{D}'$. Da nach Voraussetzung $(\mathcal{D}', m')$ ein Modell von $BB(\mathcal{S})$, also insbesondere von $K(\mathcal{S})$ ist, ist die Domäne $\mathcal{D}$ nicht leer. Für die Definition der Zuweisungsfunktion m des Σ-Redukts $(\mathcal{D}, m)$ können wir also ein Element a aus $\mathcal{D}$ wählen, um die Zuweisungsfunktion m auch auf denjenigen Konstanten $c \in \Sigma$ zu definieren, für die $m'(c) \notin \mathcal{D}$ (siehe Definition 2.26). Nach Lemma 2.27 ist $(\mathcal{D}, m)$ eine Σ-Interpretation. Es bleibt also noch zu zeigen, daß $(\mathcal{D}, m)$ ein Σ-Modell von $\mathcal{S}$ ist.

Sei $D(y_1) \wedge \cdots \wedge D(y_m) \wedge A_1 \wedge \ldots \wedge A_k \rightarrow B_1 \wedge \ldots \wedge B_l$ eine Klausel aus $BB(\mathcal{S})$ und seien $x_1, \ldots, x_n$ alle Variablen in $A_1, \ldots, A_k$ mit $m, n, k, l \geq 0$. Wir betrachten nun die Teilmenge $\mathcal{D}$ von $\mathcal{D}'$, durch die die Domäne des Σ-Redukts $(\mathcal{D}, m)$ definiert ist.

Fall 1: Für alle Tupel $\bar{d} \in \mathcal{D}^n$ ist $(\mathcal{D}', m')\frac{\bar{d}}{\bar{x}}$ kein Modell von $(A_1 \wedge \ldots \wedge A_k)$. Da $(\mathcal{D}', m')$ nach Voraussetzung ein Modell von $K(\mathcal{S})$ ist, gilt für alle Konstanten c, die in $\mathcal{S}$ auftreten: $m'(c) = m(c)$. Da m und m' auch für alle Relationssymbole auf $\mathcal{D}$ übereinstimmen, ist auch $(\mathcal{D}, m)\frac{\bar{d}}{\bar{x}}$ kein Modell von $(A_1 \wedge \ldots \wedge A_k)$. Also ist $(\mathcal{D}, m)$ ein Modell von $A_1 \wedge \ldots \wedge A_k \rightarrow B_1 \wedge \ldots \wedge B_l$.

Fall 2: Es existiert $(d_1, \ldots, d_n) \in \mathcal{D}^n$, so daß $(\mathcal{D}', m')\frac{d_1 \ldots d_n}{x_1 \ldots x_n}$ ein Modell von $(A_1 \wedge \ldots \wedge A_k)$ ist. Da $(\mathcal{D}', m')$ nach Voraussetzung ein Modell von $K(\mathcal{S})$ ist, gilt für alle Konstanten c, die in $\mathcal{S}$ auftreten: $m'(c) = m(c)$. Da m und m' auch für alle Relationssymbole auf $\mathcal{D}$ übereinstimmen, ist $(\mathcal{D}, m)\frac{d_1 \ldots d_n}{x_1 \ldots x_n}$ ein Modell von $(A_1 \wedge \ldots \wedge A_k)$.

Nach Definition 3.3 ist die Relation $m(D)$ nicht leer. Also existiert mindestens ein Tupel $(d_{n+1}, \ldots, d_{n+m}) \in (m'(D))^m$, d.h. in $\mathcal{D}^m$ derart, daß $(\mathcal{D}', m')\frac{d_{n+1} \ldots d_{n+m}}{y_1 \ldots y_m}$ ein Modell von $D(y_1) \wedge \ldots \wedge D(y_m)$ ist. Nach Definition 3.3 sind die Mengen $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_m\}$ disjunkt. Also ist $(\mathcal{D}', m')\frac{d_1 \ldots d_{n+m}}{x_1 \ldots y_m}$ ein Modell von $(D(y_1) \wedge \cdots \wedge D(y_m) \wedge A_1 \wedge \ldots \wedge A_k)$ und wegen $(\mathcal{D}', m')\frac{d_1 \ldots d_{n+m}}{x_1 \ldots y_m} \models BB(C)$ auch von $(B_1 \wedge \ldots \wedge B_l)$. Nach Definition von $BB(C)$ kommen außer $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_m\}$ keine Variablen in $B_1 \wedge \ldots \wedge B_l$ vor. Weiter

stimmen m und m' für alle Konstanten c aus $\mathcal{S}$ und für alle Relationssymbole über $\mathcal{D}$ überein. Also folgt nun, daß auch $(\mathcal{D}, m)\frac{d_1 \ldots d_{n+m}}{x_1 \ldots y_m}$ ein Modell von $(B_1 \wedge \ldots \wedge B_l)$ ist. Also ist $(\mathcal{D}, m)$ ein Modell von $A_1 \wedge \ldots \wedge A_k \rightarrow B_1 \wedge \ldots \wedge B_l$. ∎

Nachdem nun festgelegt wurde, welche Art von Formeln die PUHR-Tableaux Methode behandelt und wir gesehen haben, daß die Forderung nach Bereichsbeschränkung keine wirkliche Einschränkung ist, stellt sich nun die Frage nach dem Zusammenhang zwischen Erfüllbarkeit und Terminterpretationen für bereichsbeschränkte Klauseln in Implikationsform. Das folgende Lemma gibt darüber Auskunft.

Lemma 3.5 *Sei $\mathcal{S}$ eine Menge von bereichsbeschränkten Klauseln in Implikationsform. Dann ist $\mathcal{S}$ erfüllbar genau dann, wenn $\mathcal{S}$ ein Termmodell besitzt.*

Beweis: Da bereichsbeschränkte Klauseln in Implikationsform gleichheitsfreie universelle Formeln ohne freie Variablen sind, hat $\mathcal{S}$ nach Satz 2.31 (siehe auch [EFT92]) ein Herbrandmodell $\mathcal{H}(\mathcal{M})$. Werden nun alle Elemente (in unserem Fall nur Konstanten) des Herbranduniversums, die nicht in $\mathcal{M}$ auftreten, durch ein einziges Element, das nicht in $\mathcal{M}$ auftritt, identifiziert, so erhalten wir eine Terminterpretation $\mathcal{T}(\mathcal{M})$. $\mathcal{T}(\mathcal{M})$ ist ein Termmodell von $\mathcal{S}$, da $\mathcal{H}(\mathcal{M})$ ein Herbrandmodell von $\mathcal{S}$ ist und die Interpretation der Symbole für Herbrandinterpretationen und Terminterpretationen gleich definiert ist. Da die Klauseln in $\mathcal{S}$ bereichsbeschränkt sind, ist es unproblematisch, Terminterpretationen zu betrachten. ∎

3.2 Positive-Unit-Hyperresolution-Tableaux

In diesem Abschnitt wird nun die PUHR-Tableaux-Methode eingeführt. Zunächst werden die PUHR-Expansionsregeln und anschließend die PUHR-Tableaux definiert.

Die PUHR-Tableaux-Methode expandiert Bäume bzw. PUHR-Tableaux gemäß den PUHR-Tableaux-Expansionsregeln für eine Menge $\mathcal{S}$ von bereichsbeschränkten Klauseln in Implikationsform. Ausgangspunkt ist die Menge $\{\top\}$. Jeder Knoten eines PUHR-Tableau besteht aus einer Menge von Grundatomen oder Disjunktionen von Grundatomen, wobei, wie in Kapitel 2 erläutert, $\top$ als eine Konjunktion und $\bot$ als eine Disjunktion von null Atomen aufgefaßt wird. Für eine Menge M von Formeln wird im folgenden mit $Atoms(M)$ die Menge der Grundatome in M bezeichnet.

Definition 3.6 (PUHR-Tableaux-Expansionsregeln) Sei $\mathcal{S}$ eine Menge von bereichsbeschränkten Klauseln in Implikationsform.

Positive-Unit-Hyperresolutions- (PUHR-) Regel	$\vee$-Regel
$\dfrac{R_1 \wedge \ldots \wedge R_n \rightarrow E}{E\sigma}$	$\dfrac{E_1 \vee E_2}{E_1 \mid E_2}$

wobei $\sigma = \{\bar{c}/\bar{x}\}$ eine Grundsubstitution ist, $\bar{x}$ alle freien Variablen in $R_1 \wedge \cdots \wedge R_n$ sind und $(R_1 \wedge \cdots \wedge R_n)\sigma$ durch die Terminterpretation erfüllt wird, die durch die Menge der Grundatome, die im expandierten Knoten vorkommen, bestimmt wird. □

Definition 3.7 (PUHR-Tableaux) PUHR-Tableaux für eine Menge $\mathcal{S}$ von bereichsbeschränkten Klauseln in Implikationsform sind Bäume, deren Knoten Mengen von Grundatomen und Disjunktionen von Grundatomen sind. PUHR-Tableaux werden folgendermaßen induktiv definiert:

1. Der Baum, der nur aus dem einzigen Knoten $\{\top\}$ besteht, ist ein PUHR-Tableau für $\mathcal{S}$.

2. Sei T ein PUHR-Tableau für $\mathcal{S}$ und L ein Blatt von T.

   Ist $K = R_1 \wedge \ldots \wedge R_n \rightarrow E$ eine Klausel aus $\mathcal{S}$, die nicht von der Terminterpretation $\mathcal{T}(Atoms(L))$ erfüllt wird und wird eine Formel $E\sigma$ durch Anwendung der PUHR-Regel auf K erzeugt, so ist der Baum, der durch das Anhängen eines Kinderknotens $L \cup \{E\sigma\}$ an L entsteht, wieder ein PUHR-Tableau für $\mathcal{S}$.

   Ist $E = E_1 \vee E_2$ eine Formel in L, die nicht von der Terminterpretation $\mathcal{T}(Atoms(L))$ erfüllt wird und werden die Formeln E_1 und E_2 durch eine Anwendung der $\vee$-Regel auf E erzeugt, so ist der Baum, der durch das Anhängen von zwei Kinderknoten $L \cup \{E_1\}$ und $L \cup \{E_2\}$ an L entsteht, wieder ein PUHR-Tableau für $\mathcal{S}$.

Ein Zweig eines PUHR-Tableau heißt *geschlossen*, falls er $\bot$ enthält. Sonst heißt er *offen*. Ein PUHR-Tableau ist *offen*, falls es mindestens einen offenen Zweig enthält. Sonst ist es *geschlossen*. Ist $\mathcal{B}$ ein Zweig eines PUHR-Tableau, so bezeichne $\cup\mathcal{B}$ die Vereinigung aller Knoten in $\mathcal{B}$. Ein PUHR-Tableau ist *erfüllbar*, falls es einen Zweig $\mathcal{B}$ besitzt, so daß die Vereinigung von $\mathcal{S}$ und $\cup\mathcal{B}$ erfüllbar ist. □

Zur Veranschaulichung der Definition der PUHR-Tableaux betrachten wir zunächst einmal ein Beispiel. Gemäß Definition 3.7 enthält jeder Knoten eines PUHR-Tableau alle Formeln, die bereits im Elternknoten enthalten waren und zusätzlich die durch Anwendung einer Expansionsregel entstandene Formel. Damit ist also jeder Knoten eine Obermenge seines Elternknotens, was bald zu unübersichtlichen Bäumen führt. Deshalb werden bei der Darstellung der PUHR-Tableaux, ausschließlich die neu hinzugekommenen Formeln im Baum dargestellt. Jeder Knoten des Baumes besteht dann aus der sichtbaren Formel und der Vereinigung aller Formeln in seinen Vorgängerknoten.

Beispiel 3.8 $\mathcal{S}$ ist $\{\top \rightarrow p(a),\ p(x) \rightarrow q(x) \vee r(x),\ q(x) \rightarrow \bot,\ p(x) \wedge r(x) \rightarrow s(x)\}$. Abbildung 3.1 zeigt ein PUHR-Tableau für $\mathcal{S}$. Der Wurzelknoten des PUHR-Tableau besteht aus $\{\top\}$. Jetzt wird die PUHR-Regel auf $\top \rightarrow p(a)$ angewendet und es wird $p(a)$ als neue Formel hinzugefügt. In diesem Knoten müßte streng nach Definition 3.7 sowohl $\top$ als auch $p(a)$ stehen. Zugunsten der Lesbarkeit beschränken wir uns (wie oben bereits erwähnt) auf die neu hinzugekommene Formel $p(a)$. Im nächsten Schritt wird die PUHR-Regel auf $p(x) \rightarrow q(x) \vee r(x)$ angewendet, was zu $q(a) \vee r(a)$ führt. Als nächstes wird die

$\vee$-Regel auf $q(a) \vee r(a)$ angewendet. Dies führt zu einer Verzweigung des PUHR-Tableau. Auf beiden Seiten wird nun wieder die PUHR-Regel angewendet und zwar auf der linken Seite auf $q(x) \rightarrow \bot$ und auf der rechten Seite auf $p(x) \wedge r(x) \rightarrow s(x)$. Der linke Zweig des PUHR-Tableau für $\mathcal{S}$ ist geschlossen, da er $\bot$ enthält.

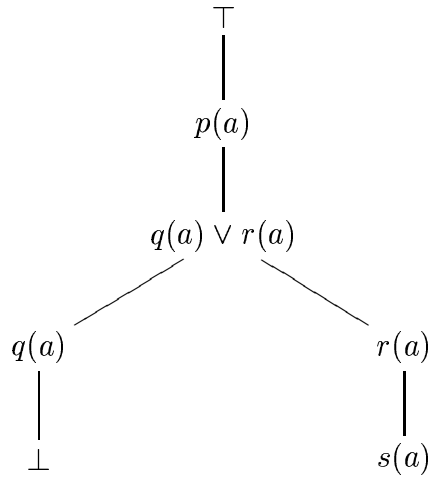


Abbildung 3.1: Ein PUHR-Tableau für Beispiel 3.8

Nach der Definition der PUHR-Tableaux-Methode sei im folgenden auf einige Besonderheiten hingewiesen.

Bemerkung 3.9 Im Gegensatz zu klassischen Tableauverfahren enthalten PUHR-Tableaux nicht sämtliche beteiligten Formeln sondern beziehen sich auf eine Formelmenge $\mathcal{S}$ außerhalb des PUHR-Tableau. Klassischen Ansätzen hätte eher entsprochen, die Wurzel eines PUHR-Tableau als $\mathcal{S}$ zu definieren und nicht als $\{\top\}$. Die Trennung zwischen den gegebenen Formeln in $\mathcal{S}$ und den mit Hilfe der Expansionsregeln erzeugten Formeln im Tableau ist eine Besonderheit der PUHR-Tableaux-Methode. Sie läßt sich folgendermaßen erklären.

Betrachten wir noch einmal die Expansionsregeln. Es zeigt sich, daß sich die gegebenen Klauseln und alle durch Anwendung einer Expansionsregel erzeugten Formeln syntaktisch wesentlich unterscheiden. Sowohl bei der Anwendung der PUHR-Regel als auch bei der Anwendung der $\vee$-Regel werden ausschließlich Atome oder Disjunktionen von Atomen erzeugt. Das rührt daher, daß die PUHR-Tableaux-Methode Klauseln in Implikationsform behandelt. Die Köpfe der Klauseln sind immer Atome oder Disjunktionen von Atomen. Also können keine anderen Formeln erzeugt werden. Gleichzeitig aber kann die PUHR-Regel nur auf Klauseln und die $\vee$-Regel nur auf Disjunktionen angewendet werden. Das bringt mit sich, daß die PUHR-Regel ausschließlich auf Formeln aus der gegebenen Klauselmenge angewendet werden kann und niemals auf eine Formel, die durch Anwendung einer Expansionsregel erzeugt wurde (In Definition 3.7 wurde diese Eigenschaft bereits berücksichtigt.) während die $\vee$-Regel nur auf Formeln, die durch Anwendung einer Expansionsregel erzeugt wurden, angewandt werden kann. Diese Eigenschaft ist eine Besonderheit der PUHR-Tableauxmethode und (wie wir später sehen werden) z.B. in der

EP-Tableaux-Methode nicht mehr gegeben. Aufgrund dieser Überlegungen bietet es sich nun an, die gegebene Klauselmenge und das zugehörige PUHR-Tableau zu trennen, insbesondere mit der Menge $\{\top\}$ zu starten anstatt etwa mit der gegebenen Klauselmenge. Um diese Trennung durchführen zu können, müssen aber alle gegebenen Formeln als Klauseln in Implikationsform gegeben sein, insbesondere auch die Grundatome wie z.B. $\top \rightarrow p(a)$ statt $p(a)$. Wären nämlich die Grundatome als solche gegeben, so müßte wieder unterschieden werden zwischen Grundatomen, die von vornherein gegeben sind und solchen, die durch die Anwendung einer Expansionsregel erzeugt wurden.

Eine zweite Besonderheit der PUHR-Tableaux-Methode ist die folgende: Um zu entscheiden, ob die PUHR-Regel auf eine Klausel $R_1 \wedge \cdots \wedge R_n \rightarrow E$ angewendet werden kann, wird getestet, ob die Klausel von der Terminterpretation $\mathcal{T}(Atoms(L))$, die durch die Grundatome des betreffenden Blatts L repräsentiert wird, ein Modell der Klausel ist. Der Test besteht also darin, eine rein semantische Eigenschaft zu überprüfen. Aufgrund der einfachen syntaktischen Gestalt einer Klausel in Implikationsform und der Definition einer Terminterpretation reduziert sich dieser Test einer semantischen Eigenschaft auf einfaches "matching" (das auch „Halbunifikation“ oder „Angleich “ genannt wird) und kann leicht durch syntaktische Begriffe formalisiert werden. Dies erlaubt, wie wir sehen werden, eine alternative Definition der PUHR-Tableaux-Methode, die auf syntaktischen Begriffen beruht.

Bevor die auf syntaktischen Begriffen basierende Definition der PUHR-Tableaux-Methode vorgestellt wird, sei noch auf eine letzte Eigenschaft der PUHR-Tableaux hingewiesen. Wie schon oben erwähnt, bestehen die Knoten eines PUHR-Tableau ausschließlich aus Grundatomen oder Disjunktionen von Grundatomen, d.h. alle durch Anwendung einer Expansionsregel erzeugten Formeln besitzen keine Variablen. Der Grund dafür liegt in der Bereichsbeschränkung der gegebenen Klauseln. Wird die PUHR-Regel auf eine Klausel K angewendet, so gibt es eine Grundsubstitution σ, so daß die Terminterpretation $\mathcal{T}(Atoms(L))$, die durch die Grundatome des betreffenden Blatts L repräsentiert wird, ein Modell von $(R_1 \wedge \cdots \wedge R_n)\sigma$ ist aber nicht von $E\sigma$. Durch die Grundsubstitution σ werden alle Variablen in $R_1 \wedge \cdots \wedge R_n$ mit Konstanten instantiiert. Da die Klausel K aber bereichsbeschränkt ist, d.h. alle im Kopf der Klausel auftretenden Variablen treten auch in deren Rumpf auf, werden durch σ auch alle Variablen in E instantiiert und damit ist die erzeugte Formel $E\sigma$ ein Grundatom oder eine Disjunktion von Grundatomen. Wie wir später sehen werden ist diese Eigenschaft der PUHR-Tableaux-Methode im Hinblick auf ihre Verwendung als Modellgenerierungsverfahren, wie es in Abschnitt 3.3 vorgestellt wird, von zentraler Bedeutung.

Im folgenden wird jetzt die alternative Definition der PUHR-Tableaux-Methode mittels syntaktischer Begriffe vorgestellt. Dieser Weg wurde in [BY96] gewählt. Dafür wird der Begriff der *Positive-Unit-Hyperresolution* eingeführt, der den PUHR-Tableaux zu ihrem Namen verholfen hat. Die Methode der Hyperresolution wurde zum ersten Mal in [Rob65a] vorgestellt.

Definition 3.10 (Positive-Unit-Hyperresolution) Sei $K = R_1 \wedge \ldots \wedge R_n \rightarrow K_1 \vee \ldots \vee K_m$ eine Klausel in Implikationsform und seien $B_1, \ldots, B_n$ (nicht notwendigerweise

verschiedene) Grundatome. Ist σ eine Substitution mit $R_i\sigma = B_i$ für $1 \leq i \leq n$, dann ist $(K_1 \vee \ldots \vee K_m)\sigma$ eine *Positive-Unit-Hyperresolvente* von K und $B_1, \ldots, B_n$. □

Auf diesem Begriff der Positive-Unit-Hyperresolvente aufbauend wird im folgenden eine alternative PUHR-Regel definiert, die die PUHR-Regel aus Definition 3.6 ersetzen kann.

Definition 3.11 (Alternative PUHR-Regel) Sei $R_1 \wedge \ldots \wedge R_n \rightarrow E$ eine bereichsbeschränkte Klausel in Implikationsform aus $\mathcal{S}$ und seien $B_1, \ldots, B_n$ Grundatome.

Alternative PUHR-Regel

$$\frac{\begin{array}{c} B_1 \\ \vdots \\ B_n \end{array}}{E\sigma}$$

wobei $E\sigma$ die Positive-Unit-Hyperresolvente von einer Klausel $K = R_1 \wedge \ldots \wedge R_n \rightarrow E$ aus $\mathcal{S}$ und $B_1, \ldots, B_n$ ist. □

Definition 3.12 (PUHR-Tableaux, alternative Definition) PUHR-Tableaux werden folgendermaßen induktiv definiert:

1. Der Baum, der nur aus dem einzigen Knoten $\{\top\}$ besteht, ist ein PUHR-Tableau für $\mathcal{S}$.

2. Sei T ein PUHR-Tableau für $\mathcal{S}$ und L ein Blatt von T. Führt die Anwendung der PUHR-Regel (bzw. der $\vee$-Regel) auf Formeln in L auf eine Formel E (bzw. zwei Formeln E_1 und E_2), die nicht von einem Atom in L subsumiert wird, so ist der Baum, der aus T durch das Anhängen eines Knotens $L \cup \{E\}$ (bzw. zwei Knoten $L \cup \{E_1\}$ und $L \cup \{E_2\}$) an L entsteht, wieder ein PUHR-Tableau für $\mathcal{S}$.

□

An dieser Stelle sei noch einmal auf die Unterschiede der beiden Definitionen hingewiesen. Erstens wird in Definition 3.6 bei der PUHR-Regel die Erzeugung der neuen Formel $E\sigma$ mittels des *semantischen* Begriffs der Terminterpretation beschrieben. Erfüllt die durch den betreffenden Knoten gegebene Terminterpretation eine Instanz des Klauselrumpfs, so wird die entsprechende Instanz des Klauselkopfs erzeugt. In der alternativen Definition 3.11 wird dagegen der *syntaktische* Begriff der Positive-Unit-Hyperresolvente verwendet, um zu definieren, wann eine neue Formel durch Anwendung der Expansionsregel erzeugt wird.

Zweitens darf in Definition 3.7 der PUHR-Tableaux eine Expansionsregel auf eine Formel nur dann angewendet werden, wenn die Formel noch nicht von der Terminterpretation erfüllt wird, die durch den betreffenden Knoten gegeben ist. Es muß also überprüft werden,

ob die durch den betreffenden Knoten gegebene Terminterpretation bereits ein Modell der Formel ist oder nicht, d.h. es wird eine *semantische* Eigenschaft getestet. Bei der alternativen Definition 3.12 darf eine Expansionsregel auf eine Formel angewendet werden, falls diese nicht durch ein Atom im Zweig subsumiert wird. Es wird also eine *syntaktische* Eigenschaft getestet. Es sollte aber offensichtlich sein, daß beide Definitionen zu der gleichen Methode führen.

Das folgende Lemma zeigt wieder, daß die Knoten eines PUHR-Tableau ausschließlich aus Grundatomen und Disjunktionen von Grundatomen bestehen. Was wir oben anhand der Modellbeziehung für Terminterpretationen gesehen haben (siehe Bemerkung 3.9) wird jetzt mit dem Begriff der Positiven-Unit-Hyperresolvente formuliert.

Lemma 3.13 *Die Positive-Unit-Hyperresolvente einer bereichsbeschränkten Klausel in Implikationsform und einer Menge von Grundatomen ist ein Grundatom oder eine Disjunktion von Grundatomen.*

Beweis: Aufgrund der Bereichsbeschränkung der gegebenen Klausel enthält eine Positive-Unit-Hyperresolvente von dieser Klausel und einer Menge von Grundatomen keine Variablen mehr. ∎

Im folgenden werden einige Ergebnisse über die PUHR-Tableaux-Methode vorgestellt [BY96]. Hierbei sei $\mathcal{S}$ immer eine endliche Menge bereichsbeschränkter Klauseln in Implikationsform. Die Beweise stützen sich auf Definition 3.6 und 3.7. Das folgende Lemma führt direkt auf die Widerlegungskorrektheit der PUHR- Tableaux-Methode.

Lemma 3.14 *Die Anwendung einer PUHR-Tableaux-Expansionsregel auf ein erfüllbares PUHR-Tableau führt wieder zu einem erfüllbaren PUHR-Tableau.*

Beweis: Sei T ein PUHR-Tableau für eine Menge $\mathcal{S}$ von bereichsbeschränkten Klauseln in Implikationsform, $\mathcal{B}$ ein endlicher Zweig von T, $L_{\mathcal{B}}$ das Blatt von $\mathcal{B}$ und $\mathcal{I}$ ein Modell der Vereinigung von $\mathcal{S}$ und $\cup\mathcal{B}$.

Es wird nun für jede PUHR-Expansionsregel gezeigt, daß ihre Anwendung auf $L_{\mathcal{B}}$ zu mindestens einem Kinderknoten N führt, so daß die Vereinigung von $\mathcal{S}$ mit $\cup\mathcal{B}$ und N wieder erfüllbar ist.

1. PUHR-Regel: Sei $R_1 \wedge \ldots \wedge R_n \rightarrow E$ eine Klausel aus $\mathcal{S}$. Damit die PUHR-Regel angewendet werden kann, muß es eine Grundsubstitution σ geben, derart daß $(R_1 \wedge \ldots \wedge R_n)\sigma$ von der durch den Zweig repräsentierten Terminterpretation $\mathcal{T}(Atoms(\cup\mathcal{B}))$ erfüllt wird und $E\sigma$ nicht. Nach Anwendung der PUHR-Regel besteht der Kinderknoten von $L_{\mathcal{B}}$ aus $\cup\mathcal{B} \cup \{E\sigma\}$. Nach Voraussetzung ist $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$. Also muß gezeigt werden, daß $\mathcal{I}$ auch ein Modell von $E\sigma$ ist. Dazu wird folgende Eigenschaft (*) benutzt:

> Ist $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$ und ist $\mathcal{T}(Atoms(\cup\mathcal{B})$ ein Modell von $(R_1 \wedge \ldots \wedge R_n)\sigma$, so ist $\mathcal{I}$ auch ein Modell von $(R_1 \wedge \ldots \wedge R_n)\sigma$.

Da $\mathcal{I}$ nach Voraussetzung ein Modell von $R_1 \wedge \ldots \wedge R_n \rightarrow E$ ist, ist $\mathcal{I}$ also auch ein Modell von $E\sigma$. Damit folgt die Behauptung.

Beweis von (*) durch Induktion über $n \in \mathbb{N}$: $n = 0$: Die Konjunktion reduziert sich zu der Formel $\top$. Da $\top$ nach Definition von jeder Interpretation erfüllt wird, ist nichts zu zeigen.

$n \rightarrow n+1$: Nach Voraussetzung ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $(R_1 \wedge \ldots \wedge R_{n+1})\sigma$, d.h. $\mathcal{T}(Atoms(\cup\mathcal{B})) \models (R_1 \wedge \ldots \wedge R_n)\sigma$ und $\mathcal{T}(Atoms(\cup\mathcal{B})) \models R_{n+1}\sigma$. Da $R_{n+1}\sigma$ ein Grundatom ist, ist $R_{n+1}\sigma \in Atoms(\cup\mathcal{B})$. Da $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$ ist, ist $\mathcal{I}$ also auch ein Modell von $R_{n+1}\sigma$. Nach Induktionsvoraussetzung ist $\mathcal{I}$ auch ein Modell von $(R_1 \wedge \ldots \wedge R_n)\sigma$ und damit von $(R_1 \wedge \ldots \wedge R_{n+1})\sigma$.
2. $\vee$-Regel: Sei $E = E_1 \vee E_2$ ein Element aus $\cup\mathcal{B}$. Da $\mathcal{I}$ nach Voraussetzung ein Modell von E ist, ist $\mathcal{I}$ ein Modell von E_1 oder von E_2. Damit ist auch das PUHR-Tableau, das durch Anwendung der $\vee$-Regel entsteht, wieder erfüllbar. ∎

Satz 3.15 (Widerlegungskorrektheit) *Gibt es ein geschlossenes PUHR-Tableau für $\mathcal{S}$, so ist $\mathcal{S}$ unerfüllbar.*

Beweis: Angenommen $\mathcal{S}$ sei erfüllbar. Dann ist das PUHR-Tableau für $\mathcal{S}$, das aus dem einzigen Knoten $\{\top\}$ besteht, ebenfalls erfüllbar. Nach Lemma 3.14 gibt es dann aber kein geschlossenes PUHR-Tableau für $\mathcal{S}$. ∎

Im folgenden wird eine Formalisierung des Standardkonzepts von *Fairness* (vgl. [Fit90]) eingeführt. Informell kann ein faires PUHR-Tableau dadurch beschrieben werden, daß jede mögliche Anwendung einer Expansionsregel entlang eines offenen Zweiges nach endlich vielen Schritten stattfindet.

Definition 3.16 (Fairness)
• Sei T ein PUHR-Tableau für $\mathcal{S}$ und $\mathcal{B}$ ein Zweig in T. Dann ist die Vereinigung $\cup\mathcal{B}$ aller Knoten im Zweig $\mathcal{B}$ *saturiert* , falls das folgende gilt:

1. Ist $R_1 \wedge \ldots \wedge R_n \rightarrow E$ eine Klausel aus $\mathcal{S}$, dann ist für alle Grundsubstitutionen σ, für die $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $R_1 \wedge \ldots \wedge R_n\sigma$ ist, auch $E\sigma \in \cup\mathcal{B}$.

2. Ist $E_1 \vee E_2 \in \cup\mathcal{B}$, dann ist $E_1 \in \cup\mathcal{B}$ oder $E_2 \in \cup\mathcal{B}$.

• Ein PUHR-Tableau T heißt *fair*, wenn $\cup\mathcal{B}$ für jeden offenen Zweig $\mathcal{B}$ von T saturiert ist. □

Mit diesem Begriff eines fairen PUHR-Tableau kann jetzt im nächsten Lemma die Modellkorrektheit der PUHR-Tableaux-Methode gezeigt werden, welche zum einen die Basis für die Widerlegungsvollständigkeit der PUHR-Tableaux-Methode ist. Zum anderen wird hier schon deutlich, wie die PUHR-Tableaux-Methode als Modellgenerierungsverfahren verwendet werden kann: Die Grundatome eines offenen Zweiges werden "eingesammelt" und repräsentieren ein Termmodell der gegebenen Klauselmenge. Aufgrund der Bereichsbeschränkung der gegebenen Klauseln (siehe Bemerkung 3.9 und Lemma 3.13) sind alle

Atome oder Disjunktionen von Atomen, die in einem PUHR-Tableau vorkommen, Grundatome oder Disjunktionen von Grundatomen. Damit wird alle positive "Information" im Tableau explizit gemacht, was im Falle von freien Variablen nicht der Fall wäre.

Lemma 3.17 (Modellkorrektheit) *Sei T ein PUHR-Tableau für $\mathcal{S}$ und $\mathcal{B}$ ein offener Zweig in T. Ist T fair, so ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $\mathcal{S}$.*

Beweis: Angenommen $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ist kein Modell von $\mathcal{S}$. Dann gibt es mindestens eine Klausel $K = R_1 \wedge \ldots \wedge R_n \rightarrow E$, so daß $\mathcal{T}(Atoms(\cup\mathcal{B}))$ kein Modell von K ist. Also gibt es eine Grundsubstitution σ, so daß $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $(R_1 \wedge \ldots \wedge R_n)\sigma$ ist, aber nicht von $E\sigma$. Da T fair ist, ist $E\sigma \in \cup\mathcal{B}$. Nach Bemerkung 3.9 bzw. Lemma 3.13 ist $E\sigma$ ein Grundatom oder eine Disjunktion von Grundatomen. Ist $E\sigma$ ein Atom, so ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $E\sigma$, Widerspruch. Ist $E\sigma$ eine Disjunktion von Atomen, so gibt es, da T fair ist, mindesten ein Disjunkt, das in $\cup\mathcal{B}$ enthalten ist. Damit ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $E\sigma$, Widerspruch. ∎

Mit Lemma 3.17 folgt die Widerlegungsvollständigkeit der PUHR-Tableaux-Methode nun direkt.

Satz 3.18 (Widerlegungsvollständigkeit) *Ist $\mathcal{S}$ unerfüllbar, dann ist jedes faire PUHR-Tableau für $\mathcal{S}$ geschlossen.*

Beweis: Sei T ein faires PUHR-Tableau für $\mathcal{S}$ und $\mathcal{B}$ ein offener Zweig in T. Nach Lemma 3.17 ist dann $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $\mathcal{S}$ und $\mathcal{S}$ erfüllbar. ∎

Das Ende dieses Abschnitts bildet der Beweis der Vollständigkeit der PUHR-Tableaux-Methode für minimale Modelle.

Satz 3.19 *Sei T ein PUHR-Tableau für $\mathcal{S}$ und $\mathcal{M}$ eine Menge von Grundatomen. Ist $\mathcal{T}(\mathcal{M})$ ein Modell von $\mathcal{S}$, so gibt es einen Zweig $\mathcal{B}$ in T derart, daß $Atoms(\cup\mathcal{B}) \subseteq \mathcal{M}$.*

Beweis: Da $\mathcal{S}$ erfüllbar ist, ist T nach Satz 3.15 ein offenes Tableau. Sei $\mathcal{M}_\mathcal{B}$ die Menge der offenen Zweige von T. Angenommen, die Behauptung gilt nicht. Dann ist für jeden Zweig in T, insbesondere jeden offenen Zweig $\mathcal{B}$ im Tableau, $Atoms(\cup\mathcal{B}) \not\subseteq \mathcal{M}$. Mit dem Auswahlaxiom kann aus jedem Zweig $\mathcal{B} \in \mathcal{M}_\mathcal{B}$ ein Atom A gewählt werden, welches nicht Element in $\mathcal{M}$ ist. Die Menge dieser Atome $\{A_i \mid i \in \mathbb{N}\}$ ist abzählbar, da ein Atom eine endliche Zeichenreihe über einem höchsten abzählbaren Alphabet ist. Sei nun $\mathcal{S}' := \mathcal{S} \cup \{A_i \rightarrow \bot \mid i \in \mathbb{N}\}$. Da $A_i \notin \mathcal{M}$ für $i \in \mathbb{N}$ ist $\mathcal{T}(\mathcal{M})$ ein Modell von $\mathcal{S}'$, d.h. $\mathcal{S}'$ ist erfüllbar. Durch Anfügen von $\bot$ an alle offenen Zweige in T (Anwendung der PUHR-Regel auf $A_i \rightarrow \bot$, da $A_i \in \cup\mathcal{B}$) entsteht ein PUHR-Tableau T' für $\mathcal{S}'$. T' ist geschlossen, was im Widerspruch zur Erfüllbarkeit von $\mathcal{S}'$ steht. ∎

Korollar 3.20 (Vollständigkeit für minimale Modelle) *Sei T ein faires PUHR-Tableau für $\mathcal{S}$ und $\mathcal{M}$ eine Menge von Grundatomen. Ist $\mathcal{T}(\mathcal{M})$ ein minimales Modell von $\mathcal{S}$, dann gibt es einen Zweig $\mathcal{B}$ in T mit $Atoms(\cup\mathcal{B}) = \mathcal{M}$.*

Beweis: Nach Satz 3.19 gibt es einen Zweig $\mathcal{B}$ in T mit $Atoms(\cup\mathcal{B}) \subseteq \mathcal{M}$. Da T fair ist, ist nach Lemma 3.17 $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $\mathcal{S}$. Da $\mathcal{T}(\mathcal{M})$ aber ein minimales Modell von $\mathcal{S}$ ist, so ist $Atoms(\cup\mathcal{B}) = \mathcal{M}$. ∎

In diesem Abschnitt konnte die Widerlegungsvollständigkeit und -korrektheit, die Modellkorrektheit sowie die Vollständigkeit für minimale Modelle der PUHR-Tableaux-Methode gezeigt werden. Es wird sich zeigen (s. Kapitel 6), daß gerade minimale Modelle in vielen Anwendungen von zentraler Bedeutung sind. In Abschnitt 3.4 werden wir aber sehen, daß mittels der PUHR-Tableaux-Methode zwar jedes minimale Modell einer gegebenen Klauselmenge erzeugt wird, zudem aber sowohl minimale Modelle mehrfach als auch nicht-minimale Modelle erzeugt werden können. Deshalb wird in Abschnitt 3.4 eine Erweiterung der PUHR-Tableaux-Methode vorgestellt, die die Erzeugung ausschließlich minimaler Modelle erzwingt.

Schließlich sei noch erwähnt, daß die vorgestellten Ergebnisse genauso gelten, wenn in der gegebenen Sprache Funktionssymbole zugelassen sind [BY96]. Wir werden später sehen, warum es im Hinblick auf die Generierung endlicher Modelle sinnvoll ist, diese Erweiterung um Funktionssymbole zunächst noch nicht zu betrachten.

3.3 Implementierung in Prolog: SATCHMO

Der folgende Abschnitt ist der Implementierung SATCHMO [MB88] der PUHR-Tableaux-Methode gewidmet. SATCHMO steht für SATisfiability CHecking by MOdel generation und beinhaltet eine Sammlung von Prologprogrammen, die auf verschiedene Klassen von Problemen anwendbar sind. Stellvertretend für die ganze Programmsammlung wird hier dasjenige SATCHMO-Programm vorgestellt, welches faire PUHR-Tableaux mittels Tiefensuche expandiert. Dieses Programm, das im folgenden der Einfachheit halber SATCHMO genannt wird, ist widerlegungsvollständig, da *faire* PUHR-Tableaux expandiert werden.

Eine Besonderheit von SATCHMO ist die gleichzeitige Nutzung von Prolog sowohl für die Repräsentation der Klauselmenge als auch für die Implementierung. Das hat zur Folge, daß für die Implementierung der Tiefensuche die Prolog-eigene Tiefensuche sowie das von Prolog bereitgestellte „matching“ ausgenutzt werden kann. Dies wiederum führt zu einem sehr kurzen Programm.

SATCHMO bearbeitet bereichsbeschränkte Klauseln in Implikationsform. Wie bereits in Abschnitt 3.2 ausführlich beschrieben, erzwingt die Bereichsbeschränkung, daß mit der Instantiierung des Rumpfes einer Klausel auch der Kopf instantiiert ist. Deshalb bestehen die Knoten eines PUHR-Tableau ausschließlich aus Grundatomen bzw. Disjunktionen von Grundatomen. Diese Tatsache wird dahingehend ausgenutzt, daß SATCHMO die in einem Zweig auftretenden Grundatome in der dynamischen Datenbasis speichert. Die dynamische Datenbasis repräsentiert auf diese Weise zu jedem Zeitpunkt eine Terminterpretation. Der Test, ob eine Klausel durch die gegebene Terminterpretation verletzt ist, reduziert sich nun auf einfaches „matching“. Dies wäre nicht gegeben, wenn die Forderung der Be-

reichsbeschränktheit aufgehoben wäre und damit freie Variablen in einem PUHR-Tableau auftreten könnten. Kommen wir nun zurück zum Programm.

Eine Menge von Klauseln in Implikationsform, die von SATCHMO bearbeitet wird, wird folgendermaßen repräsentiert:

1. Gemäß der Prologkonvention werden Relationssymbole und Konstanten durch Bezeichner repräsentiert, die mit Kleinbuchstaben beginnen. Variablen werden durch Bezeichner dargestellt, die mit Großbuchstaben beginnen.

2. Gemäß der Prologkonvention wird eine Konjunktion durch Komma und eine Disjunktion durch Semikolon dargestellt.

3. Um die Implikation zu repräsentieren, wird ein binärer Operator `=>` deklariert.

4. $\top$ wird durch `true` und $\bot$ durch `false` dargestellt.

Beispiel 3.21 Die Klauselmenge $\mathcal{S} = \{\top \to p(a),\ q(a) \to \bot,\ p(x) \wedge q(x) \to r(x) \vee s(x)\}$ wird folgendermaßen repräsentiert:

```
true => p(a).
q(a) => false.
p(X), q(X) => r(X); s(x).
```

Abbildung 3.2 zeigt das SATCHMO Programm. SATCHMO expandiert die Zweige eines fairen PUHR-Tableau mittels Tiefensuche. Den Kern des Programms bildet das Prologprädikat `satisfiable`. Durch Zurücksetzen über `satisfiable` werden Termmodelle der gegebenen Klauselmenge erzeugt, die durch eine Menge $\mathcal{G}$ von Grundatomen in der dynamischen Datenbasis des Prologsystems repräsentiert werden. Die Elemente der Menge $\mathcal{G}$ werden durch Rücksetzen über das Prologprädikat `assume` sowohl in die dynamische Datenbank eingefügt als auch wieder herausgenommen.

Die Fairness wird durch den Aufruf des Prolog-built-in Prädikats `findall` gewährleistet. Das Prologprädikat `violated_instance` gibt bei jedem Aufruf eine Instanz einer Klausel zurück, die von der momentanen Datenbasis nicht erfüllt wird und deshalb noch erfüllt werden muß. Durch das Prologprädikat `findall` werden also alle Klauselinstanzen zurückgegeben, die zu diesem Zeitpunkt nicht erfüllt sind. So wird die Fairness gesichert. Da das Prologprädikat `findall` auch erfolgreich aufgerufen wird, falls es keine verletzten Klauseln gibt, ist es notwendig, durch einen Test sicherzustellen, daß die Menge der verletzten Klauseln nichtleer ist.

Durch den Aufruf des Prologprädikats `satisfy_all` wird nun die Menge der verletzten Klauseln elementweise erfüllt. Erfüllen bedeutet hier, daß mittels des Prologprädikats `satisfy` Grundatome in die Prologdatenbank eingefügt werden, damit die Terminterpretation, die durch die Prologdatenbank repräsentiert wird, die zuvor verletzte Klauselinstanz jetzt erfüllt. Da in einem Schritt eine Menge von Klauselinstanzen erfüllt wird, kann folgendes passieren: Durch das Erfüllen einer Klauselinstanz wird eine zweite miterfüllt. Um

```
satisfiable :-
        findall(K, violated_instance(K), Set),
        not (Set = []),
        !,
        satisfy_all(Set),
        satisfiable.
satisfiable.

violated_instance(B => H) :-
        (B => H),
        B,
        not H.

satisfy_all([]).
satisfy_all([_B => H | T]) :-
        H,
        !,
        satisfy_all(T).
satisfy_all([_B => H | T]) :-
        satisfy(H),
        satisfy_all(T).

satisfy(E) :-
        component(Atom, E),
        not (Atom = false),
        assume(Atom).

component(Atom, (Atom; _Rest)).
component(Atom, (_ ; Rest)) :-
        !,
        component(Atom, Rest).
component(Atom, Atom).

assume(Atom) :-
        asserta(Atom).
assume(Atom) :-
        once(retract(Atom)),
        fail.
```

Abbildung 3.2: Das SATCHMO-Programm

zu verhindern, daß die miterfüllte Klauselinstanz ein zweites Mal erfüllt wird, ist die zweite Prologklausel von `satisfy_all` ein Test, ob die zu erfüllende Klauselinstanz noch verletzt ist.

Um eine Klauselinstanz zu erfüllen, wird der Kopf der verletzten Klauselinstanz an `satisfy` übergeben. Der Kopf einer verletzten Klausel ist entweder ein Grundatom oder eine Disjunktion von Grundatomen. Im Falle eines Atoms liefert das Prologprädikat `component` das Atom selbst zurück. Das Atom wird, falls es nicht `false` ist, mittels des Prologprädikats `assume` in die Datenbank eingeführt. Im Falle einer Disjunktion gibt das Prologprädikat `component` durch Zurücksetzen alle Disjunktionsglieder nacheinander zurück. Dadurch werden durch Zurücksetzen über `satisfy` alle Möglichkeiten, die gegebene Disjunktion zu erfüllen, abgedeckt.

3.4 Erweiterungen der PUHR-Tableaux-Methode

In Abschnitt 3.2 wurde die Vollständigkeit der PUHR-Tableaux-Methode für minimale Termmodelle gezeigt. Für eine gegebene Menge $\mathcal{S}$ von Klauseln und ein gegebenes Termmodell $\mathcal{T}(\mathcal{G})$ befindet sich also in jedem fairen PUHR-Tableau für $\mathcal{S}$ ein Zweig $\mathcal{B}$, so daß die Grundatome in $\mathcal{B}$ genau die Grundatome der Menge $\mathcal{G}$ sind. Der Vollständigkeitssatz macht aber keine Aussage über die anderen Zweige im Tableau. So können verschiedene Zweige des Tableau das gleiche Termmodell repräsentieren (Problem der Duplikate). Genauso können Zweige im Tableau existieren, die nichtminimale Termmodelle darstellen. Am folgenden Beispiel wird diese Problematik veranschaulicht.

Beispiel 3.22 Wir betrachten die Menge von Klauseln $\mathcal{S}_5 = \{\top \rightarrow p(a) \vee p(b),\ \top \rightarrow p(a) \vee p(c),\ p(a) \rightarrow p(b) \vee p(d),\ p(b) \rightarrow p(a) \vee p(d)\}$. Abbildung 3.3 zeigt ein faires PUHR-Tableau für $\mathcal{S}_5$.

Das PUHR-Tableau hat fünf offene Zweige, die nach Lemma 3.17 jeweils ein Termmodell von $\mathcal{S}_5$ repräsentieren. Die Interpretationen $\mathcal{T}(\{p(a), p(b)\})$, $\mathcal{T}(\{p(a), p(d)\})$ und $\mathcal{T}(\{p(b), p(c), p(d)\})$ sind die minimalen Modelle von $\mathcal{S}_5$. Jedes dieser Modelle wird durch einen Zweig im PUHR-Tableau repräsentiert. Das Modell $\mathcal{T}(\{p(a), p(b)\})$ tritt allerdings zweimal auf. Es wird vom ersten (von links gezählt) sowie vom dritten Zweig des PUHR-Tableau repräsentiert. Der vierte Zweig des PUHR-Tableau repräsentiert das nichtminimale Termmodell $\mathcal{T}(\{p(a), p(b), p(c)\})$.

In [BY96, BBE+96] wurde eine Methode, die *minimal model generation procedure* (kurz MM-Prozedur), vorgestellt, um die Generierung von Duplikaten und nichtminimalen Modellen zu verhindern. Die MM-Prozedur besteht aus zwei Teilen, dem *complement-splitting* sowie einer *beschränkten Suche*. Es zeigt sich, daß die MM-Prozedur die Generierung ausschließlich minimaler Modelle erzwingt sowie die Generierung von Duplikaten verhindert.

Das nächste Kapitel dieser Arbeit wird der EP-Tableaux-Methode gewidmet sein. Wie wir später sehen werden, sind für manche Anwendungen der EP-Tableaux-Methode besonders die minimalen Modelle von Interesse (siehe Kapitel 6). Deshalb stellt sich die Frage, ob die MM-Prozedur auch auf die EP-Tableaux-Methode übertragbar ist. Aus diesem

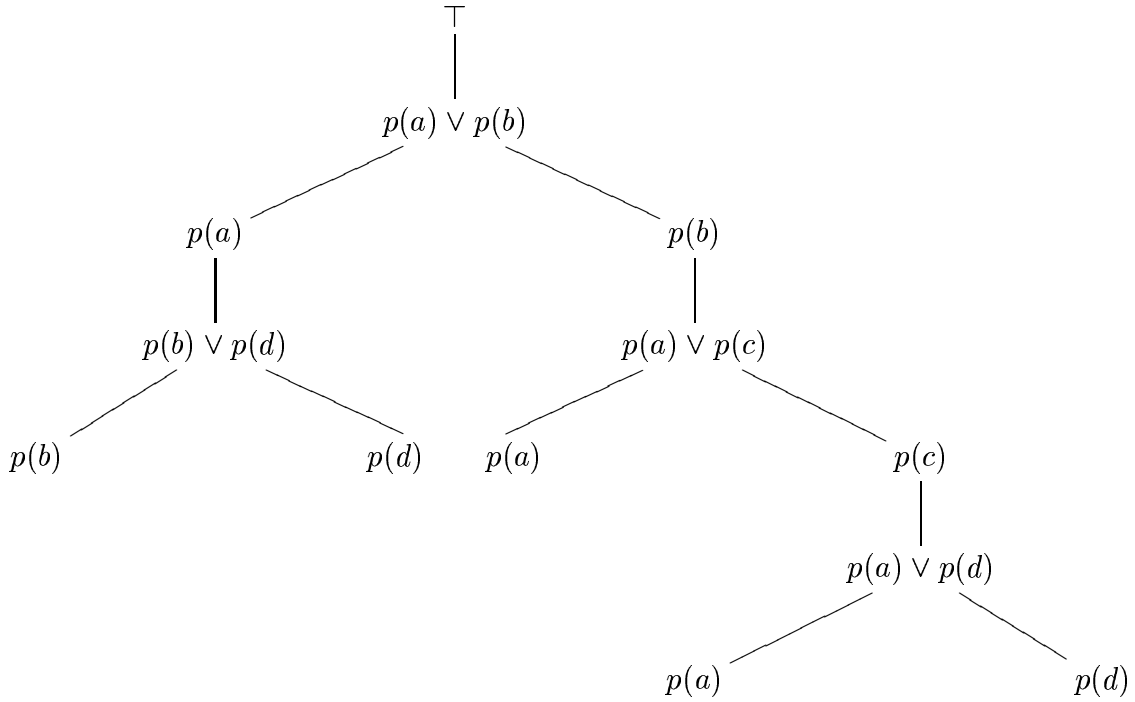


Abbildung 3.3: Ein faires PUHR-Tableau für Beispiel 3.22

Grund wird an dieser Stelle die MM-Prozedur vorgestellt. Es wird teilweise auf rein formale Beschreibungen verzichtet. Statt dessen werden die wesentlichen Eigenschaften der MM-Prozedur informell bzw. anhand von Beispielen erläutert. Die Ergebnisse werden nur zitiert, für die Beweise sei auf [BY96] verwiesen. Im folgenden sei $\mathcal{S}$ immer eine Menge von bereichsbeschränkten Klauseln.

3.4.1 Complement-Splitting

Definition 3.23 (Complement-Splitting-Regel) Für eine Disjunktion von Atomen $D = A_1 \vee \ldots \vee A_n$ bezeichne $Neg(D)$ die endliche Menge von Klauseln in Implikationsform $Neg(D) := \{A_1 \rightarrow \bot, \ldots, A_n \rightarrow \bot\}$.

Complement-Splitting-Regel:

$$\frac{E_1 \vee E_2}{\begin{array}{l|l} E_1 & E_2 \\ Neg(E_2) & \end{array}}$$

□

Nach der Definition der Complement-Splitting-Regel können jetzt die *PUHR-Complement-Tableaux* definiert werden. Sie unterscheiden sich von den PUHR-Tableaux ausschließlich dadurch, daß die $\vee$-Regel durch die Complement-Splitting-Regel ersetzt wird.

Definition 3.24 (PUHR-Complement-Tableaux) PUHR-Complement-Tableaux für eine Menge $\mathcal{S}$ von bereichsbeschränkten Klauseln in Implikationsform sind Bäume, deren Knoten Mengen von Grundatomen und Disjunktionen von Grundatomen sind. PUHR-Complement-Tableaux werden folgendermaßen induktiv definiert:

1. Der Baum, der nur aus dem einzigen Knoten $\{\top\}$ besteht, ist ein PUHR-Complement-Tableau für $\mathcal{S}$.

2. Sei T ein PUHR-Complement-Tableau für $\mathcal{S}$ und L ein Blatt von T. Weiter sei $K = R_1 \wedge \ldots \wedge R_n \rightarrow E$ eine Klausel aus $\mathcal{S}$, die nicht von der Terminterpretation $\mathcal{T}(Atoms(L))$ erfüllt wird und $E\sigma$ eine Formel, die durch Anwendung der PUHR-Regel auf K erzeugt wird. Dann ist der Baum, der durch das Anhängen eines Kinderknotens $L \cup \{E\sigma\}$ an L entsteht, wieder ein PUHR-Complement-Tableau für $\mathcal{S}$.

   Sei $E = E_1 \vee E_2$ eine Formel in L, die nicht von der Terminterpretation $\mathcal{T}(Atoms(L))$ erfüllt wird und seien die Formeln E_1 und E_2 und die Formelmenge $Neg(E_2)$ durch eine Anwendung der Complement-Splitting-Regel auf E erzeugt. Dann ist der Baum, der durch das Anhängen von zwei Kinderknoten $L \cup \{E_1, Neg(E_2)\}$ und $L \cup \{E_2\}$ an L entsteht, wieder ein PUHR-Complement-Tableau für $\mathcal{S}$.

□

Die Begriffe *offenes* und *geschlossenes* PUHR-Complement-Tableau werden genau wie im Falle der PUHR-Tableaux (siehe Definition 3.7) definiert.

Dagegen muß die Definition eines *fairen* PUHR-Complement-Tableau an die neue Complement-Splitting-Regel angepaßt werden.

Definition 3.25 (Fairness)
• Sei T ein PUHR-Complement-Tableau für $\mathcal{S}$ und $\mathcal{B}$ ein Zweig in T. Dann ist die Vereinigung aller Knoten $\cup\mathcal{B}$ im Zweig $\mathcal{B}$ *saturiert* , falls das folgende gilt:

1. Ist $R_1 \wedge \ldots \wedge R_n \rightarrow E$ eine Klausel aus $\mathcal{S}$, dann ist für alle Grundsubstitutionen σ, für die $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $R_1 \wedge \ldots \wedge R_n\sigma$ ist, auch $E\sigma \in \cup\mathcal{B}$.

2. Ist $E_1 \vee E_2 \in \cup\mathcal{B}$, dann ist $\{E_1\} \cup Neg(E_2) \subseteq \cup\mathcal{B}$ oder $E_2 \in \cup\mathcal{B}$.

• Ein PUHR-Complement-Tableau T heißt *fair*, wenn $\cup\mathcal{B}$ für jeden offenen Zweig $\mathcal{B}$ von T saturiert ist. □

Wie der folgende Satz zeigt, wird durch die Erweiterung der PUHR-Tableaux auf die PUHR-Complement-Tableaux die Eigenschaft der Modellvollständigkeit für minimale Modelle nicht angetastet.

Satz 3.26 *Sei T ein faires PUHR-Complement-Tableau für $\mathcal{S}$ und $\mathcal{M}$ eine Menge von Grundatomen. Ist $\mathcal{T}(\mathcal{M})$ ein minimales Modell* [1] *von $\mathcal{S}$, dann gibt es einen Zweig $\mathcal{B}$ in T mit $Atoms(\cup\mathcal{B}) = \mathcal{M}$.*

[1] In [BY96] sind alle in diesem Abschnitt zitierten Ergebnisse für Herbrandinterpretationen bzw. -modelle formuliert. Allerdings können sie, genauso wie alle anderen Ergebnisse des Kapitels 3, auf Terminterpretationen bzw. -modelle übertragen werden.

Bemerkung 3.27 Die Complement-Splitting-Methode kann in SATCHMO eingebaut werden. Auf die Implementierung [BY96] soll an dieser Stelle nicht eingegangen werden. Sie findet sich in Anhang A.

Beispiel 3.28 Wir betrachten noch einmal die Klauselmenge $\mathcal{S}_5$ aus Beispiel 3.22: $\mathcal{S}_5 = \{\top \to p(a) \vee p(b),\ \top \to p(a) \vee p(c),\ p(a) \to p(b) \vee p(d),\ p(b) \to p(a) \vee p(d)\}$. Abbildung 3.4 zeigt ein PUHR-Complement-Tableau für $\mathcal{S}_5$. Im Gegensatz zum PUHR-Tableau in Abbildung 3.3 besitzt das PUHR-Complement-Tableau für $\mathcal{S}_5$ nur noch vier offene Zweige. Die minimalen Modelle $\mathcal{T}(\{p(a), p(b)\})$, $\mathcal{T}(\{p(a), p(d)\})$ und $\mathcal{T}(\{p(b), p(c), p(d)\})$ werden durch jeweils einen Zweig des Tableau repräsentiert. Der dritte offene Zweig von links repräsentiert das nichtminimale Modell $\mathcal{T}(\{p(a), p(b), p(c)\})$.

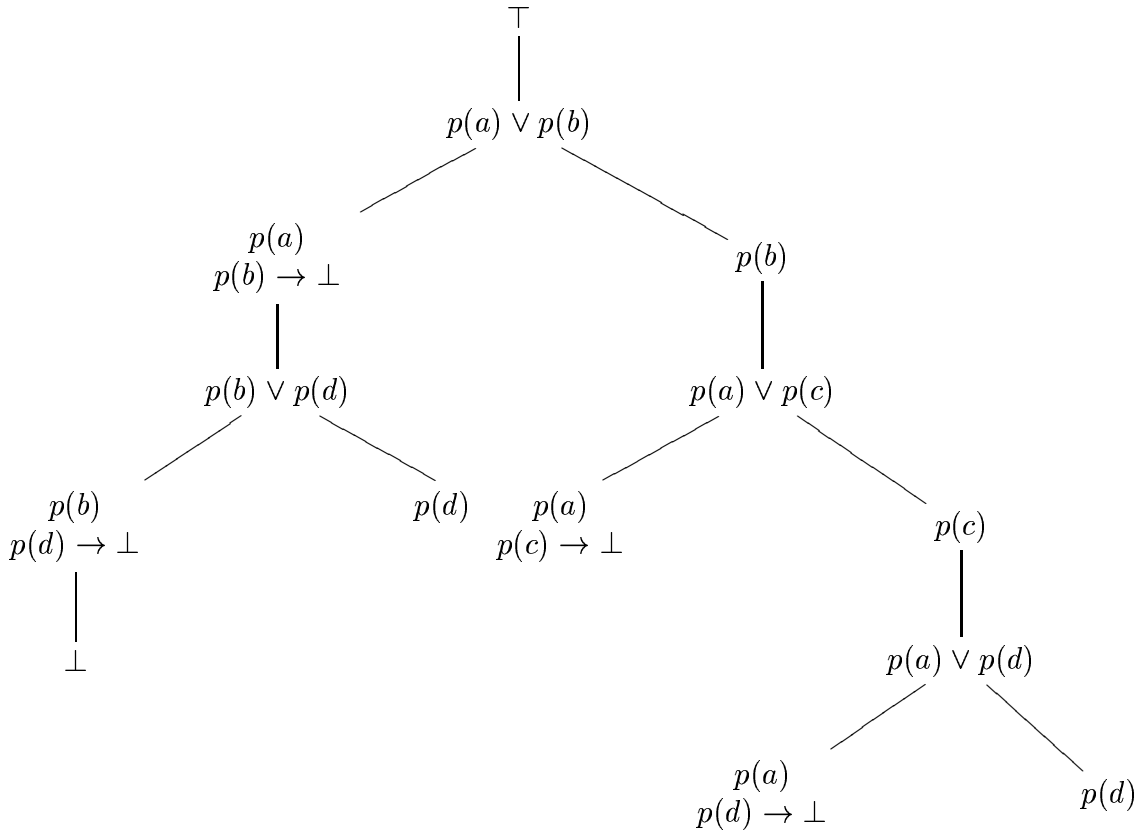


Abbildung 3.4: Ein PUHR-Tableau für Beispiel 3.28

Wie Beispiel 3.28 demonstriert hat, verhindert die Methode des Complement-Splitting zwar die Generierung von Duplikaten, die Generierung von nichtminimalen Modellen kann dagegen nicht ausgeschlossen werden. Dieses Problem kann aber durch die *beschränkte Suche*, die im nächsten Abschnitt vorgestellt wird, gelöst werden.

3.4.2 Beschränkte Suche

Die beschränkte Suche [BY96] basiert auf folgenden zwei Beobachtungen. Wird ein PUHR-Complement-Tableau von oben nach unten und von links nach rechts durchlaufen, so repräsentiert erstens der erste offene Zweig des Tableau ein minimales Modell. Zweitens repräsentiert jeder Zweig $\mathcal{B}_2$, der sich im PUHR-Complement-Tableau rechts von einem Zweig $\mathcal{B}_1$ befindet, ein $\leq$-größeres Modell als $\mathcal{B}_1$, sofern $\mathcal{B}_1$ und $\mathcal{B}_2$ bzgl. $\leq$ vergleichbar sind. „$\leq$“ bezeichne hier die in Definition 2.39 definierte Ordnung auf Termmodellen. Das Ziel der beschränkten Suche ist zu verhindern, daß nichtminimale Modelle generiert werden, d.h. Zweige eines PUHR-Complement-Tableau expandiert werden, die nichtminimale Modelle repräsentieren. Aufgrund der oben beschriebenen Beobachtungen kann das dadurch verhindert werden, das die Generierung eines Modells abgebrochen wird, falls bereits ein $\leq$-kleineres (endliches) Modell erzeugt wurde. Dies wird dadurch erreicht, daß die Menge $\mathcal{S}$ der gegebenen Klauseln durch $Neg(\{A_1, \ldots, A_n\}) := \{A_1 \wedge \cdots \wedge A_n \rightarrow \bot\}$ erweitert wird, sobald ein Modell $\mathcal{T}(\{A_1, \ldots, A_n\})$ erzeugt wurde.

Definition 3.29 (MM-Prozedur) Der Aufbau eines PUHR-Complement-Tableau T für eine Menge $\mathcal{S}$ von bereichsbeschränkten Klauseln heißt *MM-Prozedur*, falls

- T mit Tiefensuche und von links nach rechts aufgebaut wird und
- nach jeder Generierung eines Modells $\mathcal{T}(\mathcal{M})$ die Menge $\mathcal{S}$ durch $Neg(\mathcal{M})$ erweitert wird.

□

Im folgenden betrachten wir nur Klauselmengen, deren minimale Modelle durch *endliche* Mengen von Grundatomen repräsentiert werden können.

Definition 3.30 Eine Klauselmenge $\mathcal{S}$ heißt *finitär*, falls jedes minimale Termmodell von $\mathcal{S}$ durch eine endliche Menge von Grundatomen repräsentiert werden kann. □

Für finitäre Klauselmengen kann die Korrektheit und Vollständigkeit der MM-Prozedur gezeigt werden.

Satz 3.31 (Korrektheit und Vollständigkeit der MM-Prozedur) *Ist $\mathcal{S}$ finitär, so gilt für die MM-Prozedur angewendet auf $\mathcal{S}$ folgendes:*

1. *Die MM-Prozedur terminiert.*
2. *Die MM-Prozedur generiert alle minimalen Modelle (Vollständigkeit).*
3. *Jedes von der MM-Prozedur erzeugte Modell ist minimal (Korrektheit).*
4. *Kein Modell wird mehrmals generiert.*

An dieser Stelle sei darauf hingewiesen, daß die Eigenschaften *finitär* und *endlich erfüllbar* nicht gleichbedeutend sind. So ist eine finitäre Klauselmenge zwar endlich erfüllbar, die Umkehrung gilt jedoch nicht, wie folgendes Beispiel zeigt.

Beispiel 3.32 Die Klauselmenge $\mathcal{S} = \{\top \rightarrow p(a),\ p(x) \rightarrow q(x) \vee p(f(x))\}$ ist endlich erfüllbar. So ist z.B. die Terminterpretation $\mathcal{T}(\{p(a), q(a)\})$ ein endliches Termmodell von $\mathcal{S}$. Dagegen ist $\mathcal{S}$ nicht finitär, da das minimale Termmodell $\mathcal{T}(\{p(a), p(f(a)), p(f(f(a))), \ldots\})$ nicht endlich repräsentierbar ist.

Bemerkung 3.33 Auch die MM-Prozedur kann in SATCHMO integriert werden. Die vollständige Implementierung [BY96] findet sich in Anhang B.

Abschließend sei ein Ablauf der MM-Prozedur an einem Beispiel illustriert.

Beispiel 3.34 Betrachten wir noch einmal die Klauselmenge $\mathcal{S}_5$ aus Beispiel 3.22 und Beispiel 3.28: $\mathcal{S}_5 = \{\top \rightarrow p(a) \vee p(b),\ \top \rightarrow p(a) \vee p(c),\ p(a) \rightarrow p(b) \vee p(d),\ p(b) \rightarrow p(a) \vee p(d)\}$. Abbildung 3.5 zeigt einen Ablauf der MM-Prozedur angewandt auf $\mathcal{S}_5$. Es werden ausschließlich die minimalen Modelle von $\mathcal{S}_5$, $\mathcal{T}(\{p(a), p(b)\})$, $\mathcal{T}(\{p(a), p(d)\})$ und $\mathcal{T}(\{p(b), p(c), p(d)\})$, erzeugt.

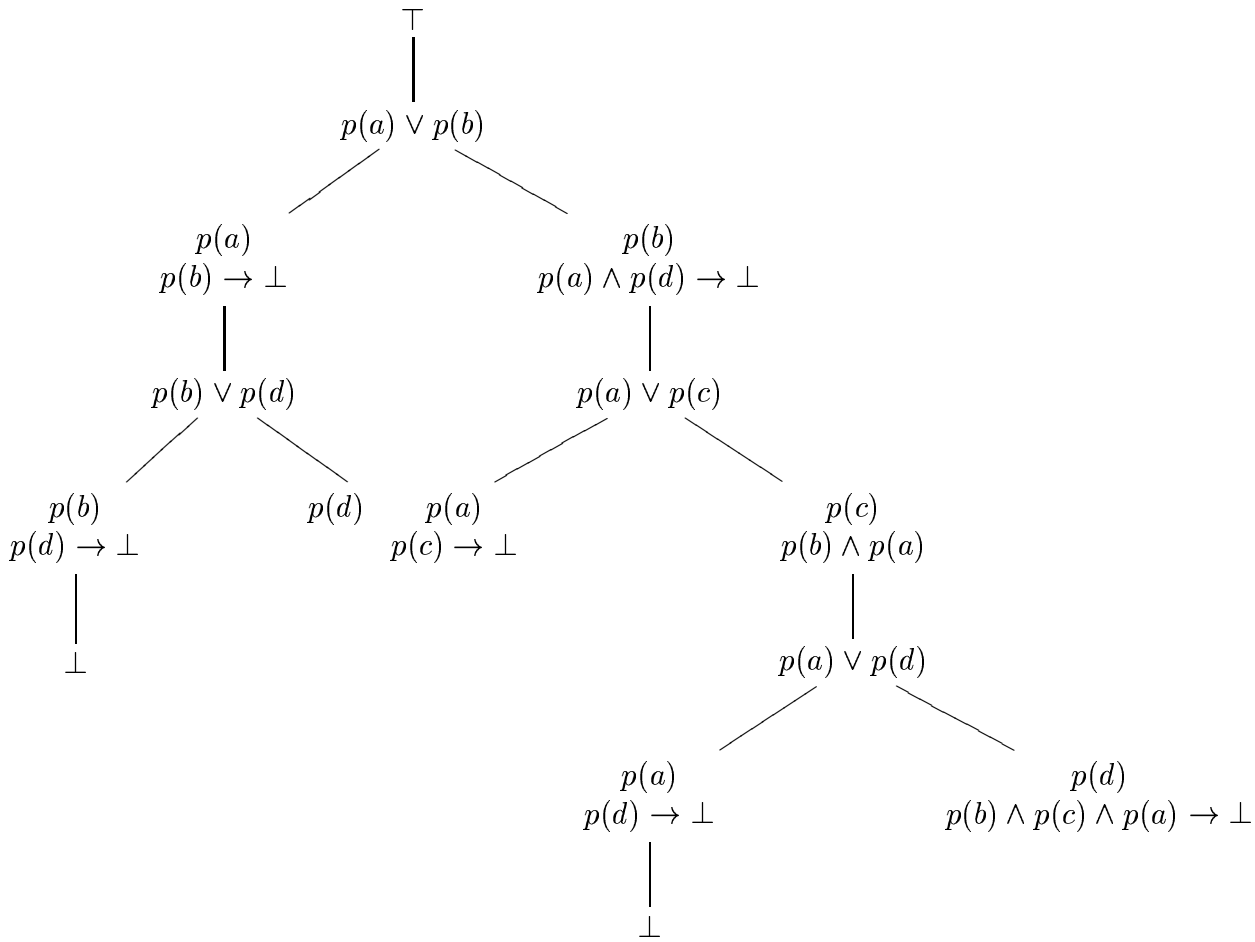


Abbildung 3.5: Ein Ablauf der MM-Prozedur für Beispiel 3.34.

Kapitel 4

Die EP-Tableaux-Methode

In diesem Kapitel wird die Extended-Positive-Tableaux-Methode (kurz EP-Tableaux-Methode) vorgestellt [BT98]. Die EP-Tableaux-Methode wurde mit dem Ziel entworfen, eine Methode zur Verfügung zu haben, die nicht nur korrekt und vollständig für die Unerfüllbarkeit (Widerlegung) sondern gleichzeitig auch vollständig für die Erfüllbarkeit im Endlichen ist. Die Überzeugung, daß gerade diese Eigenschaft von zentraler Bedeutung ist, wurde durch verschiedene Anwendungen motiviert. Kapitel 6 geht darauf genauer ein. Ausgangspunkt für die EP-Tableaux-Methode war die PUHR-Tableaux-Methode. Der Name „Extended-Positive-Tableaux-Methode“ bezieht sich darauf, daß die PUHR-Tableaux-Methode dahingehend erweitert wurde, daß wesentlich komplexere Formeln (verglichen mit Klauseln) durch die EP-Tableaux-Methode behandelt werden können. Gleichzeitig wurde aber das Prinzip, nur „positive Information“ zu betrachten, beibehalten, was für viele Anwendungen von großer Bedeutung ist, wie in Kapitel 6 zu sehen sein wird. Der Kern der PUHR-Tableaux-Methode, die Positive-Unit-Hyperresolution, wurde an die komplexeren Formeln angepaßt. Allerdings kann dann nicht mehr von Positive-Unit-Hyperresolution gesprochen werden, was sich auch im Namen EP-Tableaux-Methode niederschlägt. Aufgrund der komplexeren Formeln ist es jetzt wesentlich eleganter, semantische Begriffe anstelle des Begriffs der Positive-Unit-Hyperresolution einzusetzen.

In Abschnitt 4.3 wird die PRQ-Syntax, die der EP-Tableaux-Methode zugrundeliegt, eingeführt. Die PRQ-Syntax stellt eine deutliche Erweiterung der Klauselsyntax, wie sie in der PUHR-Tableaux-Methode verwendet wurde, dar. Welche Überlegungen von der Klauselsyntax zur PRQ-Syntax führen, wird in Abschnitt 4.1 zunächst anhand von Beispielen diskutiert. In Abschnitt 4.2 wird dann eine Vorstufe der PRQ-Syntax, die sogenannte erweiterte Syntax, eingeführt und die PUHR-Tableaux-Methode auf die erweiterte Syntax ausgedehnt. Gleichzeitig werden wir aber in Abschnitt 4.2 auch sehen, daß die erweiterte Syntax für unsere Belange noch nicht ausreicht und deshalb ein zweiter Erweiterungsschritt notwendig ist, der schließlich zur PRQ-Syntax (siehe Abschnitt 4.3) führt.

Nach der Bereitstellung der PRQ-Syntax können dann die EP-Tableaux in Abschnitt 4.4 definiert werden und die Ergebnisse über die Korrektheit und Vollständigkeit der EP-Tableaux-Methode (siehe Abschnitt 4.5 und 4.6) vorgestellt werden.

Es wird wieder eine Sprache angenommen, die keine Funktionssymbole enthält.

4.1 Motivation für die Erweiterung der Klauselsyntax

Beim Entwurf der Syntax der EP-Tableaux-Methode führte die Untersuchung der folgenden drei Punkte schrittweise von der Klauselsyntax zur PRQ-Syntax.

- Es muß geklärt werden, wie mächtig die zu definierende Sprache sein soll.
- Es stellt sich die Frage der Effizienz des Verfahrens in Abhängigkeit von der zugelassenen Syntax.
- Im Hinblick auf Anwendungen sollte die zugelassene Syntax natürlich und einfach zu verwenden sein.

Wie sich in Abschnitt 4.2 herausstellen wird, ist die Frage der Mächtigkeit von weitreichender Bedeutung. Doch für die ersten Erweiterungen der Klauselsyntax waren Überlegungen zur Effizienz des Verfahrens in Abhängigkeit von der zugelassenen Syntax ausschlaggebend. So stellt sich heraus, daß durch geringfügige Erweiterungen der Klauselsyntax die Anzahl der zu betrachtenden Formeln verringert werden kann. Anstelle von zwei oder mehreren Formeln genügt dann die Bearbeitung einer einzigen Formel. Gleichzeitig ist diese eine Formel aber kaum komplexer als eine Klausel, so daß ihre Bearbeitung kaum aufwendiger ist als die Bearbeitung einer Klausel. In diesem Sinn wird im folgenden Effizienzsteigerung verstanden. Die folgenden zwei Beispiele veranschaulichen diese Frage der Effizienz. Erinnern wir uns noch einmal an die PUHR-Tableaux-Methode und ihre Implementierung SATCHMO. Die zugelassenen Formeln waren bereichsbeschränkte Klauseln in Implikationsform. Da wir im Laufe dieses Abschnitts von der reinen Klauselform immer mehr abweichen werden, werden im folgenden bei Klauseln die universellen Quantoren explizit notiert.

Beispiel 4.1 Sei $\mathcal{S} = \{\top \to p(a),\ \forall x(p(x) \to q(x)),\ \forall x(p(x) \to r(x))\}$. Im folgenden wird beschrieben, was beim Ablauf von SATCHMO (siehe Abbildung 3.2, Seite 40), angewandt auf $\mathcal{S}$, passiert. In einem ersten Schritt wird $p(a)$ in die Datenbank eingefügt. Im nächsten Schritt müßte sowohl für $\forall x(p(x) \to q(x))$ als auch für $\forall x(p(x) \to r(x))$ getestet werden, ob der Rumpf der Klauseln durch die durch die Datenbank gegebene Terminterpretation erfüllt ist, obgleich für beide Klauseln der Rumpf derselbe ist. Dieses überflüssige doppelte Testen könnte dadurch umgangen werden, daß im Kopf einer Klausel auch Konjunktionen zugelassen wären. Denn die Klauseln $\forall x(p(x) \to q(x))$ und $\forall x(p(x) \to r(x))$ sind logisch äquivalent zu der einen Formel $\forall x(p(x) \to (q(x) \wedge r(x)))$. In diesem Fall müsste also nur einmal getestet werden, ob $p(x)$ erfüllt ist.

Es bietet sich also an, die Syntax dahingehend zu erweitern, daß im Kopf einer Klausel auch Konjunktionen zugelassen werden. Als Konsequenz daraus ist dann die Einführung einer zusätzlichen Tableauxexpansionsregel notwendig. Wird etwa die PUHR-Regel auf $\forall x(p(x) \to (q(x) \wedge r(x)))$ angewendet, so entsteht eine Konjunktion von Grundatomen, die weiter verarbeitet werden muß. Dafür wird eine $\wedge$-Regel benötigt, wie sie z.B. auch im klassischen Tableau [Smu68, Fit90] verwendet wird.

Eine ähnliche Einsparung von Tests, wie sie die Einführung der Konjunktion in Klauselköpfen mit sich bringt, kann auch durch die Einführung von Disjunktionen im Rumpf von Klauseln erreicht werden.

Beispiel 4.2 Sei $\mathcal{S} = \{\top \rightarrow p(a),\ \top \rightarrow q(a),\ \forall x(p(x) \rightarrow r(x)),\ \forall x(q(x) \rightarrow r(x))\}$. Hier wird in einem ersten Schritt $p(a)$ und $q(a)$ in die Datenbank eingefügt. Durch die so entstandene Terminterpretation ist sowohl $\forall x(p(x) \rightarrow r(x))$ als auch $\forall x(q(x) \rightarrow r(x))$ nicht erfüllt. Da wir die faire SATCHMO-Variante betrachten, werden beide Klauseln im nächsten Schritt erfüllt. Es wird also z.B. zuerst die Klausel $\forall x(p(x) \rightarrow r(x))$ behandelt und das Atom $r(a)$ in die Datenbank eingefügt. Anschließend kommt die Klausel $\forall x(q(x) \rightarrow r(x))$ an die Reihe, und erst durch einen neuerlichen Test wird erkannt, daß $\forall x(q(x) \rightarrow r(x))$ in der Zwischenzeit erfüllt wurde. Betrachten wir dagegen die zu $\forall x(p(x) \rightarrow r(x))$ und $\forall x(q(x) \rightarrow r(x))$ logisch äquivalente Formel $\forall x((p(x) \vee q(x)) \rightarrow r(x))$, so wird einmal getestet, ob die Disjunktion in der Implikation erfüllt ist und falls ja, wird die rechte Seite der Implikation einmal erfüllt.

Die Entscheidung, Disjunktionen auch im Rumpf von Klauseln zuzulassen, bringt aber Schwierigkeiten mit sich, wie das folgende Beispiel zeigen wird. Allerdings sind diese Schwierigkeiten durch eine Erweiterung des Begriffs der Bereichsbeschränktheit einfach zu behandeln.

Beispiel 4.3 Betrachten wir die Menge $\mathcal{S} = \{\top \rightarrow p(a),\ \forall x \forall y(p(x) \vee q(y) \rightarrow r(y))\}$. Im ersten Schritt wird $p(a)$ in die Datenbank eingeführt. Die so entstandene Terminterpretation $\mathcal{T}(\{p(a)\})$ ist kein Modell von $\forall x \forall y(p(x) \vee q(y) \rightarrow r(y))\}$. Gemäß der PUHR-Regel müßte nun die rechte Seite der Implikation $r(y)[a/x]$ erfüllt werden, d.h. in die Datenbank eingeführt werden. In diesem Fall ist aber die Variable y nicht instantiiert.

Die PUHR-Tableaux-Methode ist für bereichsbeschränkte Klauseln (siehe Definition 3.1) in Implikationsform definiert. Erinnern wir uns an die Definition einer bereichsbeschränkten Klausel in Implikationsform. Die Bereichsbeschränkung bedeutete, daß jede Variable, die im Kopf einer Klausel auftritt auch im Rumpf der Klausel vorkommt. Durch die Bereichsbeschränkung wird also erzwungen, daß jede Variable im Kopf instantiiert ist, sobald der Rumpf der Klausel instantiiert ist. Dies ist die Basis dafür, daß durch die Datenbank eine Terminterpretation repräsentiert wird, d.h. diese Eigenschaft der betrachteten Formeln muß auch bei einer Erweiterung der Syntax unbedingt erhalten bleiben. Wie Beispiel 4.3 aber zeigt, reicht der Begriff der Bereichsbeschränktheit, wie er in Abschnitt 3.1 definiert wurde, nicht aus. In $\forall x \forall y(p(x) \vee q(y) \rightarrow r(y))$ tritt y auch auf der linken Seite der Implikation auf. Trotzdem wird y in machen Fällen, wie oben beschrieben, nicht instantiiert. Wir werden in Abschnitt 4.2 sehen, wie der Begriff der Bereichsbeschränkung an eine erweiterte Syntax angepaßt werden kann.

Bevor die erweiterte Syntax formalisiert wird, wenden wir uns noch dem Aspekt der „Bequemlichkeit“ einer Syntax zu. Obgleich es möglich ist, „vieles“[1] durch logisch äquivalente Umformungen in die geforderte Klauselsyntax zu bringen, ist es trotzdem für Anwendungen von Vorteil, eine flexiblere Syntax zur Verfügung zu haben. So stellt sich die

[1] Was „vieles“ in diesem Zusammenhang bedeutet, wird in Abschnitt 4.2 noch genauer beleuchtet.

Frage, ob es nicht möglich wäre, auf der rechten Seite einer Implikation auch universell quantifizierte Implikationen, also geschachtelte Implikationen und Allquantoren, zuzulassen. Betrachten wir wieder ein Beispiel.

Beispiel 4.4 Statt der Klausel $K = \forall x \forall y(mensch(x) \wedge verheiratet(x, y) \rightarrow verheiratet(y, x))$ könnte auch die logisch äquivalente Formel $F = \forall x(mensch(x) \rightarrow \forall y(verheiratet(x, y) \rightarrow verheiratet(y, x)))$ betrachtet werden.

Allerdings hat auch die Einführung geschachtelter Implikationen und Allquantoren wieder verschiedene Konsequenzen, die im folgenden behandelt werden.

Die Formulierung der PUHR-Regel kann beibehalten werden. Jedoch entstehen durch die Anwendung der PUHR-Regel auf eine Formel der Form $\forall x(F_1 \rightarrow \forall y(F_2 \rightarrow F_3))$ nicht wie bisher nur Grundatome oder Disjunktionen von Grundatomen sondern vielmehr wieder universell quantifizierte Implikationen. Auf diese neu erzeugten Formeln sollte in einem weiteren Schritt wieder die PUHR-Regel anwendbar sein. Erinnern wir uns an die Definition der PUHR-Tableaux-Methode in Abschnitt 3.2, so wird dort unterschieden zwischen Formeln aus der gegebenen Klauselmenge $\mathcal{S}$ und Formeln, die durch Anwendung einer Expansionsregel entstanden sind. Da die PUHR-Regel ausschließlich auf Formeln aus der Menge $\mathcal{S}$ und die $\vee$-Regel ausschließlich auf neu entstandene Formeln angewendet werden kann, ist es möglich, die Menge $\mathcal{S}$ und das Tableau für $\mathcal{S}$ völlig getrennt zu betrachten. Für ein PUHR-Tableau für $\mathcal{S}$ kann deshalb auch als Startpunkt die Menge $\{\top\}$ gewählt werden. Andererseits müssen alle gegebenen Formeln, für die ein Tableau aufgebaut werden soll, in Klauselform vorliegen, insbesondere auch die Grundatome, was zu Klauseln z.B. der Form $\top \rightarrow p(a)$ führt.

Im Falle einer erweiterten Syntax mit geschachtelten Implikationen und Allquantoren kann diese strenge Trennung nicht mehr beibehalten werden. Die PUHR-Regel wird sowohl auf Formeln der gegebenen Formelmenge als auch auf Formeln, die durch Anwendung der Tableauxexpansionsregeln entstanden sind, angewendet. Daher scheint es sinnvoll, ein Tableau für die erweiterte Syntax mit der Menge $\mathcal{S}$ zu beginnen statt mit $\{\top\}$. Damit ist sichergestellt, daß alle Formeln, auf die eine Expansionsregel angewendet werden kann, im betreffenden Blatt des Tableau enthalten sind. Damit besteht andererseits aber auch die Möglichkeit, Grundatome und geschlossene Disjunktionen und Konjunktionen direkt zu formulieren anstatt sie in eine Implikation zu packen. Zum Beispiel kann die Formel $p(a) \vee \forall x(r(x) \rightarrow q(x))$ direkt durch die $\vee$-Regel bearbeitet werden. Dadurch vergrößern sich natürlich die Möglichkeiten für die Erweiterung der Syntax.

Wenn wir noch einmal die Definition der PUHR-Tableaux-Methode betrachten, so hat die Erweiterung der Syntax durch geschachtelte Implikationen und Allquantoren Konsequenzen für die alternative Definition (siehe Definition 3.11 und 3.12) der PUHR-Tableaux-Methode. In Abschnitt 3.2 wurde dargestellt, daß die PUHR-Tableaux-Methode sowohl über die Modellbeziehung bezüglich Terminterpretationen als auch über den Begriff der Positive-Unit-Hyperresolvente und der Subsumption durch Atome definiert werden kann. Für die erweiterte Syntax kann die Definition einer Tableaux-Methode über die Modellbeziehung bezüglich Terminterpretationen direkt von der PUHR-Tableaux-Methode über-

nommen werden. Die Definition über die Positive-Unit-Hyperresolvente und die Subsumption erfordert eine starke Erweiterung der beiden Begriffe, die allein für diesen Zweck nicht gerechtfertigt scheint.

4.2 Eine erste Erweiterung der Klauselsyntax

Nachdem in Abschnitt 4.1 verschiedene Möglichkeiten, die Klauselsyntax zu erweitern, diskutiert wurden, wird im folgenden eine erste Erweiterung der Klauselsyntax eingeführt. Der Kern dieser Erweiterung ist die Ausdehnung des Begriffs der Bereichsbeschränkung auf komplexere Formeln. Dazu ist der Begriff eines *einfachen Bereichs* für Variablen notwendig, der zunächst eingeführt wird. Nach der Definition der erweiterten Syntax kann dann die PUHR-Tableaux-Methode auf diese Syntax ausgedehnt werden.

Definition 4.5 (Einfacher Bereich von Variablen) Eine Formel B heißt *einfacher Bereich für die Variablen* $x_1, \ldots, x_n$, falls

1. B ein Atom ist, in dem $x_1, \ldots, x_n$ vorkommen.
2. B eine Konjunktion von Atomen ist und $x_1, \ldots, x_n$ in B vorkommen.
3. B eine Disjunktion ist und jedes Disjunktionsglied ein einfacher Bereich für $x_1, \ldots, x_n$ ist.

□

Beispiel 4.6 Die Formel $(p(x) \wedge r(y)) \vee q(x, y)$ ist ein einfacher Bereich für x und y, wohingegen die Formel $p(x) \vee q(x, y)$ kein einfacher Bereich für x und y ist, weil y im linken Disjunkt $p(x)$ nicht auftritt.

Damit können bereichsbeschränkte Formeln in erweiterter Syntax definiert werden.

Definition 4.7 (Erweiterte Syntax) Eine Formel F ist in *bereichsbeschränkter erweiterter Syntax*, falls

1. F ein Atom ist.
2. F die Formel $\top$ oder $\bot$ ist.
3. F eine Konjunktion oder Disjunktion von Formeln in erweiterter Syntax ist.
4. F von der Form $\forall x_1 \ldots x_n(B \rightarrow G)$ ist, wobei G eine Formel in erweiterter Syntax und B ein einfacher Bereich für $x_1, \ldots, x_n$ ist.

□

Nach dieser Definition ist z.B. auch die Formel $\forall x\ (p(x) \rightarrow q(y))$ in bereichsbeschränkter erweiterter Syntax. Dies stellt aber kein Problem dar, weil im folgenden nur geschlossene Formeln in bereichsbeschränkter erweiterter Syntax betrachtet werden.

Beispiel 4.8 Die Formel $\forall x \forall y\ ((p(x) \wedge r(y)) \vee q(x,y) \rightarrow \forall z(s(z) \vee t(z) \rightarrow (u(z) \wedge v(z))$ ist in bereichsbeschränkter erweiterter Syntax, die Formel $\forall x(p(x) \rightarrow \forall y(s(x) \rightarrow (u(y) \wedge v(y))$ dagegen nicht, da es für y keinen Bereich gibt.

Um die Erweiterung der PUHR-Tableaux-Methode für die erweiterte Syntax zu definieren, werden zuerst die Expansionsregeln definiert. Wie bereits oben motiviert, wird eine weitere Expansionsregel, die $\wedge$-Regel, benötigt, die der α-Regel von [Smu68, Fit90] entspricht. Die PUHR-Regel der PUHR-Tableaux-Methode wird zu einer erweiterten PUHR-Regel, deren Definition nicht mehr auf dem Begriff der Positiv-Unit-Hyperresolution sondern auf der Modellbeziehung beruht.

Definition 4.9 (Expansionsregeln)

Erweiterte PUHR-Regel:	$\vee$ -Regel:	$\wedge$-Regel:
$\dfrac{\forall \bar{x}(B(\bar{x}) \rightarrow F)}{F\sigma}$	$\dfrac{E_1 \vee E_2}{E_1 \mid E_2}$	$\dfrac{E_1 \wedge E_2}{\begin{array}{c} E_1 \\ E_2 \end{array}}$

wobei $\sigma = \{\bar{c}/\bar{x}\}$ eine Grundsubstitution ist und $B\sigma$ durch die Terminterpretation erfüllt wird, die durch die Menge der Grundatome, die im expandierten Knoten vorkommen, bestimmt wird. □

Damit können jetzt Tableaux für die erweiterte Syntax definiert werden.

Definition 4.10 (Erweiterte PUHR-Tableaux) Erweiterte PUHR-Tableaux für eine Menge $\mathcal{S}$ von geschlossenen Formeln in bereichsbeschränkter erweiterter Syntax sind Bäume, deren Knoten Mengen von geschlossenen Formeln sind. Erweiterte PUHR-Tableaux werden folgendermaßen induktiv definiert:

1. Der Baum, der nur aus dem einzigen Knoten $\mathcal{S}$ besteht, ist ein erweitertes PUHR-Tableau für $\mathcal{S}$.

2. Sei T ein erweitertes PUHR-Tableau für $\mathcal{S}$, sei L ein Blatt von T und F eine Formel in L, die von der Terminterpretation $\mathcal{T}(Atoms(L))$ nicht erfüllt wird. Werden an T ein oder mehrere Kinderknoten angehängt, die durch Anwendung der passenden Expansionsregel auf F entstehen, so ist der entstandene Baum wieder ein erweitertes PUHR-Tableau für $\mathcal{S}$. Jeder Kinderknoten enthält dabei alle Formeln aus L sowie eine bzw. im Falle der Konjunktion zwei neue Formeln gemäß der Expansionsregel in Definition 4.9. Bei einer Konjunktion wird ein Konjunktionsglied nur dann angefügt, falls es nicht von $\mathcal{T}(Atoms(L))$ erfüllt wird. □

Damit wurde eine Tableaux-Methode für die erweiterte Syntax definiert, und es stellt sich als nächstes die Frage nach den Eigenschaften der so definierten Methode. Diese Untersuchungen werden noch zurückgestellt, da die in den folgenden Abschnitten definierte Extended-Positive-Tableaux-Methode eine direkte Erweiterung der oben definierten Tableaux-Methode ist und diese also mit einschließt.

Eine andere Frage dagegen, die zu Beginn von Abschnitt 4.1 formuliert wurde, muß an dieser Stelle diskutiert werden, nämlich die Frage nach der Mächtigkeit der festgelegten Sprache. Dem Entwurf der erweiterten Syntax sind verschiedene Überlegungen vorausgegangen. Es wurden der Einfluß der Syntax auf die Effizienz untersucht und die Frage der Bequemlichkeit und Natürlichkeit im Hinblick auf Anwendungen diskutiert. Auf die Frage der Mächtigkeit der Sprache wurde bis jetzt aber noch nicht eingegangen. Natürlich ist es wünschenswert eine möglichst mächtige Sprache zur Verfügung zu haben, da dadurch der Anwendungsbereich der Sprache erweitert wird. Andererseits sind aber bezüglich der Mächtigkeit aufgrund der verschiedenen Entscheidbarkeitsergebnisse wiederum Grenzen gesetzt. Bekanntlich ist die Sprache der ersten Stufe da ein geeigneter Kompromiß, was einerseits die Mächtigkeit und andererseits die Ergebnisse bezüglich Entscheidbarkeit und Semi-Entscheidbarkeit angeht.

Es bleibt also zu untersuchen, ob die hier definierte Sprache[2] die volle Sprache der ersten Stufe ohne Funktionssymbole bereits abdeckt, und falls dies nicht der Fall ist, wie die Sprache erweitert werden könnte, um die gesamte Logik der ersten Stufe ohne Funktionssymbole abzudecken.

Die erste Frage ist einfach zu beantworten: Die durch die erweiterte Syntax definierte Sprache (ohne Funktionssymbole), wie sie oben definiert wurde, ist sicher nicht so mächtig wie die Sprache der ersten Stufe ohne Funktionssymbole, denn jede Formel in der erweiterten Syntax ist logisch äquivalent zu einer Menge von Klauseln. Eine Klauselsprache ohne Funktionssymbole hat aber bekanntlich eine geringere Mächtigkeit als die volle Logik der ersten Stufe ohne Funktionssymbole. Zum Beispiel läßt sich damit nicht ausdrücken, daß jede natürliche Zahl einen Nachfolger hat.

Die zweite Frage nach einer Erweiterung der erweiterten Syntax hat mehrere Antworten. Um die volle Logik der ersten Stufe ohne Funktionssymbole abzudecken, kann die durch die erweiterte Syntax definierte Sprache entweder um Funktionssymbole oder um Existenzquantoren erweitert werden.[3]

Im Gebiet der Automatischen Deduktion wird die gewünschte Ausdrucksstärke meistens durch eine Sprache mit Funktionssymbolen sichergestellt. Es ist sogar weit verbreitet, eine Sprache mit Funktionssymbolen *und* Existenzquantoren zuzulassen, aber vor der eigentlichen Verarbeitung der Formeln die Existenzquantoren mittels der sogenannten *Skolemisierung* auf Funktionssymbole zurückzuführen.

Beispiel 4.11 Aus $F = p(a) \wedge \forall x(p(x) \rightarrow \exists y p(y))$ entsteht durch Skolemisierung die Formel $Sk(F) = p(a) \wedge \forall x(p(x) \rightarrow p(f(x)))$. Die Formeln F und $Sk(F)$ sind zwar nicht

[2]Wie in der ganzen Arbeit nehmen wir an, daß in der Signatur keine Funktionssymbole enthalten sind.

[3]Strenggenommen sind die beiden Mächtigkeiten nur in einer Sprache mit Gleichheit wechselseitig simulierbar. Aber die Gleichheit kann hinreichend approximiert werden und wird hier deshalb nicht betrachtet.

logisch äuqivalent, aber erfüllbarkeitsäquivalent. Dies ist für die meisten Anwendungen in der Automatischen Deduktion völlig ausreichend.

Auch für die PUHR-Tableaux-Methode wurde dieser Weg beschritten. Sie wurde ursprünglich für eine Sprache mit Funktionssymbolen definiert, damit eine Sprache der gewünschten Mächtigkeit zur Verfügung steht.

Im Hinblick auf die Erfüllbarkeit im Endlichen, die das Hauptanliegen dieser Arbeit ist, sind Funktionssymbole aber problematisch. Eine Sprache mit Funktionssymbolen hat immer ein unendliches Herbranduniversum. Damit sind endliche Herbrandmodelle von vornherein ausgeschlossen. Für Termmodelle wäre zunächst eine Erweiterung der Definition für Sprachen mit Funktionssymbolen notwendig. Prinzipiell stellen sich hier aber die gleichen Schwierigkeiten wie bei Herbrandmodellen.

Aus diesem Grund bietet es sich an, die bereits definierte Sprache statt um Funktionssymbole besser um Existenzquantoren zu erweitern, um so die volle Logik der ersten Stufe ohne Funktionssymbole zur Verfügung zu haben. Natürlich dürfen die Existenzquantoren dann nicht mittels Skolemisierung behandelt werden. Dadurch würden zum einen doch wieder Funktionssymbole eingeführt werden. Zum anderen bringt die Skolemisierung weitere Schwierigkeiten hinsichtlich der Erfüllbarkeit im Endlichen mit sich.

Beispiel 4.12 Betrachten wir F und $Sk(F)$ aus dem vorigen Beispiel 4.11. Das einfachste Termmodell von F ist $\mathcal{T}(\{p(a)\})$. Dieses Termmodell ist endlich. Für $Sk(F)$ ist der Begriff des Termmodells noch nicht definiert. Die naheliegende Definition, die auch dem Verhalten der PUHR-Tableaux-Methode in diesem Fall entspricht, ergäbe aber das unendliche Termmodell $\mathcal{T}(\{p(a), p(f(a)), p(f(f(a))), \dots\})$. So entsteht also durch Skolemisierung aus einer endlich erfüllbaren Formel eine Formel, die nur von unendlichen Termmodellen erfüllt wird.

Da für die PUHR-Tableaux-Methode existentiell quantifizierte Formeln skolemisiert werden müssen, tritt hier genau das im Beispiel beschriebene Problem auf. Insbesondere terminiert die PUHR-Tableaux-Methode bei der Verarbeitung der skolemisierten Formel nicht und wird deshalb unvollständig für die Erfüllbarkeit im Endlichen. Diese Überlegungen motivierten die Erweiterung der PUHR-Tableaux-Methode um Existenzquantoren und den Verzicht auf Skolemisierung. Somit stellt sich die Frage, welche anderen Möglichkeiten es neben der Skolemisierung gibt, um Existenzquantoren zu behandeln.

Im klassischen Tableauxverfahren, wie es etwa in [Smu68, Fit90] formuliert ist, werden die Existenzquantoren mittels der sogenannten δ-Regel explizit behandelt. Die δ-Regel veranlaßt, daß für jedes Auftreten eines Existenzquantors eine neue Konstante eingeführt wird, mit der die existentiell quantifizierte Variable dann instantiiert wird.

δ-Regel

$$\frac{\exists x F(x)}{F\{c/x\}}$$

wobei c eine neue, d.h. in diesem Zweig nichtauftretende, Konstante ist.

Diese Vorgehensweise kann auch als „Laufzeit-Skolemisierung“ bezeichnet werden, denn die neu eingeführten Konstanten entsprechen den neuen Elementen, die bei der Skolemisierung durch eine Aufschachtelung der Terme aufgrund der Skolemfunktion (das ist die für die Existenzquantoren neu eingeführte Funktion) entstehen. Deshalb führt auch die Anwendung der δ-Regel im Hinblick auf die Generierung endlicher Modelle zu den gleichen Schwierigkeiten wie die Skolemisierung.

Beispiel 4.13 Bei der Formelmenge $M = \{p(a), \forall x(p(x) \rightarrow \exists y q(y), \forall x(q(x) \rightarrow p(x))\}$ führt die Anwendung der δ-Regel nur zu einem unendlichen Termmodell $\mathcal{T}(\{p(a),\ q(c_1),\ p(c_1),\ q(c_2),\ p(c_2),\ q(c_3)\ldots\})$, obwohl die Menge M endlich erfüllbar ist.

Um ein endliches Termmodell zu generieren, wäre es in Beispiel 4.13 überhaupt nicht notwendig, eine neue Konstante einzuführen. Das Termmodell $\mathcal{T}(\{p(a)\})$ wäre bereits ein endliches Modell. Doch die δ-Regel erzwingt, daß immer neue Konstanten eingeführt werden. Dies kann aber durch eine Abwandlung der δ-Regel [BM87, Hin88, Lor94] vermieden werden. Die Idee dabei ist, die existentiell quantifizierte Variable zunächst mit Konstanten zu instantiieren, die bereits im betreffenden Zweig existieren und nur als zusätzliche Möglichkeit eine neue Konstante einzuführen. Mit dieser Methode werden die Modelle „klein“ gehalten und nur wenn es unumgänglich ist, vergrößert. Damit ist der Kerngedanke der Extended-Positive-Tableaux-Methode beschrieben. Sie ist eine Erweiterung der in diesem Abschnitt vorgestellten Tableaux-Methode durch die neue δ-Regel [BM87, Hin88, Lor94] zur Behandlung von Existenzquantoren. Die Details werden in Abschnitt 4.4 genau erklärt.

Die gleiche Idee, die der Behandlung der Existenzquantoren in der EP-Tableaux-Methode zugrundeliegt, kann auch für die Behandlung von Funktionssymbolen eingesetzt werden [BM86]. Tritt ein „neuer“ Grundterm mit einem Funktionssymbol in einer Formel auf, so wird zunächst versucht, diesen Grundterm durch bereits bekannte Grundterme zu ersetzen. Allerdings entstehen bei dieser Methode mehrere Schwierigkeiten. Erstens können Funktionssymbole in beliebiger Schachteltiefe auftreten. Deshalb kann sich die Ersetzung eines Terms durch einen anderen als äußerst aufwendig erweisen. Zum zweiten können immer wieder neue Terme erzeugt werden. Angenommen, es wurden alle Grundterme der Form $f(a)$ durch die Konstante a ersetzt. Jedoch enthält die betrachtete Formelmenge die Formel $\forall x\ (p(x) \rightarrow p(f(x)))$. Dann werden durch die Bearbeitung dieser Formel möglicherweise wieder neue Grundatome erzeugt, in denen der Term $f(a)$ auftritt. Es muß also während der gesamten Bearbeitung der Formelmenge immer wieder neu sichergestellt werden, daß alle relevanten Terme durch die passenden ersetzt werden. Dagegen genügt für die Bearbeitung einer Formel mit einem Existenzquantor die einmalige Behandlung des Existenzquantors. Zudem müssen die Verträglichkeitsaxiome für Funktionen axiomatisiert werden und zu der betrachteten Formelmenge hinzugenommen werden, d.h. für jedes Relationssymbol p und jedes Funktionssymbol f mußdie entsprechende Formel hinzugenommen werden:

$$\forall x_1 \ldots x_n, y_1, \ldots y_n\ (x_1 = y_1 \wedge \ldots \wedge x_n = y_n \rightarrow (p(x_1 \ldots x_n) \rightarrow p(y_1, \ldots, y_n)))$$
$$\forall x_1 \ldots x_n, y_1, \ldots y_n\ (x_1 = y_1 \wedge \ldots \wedge x_n = y_n \rightarrow (f(x_1 \ldots x_n) = f(y_1, \ldots, y_n)))$$

Hierbei muß die Gleichheit, die uns bis jetzt nicht zur Verfügung steht, behandelt werden. Es ist durchaus denkbar, daß die hier beschriebenen Probleme durch die Integration bestehender Methoden der Gleichheitsbehandlung (wie sie etwa in [Her93, DdS94] vorgestellt wurden) und der Termersetzung zu lösen sind. Dies zu untersuchen, geht jedoch über den Rahmen dieser Arbeit hinaus. Da die Behandlung von Funktionssymbolen mittels einer solchen kombinierten Methode wesentlich aufwendiger erscheint als die explizite Behandlung von Existenzquantoren, wurde der Behandlung des Existenzquantors mit einer erweiterten δ-Regel gegenüber der Skolemisierung der Vorzug gegeben.

Bevor nun die Extended-Positive-Tableaux-Methode formuliert werden kann, wird im nächsten Abschnitt die zugehörige Syntax der positiven Formeln mit restringierter Quantifikation (kurz: PRQ-Formeln) definiert werden. Die PRQ-Formeln entstehen durch eine systematische Erweiterung der erweiterten Syntax um Existenzquantoren.

4.3 Positive Formeln mit restringierter Quantifikation

In diesem Abschnitt wird ein Fragment der Prädikatenlogik der ersten Stufe vorgestellt, welches mit „**P**ositive Formeln mit **R**estringierter **Q**uantifikation“ (kurz PRQ-Formeln) bezeichnet wird.[4] Verglichen mit der erweiterten Syntax, wie sie in Abschnitt 4.1 definiert wurde, besteht die wesentliche Erweiterung darin, Existenzquantoren explizit in der Syntax zuzulassen. Dadurch gelingt es, eine Sprache zu definieren, die gleichmächtig wie die volle Logik der ersten Stufe ist.

Genau wie bereits für die erweiterte Syntax muß auch für die Erweiterung mit Existenzquantoren der Begriff der Bereichsbeschränktheit ausgedehnt werden. Genauer bedeutet das, sobald auf ein Objekt in einer Formel Bezug genommen wird, d.h. eine Variable quantifiziert wird, sorgt ein positiver Ausdruck – ein *Bereich* – dafür, zu klären, von welcher Art das gemeinte Objekt ist. Daher kommt auch der Name PRQ-Formel, wo auf die durch Bereiche beschränkte Quantifikation hingewiesen wird. Die zugrundeliegende Sprache ist eine Sprache ohne Funktionssymbole.

Beispiel 4.14 Betrachten wir die Formel

$$\forall x\ (\underline{p(x)} \rightarrow \exists y\ (\underline{r(y)}\ \wedge\ q(x,y))).$$

Die unterstrichenen Ausdrücke sind Bereiche für x bzw. y. Um einen Bereich für eine universell (bzw. existentiell) quantifizierte Variable einzuführen, wird eine Implikation (bzw. eine Konjunktion) verwendet, entsprechend der Standardtransformation von einer Formel in einer mehrsortigen Sprache in eine Formel in einsortiger Sprache [EFT92].

Die formale Definition von PRQ-Formeln basiert auf den Hilfsbegriffen *Positive Bedingung* und *Bereich*, die zunächst eingeführt werden.

[4] Im Englischen wird dieses Fragment mit „**P**ositive Formulas with **R**estricted **Q**uantification“ bezeichnet. Dem englischen „restricted“ entspricht das deutsche „beschränkt“. Um aber das Akronym übertragen zu können, wurde hier der in der Mathematik nicht unübliche Begriff „restringiert“ verwendet.

Definition 4.15 (Positive Formeln mit restringierter Quantifikation)
• *Positive Bedingungen* werden folgendermaßen induktiv definiert:

1. Atome sind positive Bedingungen.
2. Die Formel $\top$ ist eine positive Bedingung.
3. Konjunktionen und Disjunktionen von positiven Bedingungen sind positive Bedingungen.
4. $\exists yF$ ist eine positive Bedingung, falls F eine positive Bedingung ist.

• *Bereiche für Variablen* $x_1, \ldots, x_n$ werden folgendermaßen induktiv definiert:

1. Ein Atom, in dem alle Variablen $x_1, \ldots, x_n$ vorkommen, ist ein Berich für $x_1, \ldots, x_n$.
2. $B_1 \vee B_2$ ist ein Bereich für $x_1, \ldots, x_n$, falls sowohl B_1 als auch B_2 ein Bereich für $x_1, \ldots, x_n$ sind.
3. $B_1 \wedge B_2$ (bzw. $B_2 \wedge B_1$) ist ein Bereich für $x_1, \ldots, x_n$, falls B_1 ein Bereich für $x_1, \ldots, x_n$ und B_2 eine positive Bedingung ist.
4. $\exists yB$ ist ein Bereich für $x_1, ..., x_n$, falls B ein Bereich für $y, x_1, ..., x_n$ und $x_i \neq y$ für alle $i = 1, \ldots, n$ ist.

• *Positive Formeln mit Restringierter Quantifikation* (kurz *PRQ-Formeln*) werden folgendermaßen induktiv definiert:

1. Atome sind PRQ-Formeln.
2. Die Formeln $\bot$ und $\top$ sind PRQ-Formeln.
3. Konjunktionen und Disjunktionen von PRQ-Formeln sind PRQ-Formeln.
4. Eine Formel der Gestalt $P \to F$ ist eine PRQ-Formel, falls P eine positive Bedingung und F eine PRQ-Formel ist.
5. Eine Formel der Gestalt $\forall x_1 \ldots x_n\ (B \to F)\ (n \geq 1)$ ist eine PRQ-Formel, falls B ein Bereich für $x_1, \ldots, x_n$ und F eine PRQ-Formel ist.
6. Eine Formel der Gestalt $\exists x(B \wedge F)$ ist eine PRQ Formel, falls B ein Bereich für x und F eine PRQ-Formel ist.

□

An dieser Stelle sei darauf hingewiesen, daß im folgenden ausschließlich geschlossene PRQ-Formeln betrachtet werden. Ist dies nicht der Fall, so wird es explizit vermerkt.

Bemerkung 4.16 Bereiche sind insbesondere auch positive Bedingungen.

Im Gegensatz zu der erweiterten Syntax, wie sie in Abschnitt 4.1 definiert wurde, können in PRQ-Formeln Existenzquantoren auftreten. Dies ist sowohl in Bereichen von Variablen als auch geschachtelt in Implikationen möglich.

Beispiel 4.17 Die Formel $F_1 = \forall x\ (\exists y p(x,y) \rightarrow q(y))$ ist eine PRQ-Formel, da $\exists y p(x,y)$ ein Bereich für x ist. Die Formel $F_2 = \forall xy\ (p(x) \wedge q(y) \vee r(x,y) \rightarrow \exists z\ (s(z) \wedge t(z)))$ ist ebenfalls eine PRQ-Formel, da $p(x) \wedge q(y) \vee r(x,y)$ ein Bereich für x und y und $s(z)$ ein Bereich für z ist. Die Formel $F_3 = \forall xy\ (p(x) \vee q(y) \rightarrow s(y))$ ist dagegen keine PRQ-Formel, da die linke Seite der Implikation kein Bereich für x und y ist.

Bei PRQ-Formeln der Form $P \rightarrow F$ ist die Teilformel P eine positive Bedingung. Dies ist genauso wie bei quantifizierten Formeln wegen des Prinzips der „positiven Information“ notwendig. Wird eine Implikation als universell quantifizierte Formel mit „leerem“ Quantor aufgefaßt, so muß die Teilformel P die Rolle eines Bereichs übernehmen. Eine positive Bedingung ist in gewisser Hinsicht eine Verallgemeinerung eines Bereichs. Wird auf die Forderung nach einer positiven Bedingung verzichtet, entstehen folgende Schwierigkeiten:

Beispiel 4.18 Wir betrachten die Formel $F = \neg p(a) \rightarrow q(a)$. Diese Formel wird von der Terminterpretation, die durch die leere Menge repräsentiert wird, nicht erfüllt. Soll ein Termmodell von F generiert werden, muß also das Grundatom $q(a)$ hinzugenommen werden. Im Laufe einer Berechnung kann jedoch aufgrund von anderen Formeln das Grundatom $p(a)$ hinzugenommen werden. Dadurch würde $q(a)$ überflüssig werden. Dies kann jedoch nicht mehr erkannt werden.

Satz 4.19 *Jede bereichsbeschränkte Klausel ist eine PRQ-Formel.*

Beweis: Folgt direkt aus der Definition 4.15. ∎

Bemerkung 4.20 In der Definition 4.15 der PRQ-Syntax fällt eine gewisse Asymmetrie bezüglich der Definition universell quantifizierter und existentiell quantifizierter Formeln auf. So werden Vektoren von Allquantoren zugelassen. Der Existenzquantor tritt dagegen nur einzeln auf. Dies ist wegen der speziellen Behandlung des Existenzquantors sinnvoll wie wir später sehen werden. Da die Behandlung von Vektoren von Allquantoren dagegen keinerlei Schwierigkeiten macht, sollen sie zugelassen werden. Im Hinblick auf Anwendungen ist diese Asymmetrie sinnvoll.

Das folgende Lemma wird in Abschnitt 4.6 im Beweis von Theorem 4.48 benötigt.

Lemma 4.21 *Seien $\mathcal{M}$ und $\mathcal{N}$ Mengen von Grundatomen mit $\mathcal{M} \subseteq \mathcal{N}$, sei P eine positive Bedingung und β eine Variablenbelegung. Dann gilt: Ist $\mathcal{T}(\mathcal{M})_\beta$ ein Termmodell von P, so ist auch $\mathcal{T}(\mathcal{N})_\beta$ ein Termmodell von P.*

Beweis: Durch Induktion über den Aufbau einer positiven Bedingung.

Sei P ein Atom: Seien $x_1, \ldots, x_n,\ n \in \mathbb{N}_0$ die freien Variablen in P. Nach Voraussetzung ist $\mathcal{T}(\mathcal{M})_\beta$ ein Termmodell von P. Da die Variablenbelegung als Grundsubstitution repräsentiert werden kann, gibt es eine Grundsubstitution $\sigma = \{c_1/x_1, \ldots, c_n/x_n\},\ n \in \mathbb{N}_0$, so daß $\mathcal{T}(\mathcal{M})_\beta \models P\sigma$. Also ist $P\sigma$ ein Element aus $\mathcal{M}$ ist. Wegen $\mathcal{M} \subseteq \mathcal{N}$ ist $P\sigma$ auch ein Element aus $\mathcal{N}$, d.h. $\mathcal{T}(\mathcal{N})_\beta$ ist ein Termmodell von $P\sigma$ und damit auch von P.

Für $P = \top$ ist nichts zu zeigen, da $\top$ von jeder Interpretation erfüllt wird.

$P = P_1 \wedge P_2$: Nach Voraussetzung ist $\mathcal{T}(\mathcal{M})_\beta$ ein Termmodell von $P_1 \wedge P_2$, also ist $\mathcal{T}(\mathcal{M})_\beta \models P_1$ und $\mathcal{T}(\mathcal{M})_\beta \models P_2$. Nach Induktionsvoraussetzung ist nun $\mathcal{T}(\mathcal{N})_\beta$ ein Termmodell von P_1 und von P_2, und damit erhalten wir $\mathcal{T}(\mathcal{N})_\beta \models P_1 \wedge P_2$.

$P = P_1 \vee P_2$: Nach Voraussetzung ist $\mathcal{T}(\mathcal{M})_\beta$ ein Termmodell von $P_1 \vee P_2$, also ist $\mathcal{T}(\mathcal{M})_\beta \models P_1$ oder $\mathcal{T}(\mathcal{M})_\beta \models P_2$. Nach Induktionsvoraussetzung ist nun $\mathcal{T}(\mathcal{N})_\beta$ ein Termmodell von P_1 oder von P_2, und damit erhalten wir $\mathcal{T}(\mathcal{N})_\beta \models P_1 \vee P_2$.

$P = \exists x\ B$: Nach Voraussetzung ist $\mathcal{T}(\mathcal{M})_\beta$ ein Termmodell von $P = \exists x\ B$, d.h. es existiert ein Element d in der Domäne $\mathcal{D}$ von $\mathcal{T}(\mathcal{M})_\beta$, so daß $\mathcal{T}(\mathcal{M})_{\beta\frac{d}{x}} \models B$. Also ist $\mathcal{T}(\mathcal{M})_\beta$ ein Termmodell von $B\sigma$ mit $\sigma = \{d/x\}$. Nach Induktionsvoraussetzung ist dann auch $\mathcal{T}(\mathcal{N})_\beta \models B\sigma$ und damit ist $\mathcal{T}(\mathcal{N})_\beta$ ein Termmodell von $P = \exists x\ B$. ∎

Für die meisten Anwendungen ist die Einschränkung auf PRQ-Formeln keine starke Einschränkung, denn i.a. ist in der natürlichen Sprache die Quantifikation durch implizite oder explizite Sorten eingeschränkt.

Beispiel 4.22 In der Formel

$$\forall x\ (\underline{mensch(x)} \rightarrow \exists y\ (\underline{mutter(y)}\ \wedge\ kind_von(x, y))).$$

übernehmen die unterstrichenen Ausdrücke, die auch mittels Sorten ausgedrückt werden könnten, die Rolle der Bereiche für x bzw. y.

Wie bereits erwähnt, sind die PRQ-Formeln in einem gewissen Sinne gleichmächtig wie die volle Logik der ersten Stufe. Es wird sich zeigen, daß es für jede endliche Menge $\mathcal{F}$ von Formeln der ersten Stufe eine endliche Menge PRQ($\mathcal{F}$) von PRQ-Formeln gibt mit den „gleichen“ Modellen. In welchem Sinne „gleiche Modelle“ verstanden wird, präzisiert Satz 4.24. Dafür wird zunächst eine Transformation definiert, die eine beliebige Formel in Implikationsform in die PRQ-Syntax überführt. Dabei liegt folgende Idee zugrunde. Es wird ein zusätzliches Relationssymbol zur gegebenen Sprache hinzugenommen. Mittels dieses Relationssymbols wird die Domäne eines möglichen Termmodells der gegebenen Formel repräsentiert. Diese Repräsentation der Domäne übernimmt dann die Rolle eines Bereichs.

Definition 4.23 (Transformation in PRQ-Formeln) Sei Σ die Signatur der betrachteten Sprache der ersten Stufe, D ein einstelliges Relationssymbol, das nicht in Σ enthalten ist und Σ' die Vereinigung von Σ und $\{D\}$. Weiter sei $\mathcal{R}$ folgende Transformation für Formeln in Implikationsform:

- $\mathcal{R}(\forall x\varphi) := \forall x(D(x) \rightarrow \mathcal{R}(\varphi))$
- $\mathcal{R}(\exists x\varphi) := \exists x(D(x) \wedge \mathcal{R}(\varphi))$
- $\mathcal{R}(\varphi) := \varphi$, falls φ nicht quantifiziert ist.

Sei $\mathcal{F}$ eine Menge von Formeln erster Stufe und $\mathcal{F}'$ eine Menge der zu $\mathcal{F}$ logisch äquivalenten Formeln in Implikationsform. Dann sei

$$\mathrm{PRQ}(\mathcal{F}) := \mathrm{PRQ}(\mathcal{F}') \cup K(\mathcal{F})$$

die zu $\mathcal{F}$ gehörende Menge von PRQ-Formeln, wobei

$$\mathrm{PRQ}(\mathcal{F}') := \{\mathcal{R}(\varphi) \mid \varphi \in \mathcal{F}'\}$$

und

$$K(\mathcal{F}) := \begin{cases} \{D(c) \mid c \text{ tritt in } \mathcal{F} \text{ auf}\}, & \text{falls es eine Konstante gibt, die in } \mathcal{F} \text{ auftritt} \\ \{D(a) \mid \text{für eine Konstante } a \in \Sigma\} & \text{sonst.} \end{cases}$$

□

Damit kann der bereits angekündigte Satz über die Ausdrucksstärke der PRQ-Formeln formuliert und bewiesen werden.

Satz 4.24 (Ausdrucksstärke der PRQ-Formeln) *Sei Σ die Signatur der betrachteten Sprache der ersten Stufe, D ein einstelliges Relationssymbol, das nicht in Σ enthalten ist und Σ' die Vereinigung von Σ und $\{D\}$. Dann existiert für jede endliche Menge $\mathcal{F}$ von geschlossenen Formeln der ersten Stufe über Σ eine endliche Menge $PRQ(\mathcal{F})$ von geschlossenen PRQ-Formeln über Σ' so daß gilt:*

1. *Ist $(\mathcal{D}, m)$ ein Σ-Modell von $\mathcal{F}$ mit der Domäne $\mathcal{D}$ und der Zuweisungsfunktion m, und ist m' folgende über Σ' definierte Abbildung:*

$$m'(s) := \begin{cases} m(s), & falls \quad s \neq D \\ \mathcal{D}, & falls \quad s = D \end{cases}$$

   dann ist $(\mathcal{D}, m')$ ein Σ'-Modell von $PRQ(\mathcal{F})$.

2. *Ist $(\mathcal{D}', m')$ ein Σ'-Modell von $PRQ(\mathcal{F})$, dann existiert ein Σ-Redukt $(\mathcal{D}, m)$ von $(\mathcal{D}', m')$, so daß $(\mathcal{D}, m)$ ein Modell von $\mathcal{F}$ ist.*

Beweis: Sei PRQ($\mathcal{F}$) die zu $\mathcal{F}$ gehörende Menge von geschlossenen PRQ-Formeln wie sie in Definition 4.23 definiert wurde. Es bleibt also zu zeigen, daß für diese Menge PRQ($\mathcal{F}$) die Aussagen von Satz 4.24 gelten.

Zu 1.: Nach Voraussetzung ist $(\mathcal{D}, m)$ ein Σ-Modell von $\mathcal{F}$ und da $\mathcal{F}'$ logisch äquivalent zu $\mathcal{F}$ ist, auch von $\mathcal{F}'$. Wir zeigen durch Induktion über den Aufbau von $\varphi \in \mathcal{F}'$, daß folgendes gilt:

Ist $(\mathcal{D}, m)$ ein Σ-Modell von φ, so ist $(\mathcal{D}, m')$ ein Σ'-Modell von $\mathcal{R}(\varphi)$.

Enthält φ keine Quantoren, so gilt die Behauptung offensichtlich, da $\mathcal{R}(\varphi) = \varphi$. Wegen $m'(D) = \mathcal{D}$ gilt die Behauptung auch für $\varphi = \forall x\psi$ und $\varphi = \exists x\psi$.

Wegen $m'(D) = \mathcal{D}$ ist $(\mathcal{D}, m')$ auch ein Modell von $K(\mathcal{F})$ und damit von PRQ($\mathcal{F}$).

Zu 2.: Sei nun $(\mathcal{D}', m')$ ein Σ'-Modell von PRQ($\mathcal{F}$). Wir definieren ein Σ-Redukt $(\mathcal{D}, m)$ von $(\mathcal{D}', m')$ durch Festlegung der Domäne $\mathcal{D} := m'(D)$. Da nach Voraussetzung $(\mathcal{D}', m')$ ein Modell von PRQ($\mathcal{F}$), also insbesondere von $K(\mathcal{F})$ ist, ist die Domäne $\mathcal{D}$ nicht leer. Für die Definition der Zuweisungsfunktion m können wir also ein Element d aus $\mathcal{D}$ wählen, um die Zuweisungsfunktion m auch auf denjenigen Konstanten $c \in \Sigma$ zu definieren, für die $m'(c) \notin \mathcal{D}$ (siehe Definition 2.26). Nach Lemma 2.27 ist $(\mathcal{D}, m)$ eine Σ-Interpretation. Es bleibt also noch zu zeigen, daß $(\mathcal{D}, m)$ ein Σ-Modell von $\mathcal{F}$ ist. Wir zeigen durch Induktion über den Aufbau einer Formel $\varphi \in \mathcal{F}'$, daß $(\mathcal{D}, m)$ ein Σ-Modell von $\mathcal{F}'$ ist. Aufgrund der logischen Äquivalenz von $\mathcal{F}$ und $\mathcal{F}'$ folgt die Behauptung dann sofort.

1. Sei φ ein Atom. Nach Definition von $\mathcal{R}$ tritt das Relationssymbol D in φ nicht auf.

(i) φ sei ein Grundatom. Nach Voraussetzung ist $(\mathcal{D}', m')$ ein Σ'-Modell von PRQ($\mathcal{F}$), also einmal von $\mathcal{R}(\varphi)$ und wegen $\mathcal{R}(\varphi) = \varphi$ auch von φ und zum zweiten auch von $K(\mathcal{F})$. Aufgrund des letzteren ist für jede Konstante c in φ die Interpretation $m'(c)$ in der Relation $m'(D)$ enthalten. Nach Definition des Σ-Redukts $(\mathcal{D}, m)$ ist dann $m(c) = m'(c)$. Da weiter für jedes n-stellige Relationssymbol die Zuweisungsfunktion durch $m(R) = m'(R) \cap \mathcal{D}^n$ definiert ist und da $(\mathcal{D}', m')$ ein Modell von φ ist, erhalten wir, daß auch $(\mathcal{D}, m)$ ein Modell von φ ist.

(ii) φ enthalte freie Variablen. In diesem Fall gilt die Behauptung nur dann, wenn die freien Variablen durch Elemente gebunden werden, die in der Relation $m'(D)$ enthalten sind. Deshalb wird hier und im folgenden eine Variablenbelegung β mit $\beta(x) \in m'(D)$ und $(\mathcal{D}', m')_\beta \models \varphi$ angenommen.

Da in $\mathcal{F}'$ aber nur geschlossene Formeln enthalten sind und die Bereichsbeschränkung gerade durch die Relation D erzwungen wurde, ist obige Bedingung keine Einschränkung. Die Argumentation verläuft dann wie in (i). Es sei darauf hingewiesen, daß auf Fall (ii) nicht verzichtet werden kann, auch wenn $\mathcal{F}'$ nur aus geschlossenen Formeln besteht, da sonst der Induktionsanfang nicht vollständig behandelt wäre.

2. Sei φ eine Konjunktion oder Disjunktion. Die Behauptung folgt direkt unter Benutzung der Induktionsvoraussetzung. Das gleiche gilt für $\varphi = \neg\psi = \psi \to \bot$.

3. Sei $\varphi = \forall x\psi$. Nach Definition 4.23 ist $\mathcal{R}(\varphi) = \forall x(D(x) \to \mathcal{R}(\psi))$ und nach Voraussetzung ist $(\mathcal{D}', m')$ ein Σ'-Modell von $\mathcal{R}(\varphi)$. Da φ eine geschlossene Formel ist

und aufgrund der Definition der Transformation $\mathcal{R}$ treten in $\mathcal{R}(\psi)$ außer x keine freien Variablen auf. Also ist $(\mathcal{D}', m')$ ein Σ'-Modell von $\mathcal{R}(\psi)$ für jede Variablenbelegung β mit $\beta(x) \in m'(D)$. Nach Induktionsvoraussetzung ist dann $(\mathcal{D}, m)$ ein Σ-Modell von ψ für jede Variablenbelegung β mit $\beta(x) \in m'(D)$. Da nach Definition des Σ-Redukts $(\mathcal{D}, m)$ die Domäne $\mathcal{D} = m'(D)$, folgt, daß $(\mathcal{D}, m)$ auch ein Σ-Modell von φ ist.

4. Sei $\varphi = \exists x\psi$. Die Transformation von φ ist nach Definition 4.23 $\mathcal{R}(\varphi) = \exists x(D(x) \wedge \mathcal{R}(\psi))$. Mit der gleichen Argumentation wie in 3. folgt die Behauptung. ∎

Bemerkung 4.25 Es ist offensichtlich, daß die oben beschriebene Transformation hinsichtlich Effizienz sicherlich verbesserungswürdig ist. Es wäre vielmehr wünschenswert, eine Transformation zu haben, die bei den gegebenen Formeln so wenig wie möglich verändert. Für die Formel $F = \forall x \forall y(p(x) \rightarrow r(x, y))$ würde etwa obige Transformation

$$\mathcal{R}(F) = \forall x(D(x) \rightarrow \forall y(D(y) \rightarrow (p(x) \rightarrow r(x, y))))$$

ergeben. Dabei wird für die Variable x unnötigerweise ein Bereich erzeugt. Durch eine geschickte Transformation könnte die Formel F in die wesentlich einfachere PRQ-Formel $F' = \forall x \forall y(p(x) \wedge D(y) \rightarrow r(x, y))$ überführt werden. Dabei würde die Gültigkeit von Satz 4.24 nicht angetastet. Andererseits ist die Idee, die hinter dem Entwurf der PRQ-Syntax steht, eine Syntax zu bieten, die von vornherein schon soviel zuläßt, daß Spezifikationen direkt in dieser Syntax formuliert werden können. Dann aber dient die oben beschriebene Transformation ausschließlich dem Nachweis, daß es grundsätzlich möglich ist, beliebige Formeln erster Stufe in PRQ-Syntax so zu transformieren, daß die Modelle erhalten bleiben.

In Abschnitt 2.2 haben wir gesehen, daß die Betrachtung von Herbrandinterpretationen keine Einschränkung hinsichtlich der Erfüllbarkeit von universellen und gleichheitsfreien Sätzen, insbesondere Klauseln, darstellt(siehe Satz 2.31). Dies gilt jedoch nicht, wenn es um die Erfüllbarkeit im Endlichen geht, wie Beispiel 2.32 verdeutlichte.

Das folgende Korollar zeigt, daß dagegen die Betrachtung von Terminterpretationen keine Einschränkung im Hinblick auf die Erfüllbarkeit im Endlichen und damit die Generierung endlicher Modelle bedeutet.

Korollar 4.26 *Für jede endliche Menge $\mathcal{F}$ von geschlossenen Formeln der ersten Stufe existiert eine endliche Menge $PRQ(\mathcal{F})$ von geschlossenen PRQ-Formeln, so daß folgendes gilt:*

> *$\mathcal{F}$ ist endlich erfüllbar genau dann, wenn $PRQ(\mathcal{F})$ ein endliches Termmodell besitzt.*

Bemerkung 4.27 Dieses Korollar ist offensichtlich nicht auf Herbrandinterpretationen übertragbar, da jede Herbrandinterpretation über einer Signatur mit unendlich vielen Konstanten eine unendliche Domäne besitzt.

Beweis: Mit Satz 4.24 folgt direkt, daß für jede endliche Menge $\mathcal{F}$ geschlossener Formeln der ersten Stufe eine endliche Menge PRQ($\mathcal{F}$) von geschlossenen PRQ-Formeln existiert, so daß

$\mathcal{F}$ endlich erfüllbar ist genau dann, wenn PRQ($\mathcal{F}$) endlich erfüllbar ist.

Um Korollar 4.26 zu beweisen, muß noch gezeigt werden, daß gilt:

PRQ($\mathcal{F}$) ist endlich erfüllbar genau dann, wenn PRQ($\mathcal{F}$) ein endliches Termmodell besitzt.

Wenn PRQ($\mathcal{F}$) ein endliches Termmodell besitzt, ist PRQ($\mathcal{F}$) trivialerweise endlich erfüllbar. Der Beweis der anderen Richtung der Behauptung ist dagegen etwas aufwendiger.

Sei also PRQ($\mathcal{F}$) endlich erfüllbar und sei die Interpretation $\mathcal{I} = (\mathcal{D}, m)$ ein endliches Modell von PRQ($\mathcal{F}$). Dann erhalten wir durch die unten beschriebene Konstruktion eine endliche Terminterpretation $\mathcal{T}(\mathcal{G})$. In einem zweiten Schritt zeigen wir dann durch Induktion über den Aufbau von PRQ-Formeln, daß $\mathcal{T}(\mathcal{G})$ ein Modell von PRQ($\mathcal{F}$) ist.

Beginnen wir mit der Konstruktion einer Terminterpretation $\mathcal{T}(\mathcal{G})$. Da unsere Sprache nach Definition über ω viele Konstanten verfügt und nur endlich viele Konstanten in der endlichen Formelmenge $\mathcal{F}$ vorkommen, können wir eine bijektive Abbildung g von der endlichen Domäne $\mathcal{D}$ von $\mathcal{I}$ in eine endliche Menge $\underline{\mathcal{D}}$ von neuen Konstanten, die nicht in $\mathcal{F}$ auftreten, definieren:

$$\begin{array}{rccc} g & : & \mathcal{D} & \rightarrow & \underline{\mathcal{D}} \\ & & d & \mapsto & \underline{d} \end{array}$$

Betrachten wir nun die endlich vielen Konstanten $c_1, \ldots, c_l$, die in $\mathcal{F}$ auftreten. Für jede dieser Konstanten c_i gibt es ein Element $d_i \in \mathcal{D}$, so daß $m(c_i) = d_i$. Die Funktion g bildet jedes dieser Elemente d_i auf eine Konstante $\underline{d_i}$ ab. Nun ersetzen wir in der Menge $\underline{\mathcal{D}}$ genau diese Konstanten $\underline{d_i}$ durch c_i, für die $m(c_i) = g^{-1}(\underline{d_i})$ gilt. So erhalten wir eine neue Menge von Konstanten, die wir mit $\underline{\mathcal{C}}$ bezeichnen wollen. Bei dieser Ersetzung kann es passieren, daß eine Konstante $\underline{d}$ durch mehrere Konstanten c ersetzt wird, nämlich dann, wenn durch die Abbildung m verschiedene Konstanten durch das gleiche Element aus $\mathcal{D}$ interpretiert werden. Da in $\mathcal{F}$ aber nur endlich viele Konstanten auftreten, bleibt die neue Menge $\underline{\mathcal{C}}$ auf jeden Fall endlich. Jetzt definieren wir eine Relation $f : \mathcal{D} \rightarrow \underline{\mathcal{C}}$ durch:

$$f(d) := \begin{cases} g(d), & \text{falls } g(d) \in \underline{\mathcal{C}} \\ c \text{ für } d = m(c) \text{ und } c \in \underline{\mathcal{C}} & \text{sonst.} \end{cases}$$

Die Relation f ist keine Funktion, da einem Element $d \in \mathcal{D}$ mittels f möglicherweise mehrere Konstanten c mit $m(c) = d$ zugeordnet werden. Dagegen ist die Relation f^{-1} eine surjektive Funktion. Für jedes Element $d \in \mathcal{D}$ gibt es mindestens eine Konstante $c \in \underline{\mathcal{C}}$, so daß $f^{-1}(c) = d$ gilt. Dies ist für unsere Belange wesentlich.

Sei nun $R(\mathcal{F})$ die Menge der Relationssymbole, die in $\mathcal{F}$ auftreten. Um die Menge $\mathcal{G}$ der in der Terminterpretation $\mathcal{T}(\mathcal{G})$ erfüllten Grundatome zu definieren, betrachten wir

nun alle Grundatome über $R(\mathcal{F}) \cup \underline{\mathcal{C}}$ und definieren für jedes k-stellige Relationssymbol $r \in R(\mathcal{F})$ und jedes k-Tupel $c_1, \ldots, c_k$ von Konstanten in $\underline{\mathcal{C}}$

$$r(c_1 \ldots c_k) \in \mathcal{G} \quad :\text{gdw} \quad (d_1, \ldots, d_k) \in m(r) \text{ und } d_i = f^{-1}(c_i) \text{ für } d_i \in \mathcal{D} \text{ und } 1 \leq i \leq k$$

Damit ist die Menge $\mathcal{G}$ definiert. $\mathcal{G}$ ist endlich, da sowohl $R(\mathcal{F})$ als auch $\underline{\mathcal{C}}$ endlich sind. Nach Definition besteht die Domäne einer Terminterpretation aus den Konstanten, die in $\mathcal{G}$ auftreten und einer zusätzlichen Konstante, etwa c_0. Diese Konstante c_0 ist notwendig, um alle Konstanten in $\underline{\mathcal{C}}$ zu interpretieren, die nicht in $\mathcal{G}$ auftreten.

Es bleibt nun noch der zweite Schritt des Beweises: Wir zeigen durch Induktion über den Aufbau einer PRQ-Formel φ, daß die so definierte endliche Terminterpretation $\mathcal{T}(\mathcal{G})$ ein Modell von PRQ($\mathcal{F}$) ist.

Sei φ ein Atom: Es werden zwei Fälle unterschieden.
1. Sei φ ein Grundatom. Nach Voraussetzung ist $\mathcal{I}$ ein Modell von φ. Also ist nach Definition von $\mathcal{G}$ das Atom φ ein Element aus $\mathcal{G}$ und damit ist $\mathcal{T}(\mathcal{G})$ ein Termmodell von φ.
2. Seien $x_1 \ldots x_n$ die freien Variablen in φ. Da nach Voraussetzung $\mathcal{I}$ ein Modell von φ ist, gibt es eine Variablenbelegung β bezüglich $\mathcal{D}$ und Elemente $d_1, \ldots, d_n$ in der Domäne $\mathcal{D}$ von $\mathcal{I}$, so daß $\beta(x_i) = d_i$ und $\mathcal{I}_\beta \models \varphi$. Nach Definition von $\underline{\mathcal{C}}$ gibt es für $d_1, \ldots, d_n$ Konstanten $c_1, \ldots, c_n$ mit $f^{-1}(c_i) = d_i$. Für jedes dieser Konstantentupel $c_1, \ldots, c_n$ kann eine Substitution $\sigma = \{c_1/x_1, \ldots, c_n/x_n\}$ definiert werden. Nach Definition von $\mathcal{G}$ ist jede Formel $\varphi\sigma$ ein Element aus $\mathcal{G}$. Also ist $\mathcal{T}(\mathcal{G})_{\underline{\beta}\frac{c_1 \ldots c_n}{x_1 \ldots x_n}}$ ein Termmodell von φ für eine beliebige Variablenbelegung $\underline{\beta}$ und für jedes Konstantentupel $(c_1, \ldots, c_n)$ mit oben beschriebener Eigenschaft.

Für $\varphi = \top$ ist nichts zu zeigen. Da die Formelmenge PRQ($\mathcal{F}$) erfüllbar ist, ist die Formel $\bot$ nicht in PRQ($\mathcal{F}$).

Sei $\varphi = \varphi_1 \wedge \varphi_2$ bzw. $\varphi = \varphi_1 \vee \varphi_2$: Die Behauptung ergibt sich unmittelbar aus der Induktionsvoraussetzung.

Sei $\varphi = \exists x\ \psi$. Nach Voraussetzung gibt es eine Variablenbelegung β bezüglich $\mathcal{D}$ und ein Element $d \in \mathcal{D}$, so daß $\mathcal{I}_{\beta\frac{d}{x}} \models \psi$ gilt. Also gibt es nach Induktionsvoraussetzung eine Variablenbelegung $\underline{\beta}$ und ein Element c aus der Domäne von $\mathcal{T}(\mathcal{G})$, so daß $\mathcal{T}(\mathcal{G})_{\underline{\beta}\frac{c}{x}} \models \psi$ gilt. Damit gilt auch $\mathcal{T}(\mathcal{G}) \models \varphi$.

Sei $\varphi = \forall x_1 \ldots x_k\ (B \to F)$: Dieser Fall ist wesentlich aufwendiger zu beweisen. Nach Voraussetzung ist $\mathcal{I}$ ein Modell von $\forall x_1 \ldots x_k\ (B \to F)$. Also ist $\mathcal{I}$ für jede Variablenbelegung β bezüglich $\mathcal{D}$ ein Modell von $B \to F$. Es muß gezeigt werden, daß auch $\mathcal{T}(\mathcal{G})_{\underline{\beta}}$ für jede Variablenbelegung $\underline{\beta}$ bezüglich der Domäne von $\mathcal{T}(\mathcal{G})$ ein Modell von $B \to F$ ist. Sei $\underline{\beta}$ eine beliebige feste Variablenbelegung bezüglich der Domäne von $\mathcal{T}(\mathcal{G})$. Es sind zwei Fälle

zu unterscheiden:

Fall 1: $\mathcal{T}(\mathcal{G})_{\underline{\beta}} \not\models B$. Dann ist $\mathcal{T}(\mathcal{G})_{\underline{\beta}}$ ein Modell von $B \to F$.

Fall 2: $\mathcal{T}(\mathcal{G})_{\underline{\beta}} \models B$. Sei $\underline{\beta}(x_i) = c_i$ für $1 \leq i \leq k$. Dann gilt folgende Behauptung:

(∗) Für jede Variablenbelegung β bezüglich $\mathcal{D}$ mit $\beta = f^{-1} \circ \underline{\beta}$ ist $\mathcal{I}_\beta$ ein Modell von B.

Insbesondere ist die Belegung β eindeutig.

Beweis von (∗) durch Induktion über den Aufbau einer positiven Bedingung (damit ist dann auch der „leere“ Quantor, also die Implikation, abgedeckt, siehe Abschnitt 2.6):

Sei B ein Atom. Behauptung (∗) folgt aus der Definition der Menge $\mathcal{G}$.

Sei $B = B_1 \wedge B_2$ (bzw. $B = B_1 \vee B_2$). Behauptung (∗) folgt zusammen mit der Eindeutigkeit der Variablenbelegung β aus der Induktionsvoraussetzung.

Sei $B = \exists y B_1$. Nach Voraussetzung ist $\mathcal{T}(\mathcal{G})_{\underline{\beta}}$ ein Modell von $\exists y B_1$. Dann gibt es ein Element c aus der Domäne von $\mathcal{T}(\mathcal{G})$, so daß $\mathcal{T}(\mathcal{G})_{\underline{\beta}\frac{c}{y}} \models B_1$. Sei $d \in \mathcal{D}$ mit $d = f^{-1}(c)$. Das Element d ist eindeutig. Also folgt nach Induktionsvoraussetzung, daß $\mathcal{I}_{\beta\frac{d}{y}} \models B_1$ gilt und damit ist $\mathcal{I}_{\beta\frac{d}{y}}$ ein Modell von $\exists y B_1$.

Jetzt kann mit (∗) gezeigt werden, daß $\mathcal{T}(\mathcal{G})_{\underline{\beta}}$ ein Modell von $B \to F$ ist: Nach Voraussetzung ist $\mathcal{T}(\mathcal{G})_{\underline{\beta}}$ ein Modell von B. Mit (∗) gilt $\mathcal{I}_\beta \models B$ für die Variablenbelegung $\beta = f^{-1} \circ \underline{\beta}$. Nach Voraussetzung ist $\mathcal{I}_{\beta'}$ ein Modell von $B \to F$ für jede Variablenbelegung β' bezüglich $\mathcal{D}$, insbesondere für die Variablenbelegung β. Wegen $\mathcal{I}_\beta \models B$ ist also $\mathcal{I}_\beta$ ein Modell von F. Nach Induktionsvoraussetzung (aus der äußeren Induktion über den Aufbau der PRQ-Formeln) ist dann $\mathcal{T}(\mathcal{G})_{\underline{\beta'}}$ ein Modell von F für jede Variablenbelegung $\underline{\beta'}$ bezüglich der Domäne von $\mathcal{T}(\mathcal{G})$ mit $f^{-1}(\underline{\beta'}(x)) = \beta(x)$. Insbesondere gilt dies für die oben festgelegte Variablenbelegung $\underline{\beta}$ aufgrund der Konstruktion der Variablenbelegung β. Deshalb ist auch $\mathcal{T}(\mathcal{G})_{\underline{\beta}}$ ein Modell von F und damit ein Modell von $B \to F$.

Damit ist $\mathcal{T}(\mathcal{G})$ ein Modell von $\varphi = \forall x_1 \ldots x_n\ (B \to F)$. ∎

4.4 Extended-Positive-Tableaux

Mit der Definition der PRQ-Syntax ist nun alles bereitgestellt, um die Extended-Positive-Tableaux-Methode, kurz EP-Tableaux-Methode, zu formulieren. Wie bereits erwähnt, ist die EP-Tableaux-Methode eine Erweiterung der PUHR-Tableaux-Methode, die in Kapitel 3 vorgestellt wurde. In Abschnitt 4.1 wurde eingehend untersucht, welche Überlegungen zu einer Erweiterung der PUHR-Tableaux-Methode führen. Eine erste Erweiterung wurde

in Abschnitt 4.2 definiert, gleichzeitig aber ihre Unzulänglichkeit für die Generierung endlicher Modelle erkannt. Dies motivierte eine explizite Behandlung von Existenzquantoren und damit die Definition der PRQ-Syntax. Die EP-Tableaux-Methode beinhaltet daher eine weitere Expansionsregel für existentiell quantifizierte Teilformeln [BM87, Hin88, Lor94]. Diese Expansionsregel führt im Gegensatz zu der herkömmlichen δ-Regel [Smu68, Fit90] keine „Laufzeit-Skolemisierung" durch, d.h. die Behandlung einer existentiell quantifizierten Teilformel führt nicht notwendigerweise zur Einführung einer neuen Konstante. Diese Tatsache ist wesentlich im Hinblick auf die Generierung endlicher Modelle, wie wir in Abschnitt 4.2 bereits an Beispiel 4.13 gesehen haben. Im folgenden werden nur geschlossene PRQ-Formeln betrachtet.

Definition 4.28 (EP-Tableaux-Expansionsregeln)

$\exists$-Regel:

$$\frac{\exists x E(x)}{E\{c_1/x\} \mid \ldots \mid E\{c_k/x\} \mid E\{c_{neu}/x\}}$$

wobei $\{c_1, \ldots, c_k\}$ die Menge aller Konstanten ist, die im expandierten Knoten vorkommen und c_{neu} eine von c_i für $i = 1, \ldots, k$ verschiedene Konstante ist.

Erweiterte PUHR-Regel:

$$\frac{\forall \bar{x}(B(\bar{x}) \rightarrow F)}{F\sigma}$$

$\vee$-Regel:

$$\frac{E_1 \vee E_2}{E_1 \mid E_2}$$

$\wedge$-Regel:

$$\frac{E_1 \wedge E_2}{\begin{array}{c} E_1 \\ E_2 \end{array}}$$

wobei $\sigma = \{\bar{c}/\bar{x}\}$ eine Grundsubstitution ist und $B\sigma$ durch die Terminterpretation erfüllt wird, die durch die Menge der Grundatome, die im expandierten Knoten vorkommen, bestimmt wird. □

Da die Notation $\forall\epsilon(B \rightarrow F)$, wobei ϵ das leere Tupel bezeichne, anstelle der Implikation $B \rightarrow F$ zugelassen ist, werden durch die erweiterte PUHR-Regel sowohl universell quantifizierte PRQ-Formeln als auch Implikationen behandelt. Im folgenden wird die erweiterte PUHR-Regel abkürzend einfach mit PUHR-Regel bezeichnet.

Definition 4.29 (EP-Tableaux) EP-Tableaux für eine Menge $\mathcal{S}$ von PRQ-Formeln sind Bäume, deren Knoten Mengen geschlossener Formeln sind. EP-Tableaux werden folgendermaßen induktiv definiert:

1. Der Baum, der nur aus dem einzigen Knoten $\mathcal{S}$ besteht, ist ein EP-Tableau für $\mathcal{S}$.

2. Sei T ein EP-Tableau für $\mathcal{S}$, sei L ein Blatt von T und F eine Formel in L, die von der Interpretation $\mathcal{T}(Atoms(L))$ nicht erfüllt wird. Werden an T ein oder mehrere Kinderknoten angehängt, die durch Anwendung der passenden Expansionsregel auf

F entstehen, so ist der entstandene Baum wieder ein EP-Tableau für $\mathcal{S}$. Jeder Kinderknoten enthält dabei alle Formeln aus L sowie eine bzw. im Falle der Konjunktion zwei neue Formeln gemäß der Expansionsregel in Definition 4.28. Bei einer Konjunktion wird ein Konjunktionsglied nur dann angefügt, falls es nicht von $\mathcal{T}(Atoms(L))$ erfüllt wird.

Ein Zweig eines EP-Tableau heißt *geschlossen*, falls er $\bot$ enthält. Sonst heißt er *offen*. Ein EP-Tableau ist *offen*, falls es mindestens einen offenen Zweig enthält. Sonst ist es *geschlossen*. Ist $\mathcal{B}$ ein Zweig eines EP-Tableau, so bezeichne $\cup\mathcal{B}$ die Vereinigung aller Knoten in $\mathcal{B}$. Ein EP-Tableau ist *erfüllbar*, falls es einen Zweig $\mathcal{B}$ besitzt, so daß $\cup\mathcal{B}$ erfüllbar ist. □

Bemerkung 4.30 Wenn im folgenden von der „Anwendung einer Expansionsregel auf einen Knoten“ oder „eine Formel in einem Knoten“ gesprochen wird, ist damit das Anhängen eines oder mehrerer Kinderknoten gemeint. Wichtig ist dabei, daß die Kinderknoten sowohl *alle* Formeln, die der Elternknoten enthält sowie eine bzw. im Falle der Konjunktion möglicherweise auch zwei neue Formeln enthalten und nicht, wie die Notation von Definition 4.28 möglicherweise suggeriert, nur die neu hinzugekommenen Formeln.

Bemerkung 4.31 Im folgenden werden ausschließlich *endliche* Mengen von PRQ-Formeln betrachtet, auch wenn dies nicht explizit erwähnt wird. Dies ist im Gegensatz zur PUHR-Tableaux-Methode bei der EP-Tableaux-Methode deshalb sinnvoll, weil in einer unendlichen Menge von PRQ-Formeln möglicherweise unendlich viele Konstanten auftreten. Bei einer Anwendung der $\exists$-Regel würde dies zu einer unendlichen Verzweigung des EP-Tableau führen, was nicht erwünscht ist. Natürlich wäre es möglich, unendliche Formelmengen, in denen nur endlich viele Konstanten auftreten, zuzulassen. Da aber die EP-Tableaux-Methode im Hinblick auf eine Prozedur entworfen wurde, scheint es nicht sinnvoll, unendliche Formelmengen zuzulassen.

Lemma 4.32 *Die Anwendung einer EP-Expansionsregel auf ein EP-Tableau, das nur geschlossene PRQ-Formeln enthält führt wieder zu einem EP-Tableau, das nur geschlossene PRQ-Formeln enthält.*

Beweis: Folgt direkt aus der Definition der EP-Expansionsregeln. ∎

In diesem Zusammenhang stellt sich auch die Frage der Terminierung des Verfahrens. Das Ziel ist, ein Modell für eine gegebene Formelmenge zu generieren. Dann aber muß sichergestellt werden, daß keine weiteren Expansionsregeln angewendet werden, sobald ein Modell gefunden wurde. Dies leistet die Bedingung in Definition 4.29, daß eine Expansionsregel auf eine Formel nur dann angewendet werden darf, falls diese Formel noch nicht von der Terminterpretation, die der Zweig repräsentiert, erfüllt wird. Trotzdem ist bei komplexeren Formeln z.B. folgendes denkbar: Eine Konjunktion von zwei nichtatomaren Formeln wird mittels der $\wedge$-Regel behandelt. Da die Konjunktionsglieder nicht-atomare Formeln sind, wird die ursprüngliche Formel auch nach der Anwendung der $\wedge$-Regel nicht

von der durch den Zweig repräsentierten Terminterpretation erfüllt. Also kann die $\wedge$-Regel wieder auf dieselbe Formel angewendet werden. Auf diese Weise kann die Generierung eines Modells möglicherweise unendlich lange verschoben werden. Um diese Schwierigkeiten nicht auftreten zu lassen, wird in Abschnitt 4.5 der Begriff der *Fairness* eines EP-Tableau eingeführt werden.

Die folgenden Beispiele stellen zwei EP-Tableaux dar. Zur Verbesserung der Lesbarkeit werden die Knoten des Baumes (entgegen der Definition 4.29) nicht mit der Formelmenge, die der Knoten enthält, beschriftet, sondern ausschließlich mit den durch die Anwendung einer Expansionsregel neu hinzugekommenen Formeln. Ein jeder Knoten des Tableau enthält dann alle Formeln, die in der Abbildung in dem Zweig, der von der Wurzel bis zu diesem Knoten führt, sichtbar sind.

Beispiel 4.33 Betrachten wir die Formelmenge $\mathcal{S}_1 = \{p(a),\ \forall x(p(x) \rightarrow r(x) \vee \exists y\ q(x,y))\}$. Abbildung 4.1 zeigt ein EP Tableau für $\mathcal{S}_1$.[5] Die Wurzel des Tableau besteht aus der Menge $\mathcal{S}_1$. Im ersten Schritt wird die PUHR-Regel auf die Formel $\forall x(p(x) \rightarrow r(x) \vee \exists y q(x,y))$ angewendet. Da die Terminterpretation $\mathcal{T}(p(a))$ die Teilformel $p(x)\{a/x\}$ erfüllt, enthält der zweite Knoten zusätzlich zu $\mathcal{S}_1$ die Formel $r(a) \vee \exists y\ q(a,y)$. Die Anwendung der $\vee$-Regel auf diese Teilformel führt zu einer Verzweigung. Im linken Zweig wird die Teilformel $r(a)$ hinzugefügt, im rechten die Teilformel $\exists y\ q(a,y)$. Im linken Zweig des Tableau kann nun keine Regel mehr angewendet werden. Der Zweig ist offen und in Abschnitt 4.5 wird gezeigt werden, daß die Terminterpretation $\mathcal{T}(\{p(a), r(a)\})$, die durch die Grundatome, die im Zweig auftreten, repräsentiert wird, ein Modell von $\mathcal{S}_1$ ist. Im rechten Zweig des Tableau kann nun die $\exists$-Regel auf die Teilformel $\exists y\ q(a,y)$ angewendet werden. Die einzige Konstante, die im Zweig auftritt, ist die Konstante a. Deshalb gibt es nur zwei Kinderknoten, den linken, der durch Ersetzen der Variablen y durch die Konstante a entsteht, und den rechten. Hier wird eine neue Konstante c_{neu} eingeführt und y durch sie ersetzt. Damit können auch in diesen beiden Zweigen keine Regeln mehr angewendet werden. Beide Zweige sind offen und repräsentieren daher (siehe Abschnitt 4.5, Lemma 4.38) je ein Modell von $\mathcal{S}_1$. Der mittlere Zweig repräsentiert das Modell $\mathcal{T}(\{p(a), q(a,a)\})$ und der rechte das Modell $\mathcal{T}(\{p(a), q(a, c_{neu})\})$.

Im folgenden Beispiel wird eine Formelmenge betrachtet, die zu einem unendlichen Baum führt.

Beispiel 4.34 Sei $\mathcal{S}_2 = \{p(c_0),\ \forall x(p(x) \rightarrow \exists y\ q(x,y)),\ \forall x \forall y(q(x,y) \rightarrow p(x) \wedge p(y))\}$[6] Abbildung 4.2 zeigt ein EP-Tableau für $\mathcal{S}_2$. Die Wurzel des Baumes besteht per Definition aus $\mathcal{S}_2$. Der zweite Knoten entsteht durch Anwendung der PUHR-Regel auf die Formel $\forall x(p(x) \rightarrow \exists y\ q(x,y))$ und führt zu $\exists y\ q(c_0,y))$. Auf diese Teilformel wird nun die $\exists$-Regel angewendet. Da in dem einzigen Zweig des Tableau nur die Konstante c_0 existiert, entstehen zwei Kinderknoten, wobei die existentiell quantifizierte Variable y einmal durch die Konstante c_0 und einmal durch eine neu eingeführte Konstante c_1 ersetzt wird. In dem

[5] Streng nach Definition 4.15 müßte $\forall x(p(x) \rightarrow r(x) \vee \exists y(q(x,y)\ \underline{\wedge \top}))$ geschrieben werden.

[6] Nach Definition 4.15 müsste $\forall x(p(x) \rightarrow \exists y\ (q(x,y)\underline{\wedge \top}))$ geschrieben werden.

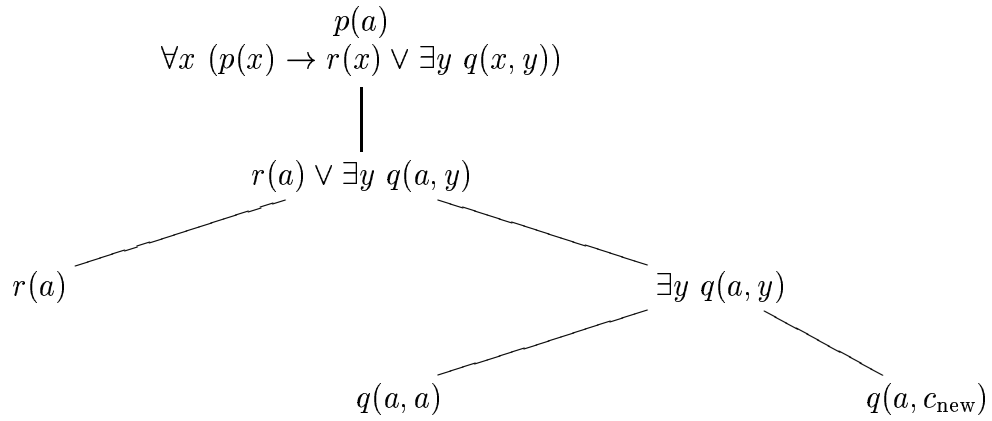


Abbildung 4.1: Ein EP-Tableau für Beispiel 4.33

Zweig, in dem die „alte“ Konstante c_0 wiederverwendet wurde, kann keine Expansionsregel mehr angewendet werden. Im anderen Zweig kann die PUHR-Regel auf die Formel $\forall x \forall y(q(x,y) \to p(x) \wedge p(y))\}$ angewendet werden. Die durch den Zweig repräsentierte Terminterpretation $\mathcal{T}(\{p(c_0), q(c_0,c_1)\})$ erfüllt $q(c_0,c_1)$, aber nicht $p(c_0) \wedge p(c_1)$. Deshalb wird diese Teilformel in den Kinderknoten eingefügt. Im nächsten Schritt wird die $\wedge$-Regel angewendet. In diesem Fall wird nur das Konjunkt $p(c_1)$ neu eingefügt, da $p(c_0)$ schon Bestandteil der Wurzel und damit jedes Knotens ist. Aufgrund des neu erzeugten Atoms $p(c_1)$ ist nun die PUHR-Regel wieder auf die Formel $\forall x(p(x) \to \exists y\ q(x,y))$ anwendbar und führt auf die Teilformel $\exists y\ q(c_1,y))$, die in den Kinderknoten eingefügt wird. Die Behandlung dieser Teilformel mit der $\exists$-Regel führt nun zu drei Nachfolgerknoten, da in diesem Zweig nun zwei Konstanten, nämlich c_0 und c_1 zur Verfügung stehen. Eine dritte neue Konstante c_2 wird neu eingeführt. Im nächsten Schritt kann nur noch auf eines der drei Blätter des Tableau eine Expansionsregel angewendet werden, nämlich auf den rechten Zweig. Und es wird deutlich, daß sich ein gewisses Muster bei der Expansion dieses Tableau wiederholt.

4.5 Widerlegungskorrektheit und -vollständigkeit

In diesem Abschnitt werden die Widerlegungskorrektheit und -vollständigkeit sowie die Modellkorrektheit der EP-Tableaux-Methode gezeigt. In den Beweisen werden im wesentlichen Standardtechniken verwendet [Fit90, BY96, Fur98].

Im folgenden bezeichne $\mathcal{S}$ eine Menge von geschlossenen PRQ-Formeln.

Lemma 4.35 *Die Anwendung einer Expansionsregel auf ein erfüllbares EP-Tableau führt wieder zu einem erfüllbaren EP-Tableau.*

Beweis: Sei T ein Tableau für eine Menge $\mathcal{S}$ von PRQ-Formeln, $\mathcal{B}$ ein endlicher Zweig von T, $L_{\mathcal{B}}$ das Blatt von $\mathcal{B}$ und $\mathcal{I} = \{\mathcal{D}, m\}$ ein Modell von $\cup\mathcal{B}$.

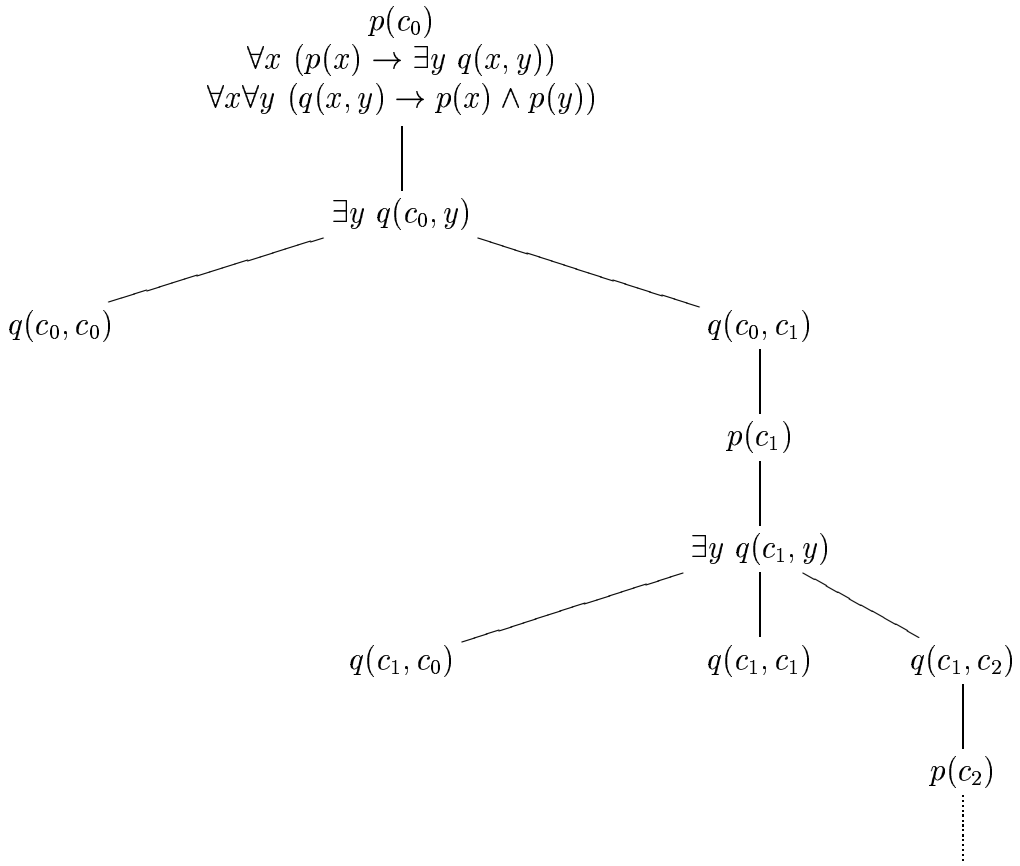


Abbildung 4.2: Ein EP-Tableau für Beispiel 4.34

Für jede EP-Expansionsregel wird nun gezeigt, daß ihre Anwendung auf $L_{\mathcal{B}}$ zu mindestens einem erfüllbaren Nachfolgerknoten, und damit zu einem erfüllbaren Zweig führt.
1. $\wedge$- und $\vee$-Regel:
Sei $E_1 \wedge E_2$ (bzw. $E_1 \vee E_2$) ein Element aus $\cup\mathcal{B}$ und sei $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$. Insbesondere ist dann $\mathcal{I}$ ein Modell von $E_1 \wedge E_2$ (bzw. von $E_1 \vee E_2$), d.h. ein Modell von E_1 und von E_2 (bzw. von E_1 oder von E_2). Nach Definition 4.29 enthält der Kinderknoten von $L_{\mathcal{B}}$ bei Anwendung der $\wedge$-Regel die Menge $\cup\mathcal{B} \cup \{E_1, E_2\}$. Damit ist $\mathcal{I}$ auch ein Modell des Zweiges, der durch Anwendung der $\wedge$-Regel entstanden ist, und das Tableau bleibt erfüllbar. Bei Anwendung der $\vee$-Regel auf $L_{\mathcal{B}}$ enthalten die Kinderknoten $\cup\mathcal{B} \cup \{E_1\}$ bzw. $\cup\mathcal{B} \cup \{E_2\}$. Also gibt es mindestens einen Kinderknoten, der von $\mathcal{I}$ erfüllt wird und das erhaltene Tableau bleibt erfüllbar.
2. $\exists$-Regel:
Sei $\exists x E(x)$ ein Element aus $\cup\mathcal{B}$ und sei $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$. Im folgenden wird zunächst eine Interpretation $\tilde{\mathcal{I}} = (\tilde{\mathcal{D}}, \tilde{m})$ definiert. Dann wird gezeigt, daß es mindestens einen Kinderknoten N von $L_{\mathcal{B}}$ gibt, so daß $\tilde{\mathcal{I}}$ ein Modell von $\cup\mathcal{B} \cup \{N\}$ ist.

Nach Voraussetzung ist $\mathcal{I} = (\mathcal{D}, m)$ ein Modell von $\exists x E(x)$, d.h. es gibt ein Element $d \in \mathcal{D}$, so daß $\mathcal{I}_{\beta\frac{d}{x}}$ für eine beliebige Variablenbelegung β ein Modell von $E(x)$ ist. Nach

Definition 4.29 enthalten die Kinderknoten von $L_{\mathcal{B}}$ bei Anwendung der $\exists$-Regel $\cup\mathcal{B} \cup \{E\{c_1/x\}\}$ oder ... oder $\cup\mathcal{B} \cup \{E\{c_n/x\}\}$ oder $\cup\mathcal{B} \cup \{E\{c_{neu}/x\}\}$, wobei $c_1, \ldots, c_n$ im Zweig auftretende Konstanten und c_{neu} eine neu eingeführte Konstante ist. Sei nun $\tilde{\mathcal{I}} = \{\tilde{\mathcal{D}}, \tilde{m}\}$ folgende Interpretation: $\tilde{\mathcal{D}} := \mathcal{D}$, für ein Relationssymbol r sei $\tilde{m}(r) := m(r)$ und für eine Konstante $c \neq c_{neu}$ sei $\tilde{m}(c) := m(c)$. Gibt es eine Konstante $c_i, i \in \{1, \ldots, n\}$ mit $\tilde{m}(c) = d$, dann wird c_{neu} durch ein beliebiges Element der Domäne $\tilde{\mathcal{D}}$ interpretiert. Falls nicht wird c_{neu} durch $d \in \tilde{\mathcal{D}}$ interpretiert. Da $\tilde{\mathcal{I}}$ sich von $\mathcal{I}$ nur durch die Interpretation der Konstante c_{neu} unterscheidet, ist $\tilde{\mathcal{I}}$ ein Modell von $\cup\mathcal{B}$. Nach Konstruktion von $\tilde{\mathcal{I}}$ ist $\tilde{\mathcal{I}}$ zudem ein Modell von $E(x)\{c_i/x\}$ mit $i \in \{1, \ldots, n, neu\}$ und $\tilde{m}(c_i) = d$. Damit gibt es einen Kinderknoten $N = \cup\mathcal{B} \cup \{E\{c_i/x\}\}$, so daß $\tilde{\mathcal{I}}$ ein Modell von $\cup\mathcal{B} \cup \{N\}$ ist, d.h. das Tableau, welches durch Anwendung der $\exists$-Regel entstanden ist, ist erfüllbar.
3. PUHR-Regel:
Sei $\forall\bar{x}(B(\bar{x}) \rightarrow F)$ ein Element aus $\cup\mathcal{B}$ und sei $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$. Damit die PUHR-Regel angewendet werden kann, muß es eine Grundsubstitution σ geben, derart daß $B\sigma$ von der durch den Zweig repräsentierten Terminterpretation $\mathcal{T}(Atoms(\cup\mathcal{B}))$ erfüllt wird und $F\sigma$ nicht. Nach Anwendung der PUHR-Regel besteht der Kinderknoten von $L_{\mathcal{B}}$ aus $\cup\mathcal{B} \cup \{F\sigma\}$. Nach Voraussetzung ist $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$. Also muß gezeigt werden, daß $\mathcal{I}$ auch ein Modell von $F\sigma$ ist. Dazu wird zunächst durch Induktion über den Aufbau einer positiven Bedingung folgende Behauptung gezeigt:

(*) Sei P eine positive Bedingung ohne freie Variablen und $\mathcal{T}(\mathcal{G})$ ein Termmodell von P. Ist $\mathcal{J}$ ein Modell der Menge $\mathcal{G}$ der Grundatome, so ist $\mathcal{J}$ auch ein Modell von P.

Beweis von (*):
Sei P ein Grundatom. Nach Voraussetzung ist $\mathcal{T}(\mathcal{G})$ ein Termmodell von P, also ist $P \in \mathcal{G}$. Da $\mathcal{J}$ ein Modell von $\mathcal{G}$ ist, ist $\mathcal{J}$ insbesondere auch ein Modell von P.

Sei $P = P_1 \wedge P_2$ (bzw. $P = P_1 \vee P_2$). Nach Voraussetzung ist $\mathcal{T}(\mathcal{G})$ ein Termmodell von P_1 und P_2 (bzw. von P_1 oder P_2). Nach Induktionsvoraussetzung ist dann $\mathcal{J}$ ein Modell von P_1 und P_2 (bzw. von P_1 oder P_2) und damit von P.

Sei $P = \exists x Q(x)$. Nach Voraussetzung ist $\mathcal{T}(\mathcal{G})$ ein Termmodell von $\exists x Q(x)$, d.h. es gibt eine Grundsubstitution σ, so daß $\mathcal{T}(\mathcal{G}) \models Q\sigma$. Nach Induktionsvoraussetzung ist dann auch $\mathcal{J}$ ein Modell von $Q\sigma$ und damit auch von P.

Mit (*) kann nun direkt gezeigt werden, daß die gegebene Interpretation $\mathcal{I}$ auch ein Modell von $F\sigma$ ist: Nach Bemerkung 4.16 ist jeder Bereich einer Variablen auch eine positive Bedingung. Weiter enthält $B\sigma$ keine freien Variablen. Da die PUHR-Regel angewendet wird, ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Termmodell von $B\sigma$. Nach Voraussetzung ist $\mathcal{I}$ ein Modell von $\cup\mathcal{B}$. Insbesondere ist dann $\mathcal{I}$ auch ein Modell von $Atoms(\cup\mathcal{B})$. Mit (*) ist dann $\mathcal{I}$ ein Modell von $B\sigma$.

Da nach Voraussetzung $\mathcal{I}$ ein Modell von $\forall\bar{x}(B(\bar{x}) \rightarrow F)$ ist, ist $\mathcal{I}$ auch ein Modell von $F\sigma$.

Damit wurde gezeigt, daß auch die Anwendung der PUHR-Regel auf ein erfüllbares Tableau wieder zu einem erfüllbaren Tableau führt. ∎

Satz 4.36 (Widerlegungskorrektheit) *Gibt es ein geschlossenes EP-Tableau für $\mathcal{S}$, so ist $\mathcal{S}$ unerfüllbar.*

Beweis: Sei T das EP-Tableau für $\mathcal{S}$, das nur aus der Wurzel $\mathcal{S}$ besteht. Angenommen $\mathcal{S}$ ist erfüllbar. Dann führt nach Lemma 4.35 jede Anwendung einer Expansionsregel auf T zu einem erfüllbaren EP-Tableau, d.h. es gibt kein geschlossenes EP Tableau für $\mathcal{S}$. ∎

In Abschnitt 4.4 haben wir bereits gesehen, daß für die EP-Tableaux der Begriff der *Fairness* benötigt wird. Genauso wie für die PUHR-Tableaux wird in der folgenden Definition nun auch für die EP-Tableaux das Standardkonzept von „Fairness“ [Fit90] formalisiert. Dabei ist wieder zu beachten, daß jeder Knoten eines EP-Tableau aus Mengen von Formeln besteht.

Definition 4.37 (Fairness)
• Sei T ein EP-Tableau für $\mathcal{S}$ und $\mathcal{B}$ ein Zweig in T. Dann ist die Vereinigung aller Knoten $\cup\mathcal{B}$ *saturiert*, falls das folgende gilt:

1. Ist $E_1 \vee E_2 \in \cup\mathcal{B}$, dann ist $E_1 \in \cup\mathcal{B}$ oder $E_2 \in \cup\mathcal{B}$.
2. Ist $E_1 \wedge E_2 \in \cup\mathcal{B}$, dann ist $E_1 \in \cup\mathcal{B}$ und $E_2 \in \cup\mathcal{B}$.
3. Ist $\exists x E(x) \in \cup\mathcal{B}$, dann existiert $E\{c_1/x\} \in \cup\mathcal{B}$ oder ... oder $E\{c_n/x\} \in \cup\mathcal{B}$ oder $E\{c_{neu}/x\} \in \cup\mathcal{B}$. $c_1, \ldots, c_n$ sind all die Konstanten, die in dem Knoten auftreten, auf den die $\exists$-Regel angewendet wird und c_{neu} ist eine Konstante, die in diesem Knoten nicht auftritt.
4. Ist $\forall\bar{x}(B(\bar{x}) \rightarrow F) \in \cup\mathcal{B}$, dann ist für alle Grundsubstitutionen σ, für die $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $B\sigma$ ist, auch $F\sigma \in \cup\mathcal{B}$.

• Ein EP-Tableau T heißt *fair*, wenn $\cup\mathcal{B}$ für jeden offenen Zweig $\mathcal{B}$ von T saturiert ist.
□

Lemma 4.38 (Modellkorrektheit) *Sei T ein EP-Tableau für $\mathcal{S}$ und $\mathcal{B}$ ein offener Zweig in T. Ist T fair, so ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $\mathcal{S}$.*

Beweis: Sei φ eine PRQ-Formel aus $\mathcal{S}$. Lemma 4.38 wird durch Induktion über den Aufbau von φ bewiesen.

1. Sei φ ein Atom. Da φ ein Element von $\mathcal{S}$ ist und damit eine geschlossene Formel, ist φ ein Grundatom und in $Atoms(\cup\mathcal{B})$ enthalten. Damit ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von φ. Dabei ist die Voraussetzung, daß φ ein Element von $\mathcal{S}$ ist, nicht notwendig. Wesentlich ist nur, daß φ in $Atoms(\cup\mathcal{B})$ enthalten ist. Genauso gilt für jedes Grundatom ψ im Zweig $\mathcal{B}$, daß $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von ψ ist. Dies zu bemerken ist notwendig, damit die Induktionsvoraussetzung auf beliebige Formeln in $\cup\mathcal{B}$ angewendet werden kann.

2.Für $\varphi = \top$ ist nichts zu zeigen. Aufgrund der Widerlegungsvollständigkeit der EP-Tableaux-Methode ist $\bot \notin \mathcal{S}$.

3. Sei $\varphi = \varphi_1 \wedge \varphi_2$ (bzw. $\varphi = \varphi_1 \vee \varphi_2$). Da das Tableau T fair ist, sind φ_1 und φ_2 (bzw. φ_1 oder φ_2) in $\cup\mathcal{B}$ enthalten. Da φ ein Element aus $\mathcal{S}$ ist, enthält φ keine freien Variablen. Nach Induktionsvoraussetzung (φ_1 und φ_2 sind nicht notwendigerweise Elemente aus $\mathcal{S}$, aber aus $\cup\mathcal{B}$!) ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von φ_1 und φ_2 (bzw. von φ_1 oder φ_2) und damit von φ.

4. Sei $\varphi = \exists x\psi$. Da T fair ist, gibt es eine Konstante c, so daß $\psi\{c/x\}$ ein Element von $\cup\mathcal{B}$ ist. Nach Induktionsvoraussetzung ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $\psi\{c/x\}$ und damit auch von φ.

5. Sei $\varphi = \forall\bar{x}(B(\bar{x}) \rightarrow F)$ und σ eine Grundsubstitution, derart daß $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $B\sigma$ ist. Da das Tableau T fair ist, ist $F\sigma$ ein Element von $\cup\mathcal{B}$. Nach Induktionsvoraussetzung ist $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $F\sigma$ und damit auch von φ.

■

Aus Lemma 4.38 folgt direkt die Widerlegungsvollständigkeit für die EP-Tableaux-Methode.

Satz 4.39 (Widerlegungsvollständigkeit) *Ist $\mathcal{S}$ unerfüllbar, dann ist jedes faire EP-Tableau für $\mathcal{S}$ geschlossen.*

Beweis: Angenommen, es gibt ein faires EP-Tableau T für $\mathcal{S}$ mit einem offenen Zweig $\mathcal{B}$. Nach Lemma 4.38 ist dann aber $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $\mathcal{S}$, was im Widerspruch zur Voraussetzung steht. ■

Korollar 4.40 *Ist $\mathcal{S}$ nicht endlich erfüllbar, so ist jeder offene Zweig eines fairen EP-Tableau für $\mathcal{S}$ unendlich.*

Beweis: Sei $\mathcal{S}$ nicht endlich erfüllbar. Angenommen es existiert ein faires EP-Tableau T mit einem endlichen offenen Zweig $\mathcal{B}$. Nach Lemma 4.38 ist nun $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein endliches Modell von $\mathcal{S}$ im Widerspruch zur Voraussetzung. ■

4.6 Die Vollständigkeit für die Erfüllbarkeit im Endlichen

In diesem Abschnitt wird die Vollständigkeit der EP-Tableaux-Methode für die Erfüllbarkeit im Endlichen gezeigt. Der Beweis ist wesentlich komplexer als die Beweise der Resultate des vorhergehenden Abschnitts. Er stützt sich auf einige Nichtstandardbezeichnungen, die zunächst eingeführt werden.

Definition 4.41 (Einfache Erweiterung) Sei $\mathcal{S}$ eine erfüllbare Menge von PRQ-Formeln, φ ein Element von $\mathcal{S}$ und $\mathcal{T}(\mathcal{G})$ ein Termmodell von $\mathcal{S}$. Eine *Einfache Erweiterung* $\mathcal{S}'$ *von* $\mathcal{S}$ *bezüglich* φ *und* $\mathcal{T}(\mathcal{G})$ wird nun folgendermaßen definiert:

1. Ist φ ein Grundatom, so ist $\mathcal{S}' := \mathcal{S}$.
2. Ist $\varphi = \varphi_1 \wedge \varphi_2$, so ist $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{\varphi_1, \varphi_2\}$.
3. Ist $\varphi = \varphi_1 \vee \varphi_2$, so ist $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{\varphi_i\}$ für ein $i \in \{1, 2\}$, derart daß $\mathcal{T}(\mathcal{G})$ ein Modell von φ_i ist.[7]
4. Ist $\varphi = \exists x \psi$, so ist $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{\psi\{c/x\}\}$, wobei c eine Konstante ist, derart daß $\mathcal{T}(\mathcal{G})$ ein Modell von $\psi\{c/x\}$ ist.[8]
5. Ist $\varphi = \forall \bar{x}(B(\bar{x}) \rightarrow F)$, so ist $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{F\{\bar{c}/\bar{x}\} \mid \bar{c}$ ist ein Tupel von Konstanten, so daß $\mathcal{T}(\mathcal{G})$ ein Modell von $B\{\bar{c}/\bar{x}\}$ ist $\}$.[9]

□

Bemerkung 4.42 Der Begriff der Einfachen Erweiterung wurde eingeführt, um die Expansion eines EP-Tableau zu modellieren. Da es erstens das Ziel ist, die Vollständigkeit der EP-Tableaux-Methode für die Erfüllbarkeit im Endlichen zu beweisen, wird die Expansion bezüglich eines gesuchten Modells modelliert. Da zweitens ein EP-Tableau durch die Anwendung einer Expansionsregel auf eine Formel schrittweise expandiert wird, wird die Expansion auch bezüglich einer Formel modelliert. Diese Überlegungen führten zum Begriff der einfachen Erweiterung. Für einfache Erweiterungen gelten folgende Beobachtungen:

- Für jedes Element φ in $\mathcal{S}$ und jedes Modell $\mathcal{T}(\mathcal{G})$ von $\mathcal{S}$ existiert mindestens eine einfache Erweiterung von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{G})$.
- Eine einfache Erweiterung $\mathcal{S}'$ von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{G})$ unterscheidet sich von $\mathcal{S}$ immer dann, wenn $\varphi \in \mathcal{S}$ eine nichtatomare Formel ist.
- Andererseits bedeutet die Existenz einer einfachen Erweiterung $\mathcal{S}'$ von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{G})$ mit $\mathcal{S} \neq \mathcal{S}'$ nicht notwendigerweise, daß eine EP-Tableau-Expansionsregel auf φ angewendet werden kann. Erinnern wir uns an Definition 4.29, so kann eine Expansionsregel nur dann auf eine Formel φ angewendet werden, wenn die vom betreffenden Zweig repräsentierte Terminterpretation die Formel φ nicht erfüllt. Ist also $\mathcal{T}(Atoms(\mathcal{S}))$ ein Modell von einer nichtatomaren Formel φ, so kann keine EP-Expansionsregel auf φ angewendet werden. Trotzdem existiert eine einfache Erweiterung $\mathcal{S}'$ von $\mathcal{S}$ bezüglich φ und eines gegebenen Termmodells $\mathcal{T}(\mathcal{G})$.

[7] Da $\mathcal{T}(\mathcal{G})$ nach Voraussetzung ein Termmodell von $\mathcal{S}$ und $\varphi \in \mathcal{S}$ ist, ist $\mathcal{T}(\mathcal{G})$ auch ein Termmodell von φ_i für mindestens ein $i \in \{1, 2\}$.

[8] Eine solche Konstante existiert, da $\mathcal{T}(\mathcal{G})$ ein Termmodell von $\mathcal{S}$ und $\varphi \in \mathcal{S}$ ist.

[9] Gibt es keine Konstanten $\bar{c}$, so daß $\mathcal{T}(\mathcal{G})$ ein Termmodell von $B\{\bar{c}/\bar{x}\}$ ist, so ist nach dieser Definition $\mathcal{S}' := \mathcal{S} \setminus \{\varphi\}$.

- Jede einfache Erweiterung einer *endlichen* Menge $\mathcal{S}$ bezüglich einer Formel φ und eines *endlichen* Modells $\mathcal{T}(\mathcal{G})$ von $\mathcal{S}$ ist wieder endlich. Dies trifft auf einfache Erweiterungen bezüglich unendlicher Modelle $\mathcal{T}(\mathcal{G})$ nicht notwendigerweise zu, da in 5. in Definition 4.41 die Menge $\{F\{\bar{c}/\bar{x}\} \mid \bar{c}$ ist ein Tupel von Konstanten, so daß $\mathcal{T}(\mathcal{G})$ ein Modell von $B\{\bar{c}/\bar{x}\}$ ist$\}$ möglicherweise unendlich ist.

Lemma 4.43 *Sei $\mathcal{S}$ eine Menge von PRQ-Formeln, φ ein Element aus $\mathcal{S}$, $\mathcal{T}(\mathcal{E})$ ein endliches, minimales Termmodell von $\mathcal{S}$ und $\mathcal{S}'$ eine einfache Erweiterung von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{E})$. Dann ist $\mathcal{T}(\mathcal{E})$ auch ein minimales Modell von $\mathcal{S}'$.*

Beweis: Der folgende Beweis von Lemma 4.43 beruht auf einer Fallunterscheidung hinsichtlich der Formel φ.

1. Sei φ ein Atom. Dann gilt für die einfache Erweiterung $\mathcal{S}'$ von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{E})$: $\mathcal{S}' = \mathcal{S}$. Damit ist $\mathcal{T}(\mathcal{E})$ natürlich auch ein endliches, minimales Termmodell von $\mathcal{S}'$.

In den folgenden Fällen wird jeweils zunächst gezeigt, daß $\mathcal{T}(\mathcal{E})$ ein Modell von $\mathcal{S}'$ ist und dann in einem zweiten Schritt, daß $\mathcal{T}(\mathcal{E})$ auch ein minimales Modell von $\mathcal{S}'$ ist.

2. Sei $\varphi = \varphi_1 \wedge \varphi_2$. Nach Definition 4.41 ist $\mathcal{S}' = (\mathcal{S} \setminus \{\varphi\}) \cup \{\varphi_1, \varphi_2\}$. Nach Voraussetzung ist $\mathcal{T}(\mathcal{E})$ ein Modell von $\mathcal{S}$, insbesondere von $\varphi = \varphi_1 \wedge \varphi_2$. Also ist $\mathcal{T}(\mathcal{E})$ ein Modell von φ_1 und φ_2 und damit von $\mathcal{S}'$.

Angenommen, $\mathcal{T}(\mathcal{E})$ sei kein minimales Modell von $\mathcal{S}'$. Dann gibt es eine echte Teilmenge $\mathcal{E}' \subset \mathcal{E}$, so daß $\mathcal{T}(\mathcal{E}')$ ein Modell von $\mathcal{S}'$ ist. Da φ_1 und φ_2 Elemente aus $\mathcal{S}'$ sind, ist dann $\mathcal{T}(\mathcal{E}')$ auch ein Modell von $\varphi = \varphi_1 \wedge \varphi_2$ und damit auch von $\mathcal{S}$. Dies steht aber im Widerspruch zur vorausgesetzten Minimalität von $\mathcal{T}(\mathcal{E})$.

3. Sei $\varphi = \varphi_1 \vee \varphi_2$. Nach Definition 4.41 gilt für $\varphi_i \in \mathcal{S}'$ mit $i \in \{1, 2\}$, daß $\mathcal{T}(\mathcal{E})$ ein Modell von φ_i ist. Deshalb ist $\mathcal{T}(\mathcal{E})$ auch ein Modell von $\mathcal{S}'$.

Angenommen, $\mathcal{T}(\mathcal{E})$ sei kein minimales Modell von $\mathcal{S}'$, d.h. es gibt eine echte Teilmenge $\mathcal{E}' \subset \mathcal{E}$, so daß $\mathcal{T}(\mathcal{E}')$ ein Modell von $\mathcal{S}'$ ist. Dann ist nach Definition 4.41 $\mathcal{T}(\mathcal{E}')$ insbesondere ein Modell von φ_i für ein $i \in \{1, 2\}$ und damit von $\varphi = \varphi_1 \vee \varphi_2$. Also ist $\mathcal{T}(\mathcal{E}')$ ein Modell von $\mathcal{S}$, was im Widerspruch zur Voraussetzung steht.

4. Sei $\varphi = \exists x \psi$. Nach Definition 4.41 gilt für $\psi\{c/x\} \in \mathcal{S}'$, daß $\mathcal{T}(\mathcal{E})$ ein Modell von $\psi\{c/x\}$ ist. Also ist $\mathcal{T}(\mathcal{E})$ auch ein Modell von $\mathcal{S}'$.

Angenommen, $\mathcal{T}(\mathcal{E})$ ist kein minimales Modell von $\mathcal{S}'$ und $\mathcal{T}(\mathcal{E}')$ mit $\mathcal{E}' \subset \mathcal{E}$ ist ein Modell von $\mathcal{S}'$. Wegen $\psi\{c/x\} \in \mathcal{S}'$ ist dann $\mathcal{T}(\mathcal{E}')$ auch ein Modell von $\varphi = \exists x \psi$ und damit von $\mathcal{S}$, was im Widerspruch zur Voraussetzung steht.

5. Sei $\varphi = \forall \bar{x}(B(\bar{x}) \rightarrow F)$. Nach Definition 4.41 gilt für alle Konstantentupel $\bar{c}$, für die $F\{\bar{c}/\bar{x}\}$ in $\mathcal{S}'$ enthalten ist, daß $\mathcal{T}(\mathcal{E})$ ein Modell von $B\{\bar{c}/\bar{x}\}$ ist. Da andererseits nach Voraussetzung $\mathcal{T}(\mathcal{E})$ ein Modell von $\varphi = \forall \bar{x}(B(\bar{x}) \rightarrow F)$ ist, ist $\mathcal{T}(\mathcal{E})$ dann auch ein Modell aller $F\{\bar{c}/\bar{x}\}$ in $\mathcal{S}'$ und damit auch von $\mathcal{S}'$.

Angenommen, $\mathcal{T}(\mathcal{E})$ ist kein minimales Modell von $\mathcal{S}'$ und $\mathcal{T}(\mathcal{E}')$ mit $\mathcal{E}' \subset \mathcal{E}$ ist ein Modell von $\mathcal{S}'$. Dann ist $\mathcal{T}(\mathcal{E}')$ auch ein Modell von $\varphi = \forall \bar{x}\ (B(\bar{x}) \rightarrow F)$, denn sei σ eine Substitution, so daß $\mathcal{T}(\mathcal{E}')$ ein Modell von $B\sigma$ ist. Nach Lemma 4.21 ist dann auch $\mathcal{T}(\mathcal{E})$ ein Modell von $B\sigma$. Da nach Voraussetzung $\mathcal{T}(\mathcal{E})$ ein Modell von φ ist, ist $\mathcal{T}(\mathcal{E})$ ein Modell von $F\sigma$. Nach Definition 4.41 ist dann $F\sigma$ in $\mathcal{S}'$ enthalten. Also ist $\mathcal{T}(\mathcal{E}')$ ein Modell von $F\sigma$ und damit von φ. Damit ist $\mathcal{T}(\mathcal{E}')$ ein Modell von $\mathcal{S}$, was der Minimalität von $\mathcal{T}(\mathcal{E})$ widerspricht. ∎

Der in der folgenden Definition eingeführte Begriff des *Ranges* wird benötigt, um einen Induktionsparameter im Beweis der Vollständigkeit für die Erfüllbarkeit im Endlichen zur Verfügung zu haben. Die Definition unterscheidet sich von Standarddefinitionen einmal darin, daß der Rang einer Formel immer bezüglich der Mächtigkeit eines Modells definiert wird. Zum anderen kann im Falle der universell quantifizierten Formeln die PUHR-Regel möglicherweise mehrmals auf ein- und diesselbe Formel angewendet werden. Dieser Tatsache wird ebenfalls Rechnung getragen.

Definition 4.44 (Rang)
• Sei φ eine (nicht notwendigerweise geschlossene) PRQ-Formel und d eine von Null verschiedene natürliche Zahl. Dann wird der *d-Rang* $rg(\varphi, d)$ von φ folgendermaßen induktiv definiert:

1. Ist φ ein Atom, so ist $rg(\varphi, d) := 0$.
2. Ist $\varphi = \varphi_1 \wedge \varphi_2$, $\varphi = \varphi_1 \vee \varphi_2$ oder $\varphi = \varphi_1 \rightarrow \varphi_2$,
   dann ist $rg(\varphi, d) := rg(\varphi_1, d) + rg(\varphi_2, d) + 1$.
3. Ist $\varphi = \exists x \psi$, dann ist $rg(\varphi, d) := rg(\psi, d) + 1$.
4. Ist $\varphi = \forall \bar{x} \psi$, dann ist $rg(\varphi, d) := rg(\psi, d) \cdot d^n$, wobei n die Länge des Tupels $\bar{x}$ ist.

• Sei $\mathcal{S}$ eine Menge von PRQ-Formeln, $\mathcal{T}(\mathcal{E})$ ein endliches minimales Modell von $\mathcal{S}$ und d die Mächtigkeit der Domäne von $\mathcal{T}(\mathcal{E})$. Dann wird der *Rang* $rg(\mathcal{S}, \mathcal{T}(\mathcal{E}))$ von $\mathcal{S}$ bezüglich $\mathcal{T}(\mathcal{E})$ definiert durch $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) := \sum_{\psi \in \mathcal{S}} rg(\psi, d)$, falls $Atoms(\mathcal{S}) \subset \mathcal{E}$, $Atoms(\mathcal{S}) \neq \mathcal{E}$und $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) := 0$, falls $Atoms(\mathcal{S}) = \mathcal{E}$.

□

Bemerkung 4.45 Der Rang einer Menge von PRQ-Formeln wurde so definiert, daß folgendes gilt: $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) = 0$ genau dann, wenn $\mathcal{T}(Atoms(\mathcal{S}))$ ein Modell von $\mathcal{S}$ ist. Mit anderen Worten ist $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) > 0$ genau dann, wenn eine EP-Tableau-Expansionsregel auf eine Formel in $\mathcal{S}$ angewendet werden kann.

Lemma 4.46 *Sei $\mathcal{S}$ eine endlich erfüllbare Menge von PRQ-Formeln, φ ein nicht atomares Element aus $\mathcal{S}$, $\mathcal{T}(\mathcal{E})$ ein endliches minimales Modell von $\mathcal{S}$ und $\mathcal{S}'$ eine einfache Erweiterung von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{E})$. Ist $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) \neq 0$, dann ist $rg(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rg(\mathcal{S}, \mathcal{T}(\mathcal{E}))$.*

Um Lemma 4.46 zu beweisen, ist folgendes Hilfslemma nötig:

Hilfslemma 4.47 *Sei φ eine PRQ-Formel, d eine natürliche Zahl und $\bar{x}$ die freien Variablen in φ. Dann gilt für jede Substitution $\sigma = \{\bar{c}/\bar{x}\}$:*

$$rg(\varphi, d) = rg(\varphi\{\bar{c}/\bar{x}\}, d).$$

Beweis:
Hilfslemma 4.47 gilt offensichtlich, da $rg(\varphi, d)$ unabhängig von den freien Variablen in φ definiert wurde. ∎

Damit kann Lemma 4.46 bewiesen werden.

Beweis: Die natürliche Zahl d bezeichne die Mächtigkeit von $\mathcal{T}(\mathcal{E})$. Es werden die verschiedenen Fälle von φ betrachtet. Die Behauptung folgt dann durch einfaches Nachrechnen.

1. Sei $\varphi = \varphi_1 * \varphi_2$, wobei $* \in \{\wedge, \vee, \rightarrow\}$.

$$\begin{aligned} rg(\mathcal{S}', \mathcal{T}(\mathcal{E})) &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - rg(\varphi, d) + rg(\varphi_1, d) + rg(\varphi_2, d) \\ &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - 1 \\ &< rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) \end{aligned}$$

2. Sei $\varphi = \exists x \psi$.

$$\begin{aligned} rg(\mathcal{S}', \mathcal{T}(\mathcal{E})) &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - rg(\varphi, d) + rg(\psi, d) \\ &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - 1 \\ &< rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) \end{aligned}$$

3. Sei $\varphi = \forall \bar{x}(B(x) \rightarrow F)$, wobei n die Länge des Tupels x ist.

$$\begin{aligned} rg(\mathcal{S}', \mathcal{T}(\mathcal{E})) &\leq rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - rg(\varphi, d) + d^n \cdot rg(F, d) \\ &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - (rg(B(x) \rightarrow F, d) \cdot d^n) + (d^n \cdot rg(F, d)) \\ &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - ((rg(B(x), d) + rg(F, d) + 1) \cdot d^n) \\ &\quad + (d^n \cdot rg(F, d)) \\ &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - (rg(B(x), d) \cdot d^n) - (rg(F, d) \cdot d^n) - d^n \\ &\quad + (d^n \cdot rg(F, d)) \\ &= rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) - (\underbrace{rg(B(x), d)}_{\geq 0} \cdot d^n) - \underbrace{d^n}_{>0} \\ &< rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) \end{aligned}$$

   Die erste Ungleichung gilt, da $\mathcal{T}(\mathcal{E})$ d Elemente hat. Daher gibt es höchstens d^n verschiedene Konstantentupel $\bar{c}$ der Länge n, so daß $F\{\bar{c}/\bar{x}\}$ in $\mathcal{S}'$ enthalten ist. Die letzte Ungleichung gilt, da die Definition eines Termmodells keine leere Domäne zuläßt.

∎

Damit wurden alle Hilfsbegriffe eingeführt und die notwendigen Eigenschaften gezeigt, um die Vollständigkeit der EP-Tableaux-Methode für die Erfüllbarkeit im Endlichen zeigen zu können.

Satz 4.48 (Vollständigkeit für die Erfüllbarkeit im Endlichen) *Sei $\mathcal{T}(\mathcal{E})$ ein endliches Termmodell von $\mathcal{S}$. Ist $\mathcal{T}(\mathcal{E})$ ein minimales Modell von $\mathcal{S}$, dann besitzt jedes fairidee EP-Tableau für $\mathcal{S}$ einen endlichen, offenen Zweig $\mathcal{B}$, derart daß, bis auf Konstantenumbenennung, die Menge der im Zweig auftretenden Grundatome $Atoms(\cup\mathcal{B})$ gleich der Menge $\mathcal{E}$ ist.*

Der Beweis basiert auf einer doppelten Induktion. Dies ist notwendig, da die PUHR-Regel entlang eines Zweiges mehrmals auf die gleiche Formel angewendet werden kann, natürlich immer bezüglich verschiedener Substitutionen der universell quantifizierten Variablen. Deshalb nimmt die syntaktische Komplexität der betrachteten Formelmenge, die ein natürlicher Induktionsparameter wäre, nach Anwendung der PUHR-Regel nicht ab. Aus diesem Grund wird eine zweite Induktion über die Anzahl der Anwendungen der PUHR-Regel geführt.

Beweis: Sei $\mathcal{T}(\mathcal{E})$ ein endliches minimales Termmodell von $\mathcal{S}$. Der Beweis wird durch Induktion über $rg(\mathcal{S}, \mathcal{T}(\mathcal{E}))$ geführt. Die Induktionshypothese lautet:

($*$) Sei $\mathcal{M}$ eine Menge von PRQ-Formeln. Ist $\mathcal{T}(\mathcal{F})$ ein endliches minimales Termmodell von $\mathcal{M}$, und ist $rg(\mathcal{M}, \mathcal{T}(\mathcal{F})) < n$, dann hat jedes faire EP-Tableau für $\mathcal{M}$ einen endlichen, offenen Zweig $\mathcal{B}$ derart, daß bis auf Konstantenumbenennung $Atoms(\cup\mathcal{B}) = \mathcal{F}$ gilt.

Induktionsanfang: $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) = 0$. Dann hat $\mathcal{S}$ ein einziges minimales Termmodell, und zwar $\mathcal{T}(Atoms(\mathcal{S}))$. Weiter hat dann jedes faire EP-Tableau für $\mathcal{S}$ nur einen einzigen Knoten, der aus $\mathcal{S}$ besteht. Damit gilt die Behauptung trivialerweise.

Induktionsschritt: $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) = n > 0$. Sei T ein faires EP-Tableau für $\mathcal{S}$. Da $\mathcal{S}$ erfüllbar ist, ist nach Satz 4.36 T ein offenes Tableau. Da $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) > 0$, gibt es mindestens eine Formel φ in $\mathcal{S}$, auf die eine EP-Expansionsregel angewendet werden kann. Da T nach Voraussetzung fair ist, hat die Wurzel von T notwendigerweise mindestens einen Kinderknoten. Sei nun $\varphi \in \mathcal{S}$ diejenige Formel, auf die eine EP-Expansionsregel angewendet wird, was zu einem oder mehreren Kinderknoten der Wurzel von T führt. Im folgenden wird eine Fallunterscheidung bezüglich der Form von φ durchgeführt.

Fall 1: Sei $\varphi = \varphi_1 \wedge \varphi_2$, $\varphi = \varphi_1 \vee \varphi_2$ oder $\varphi = \exists x\psi$. Nach Definition 4.41 gibt es mindestens einen Kinderknoten N der Wurzel, so daß der Knoten N aus φ und einer Menge $\mathcal{S}'$ besteht, die, bis auf Konstantenumbenennung im Falle $\varphi = \exists x\psi$, eine einfache Erweiterung von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{E})$ ist.

Weiter darf eine EP-Expansionsregel höchstens einmal auf eine Formel der Form φ angewendet werden. Deshalb ist das Tableau, dessen Wurzel der Knoten N in T ist, ein

EP-Tableaux T' für die einfache Erweiterung $\mathcal{S}'$, sofern die Formel φ aus allen Knoten eliminiert wird. Da auf φ aber keine Expansionsregel mehr angewendet werden darf, und da φ kein Grundatom ist und deshalb $Atoms(\cup\mathcal{B})$ nicht beeinflußt, ist die Existenz von φ im Knoten N und in seinen Nachfolgerknoten völlig irrelevant. Das EP-Tableau T' ist fair, da T fair ist.

Nach Lemma 4.43 ist für jede einfache Erweiterung $\mathcal{S}'$ von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{E})$ das Termmodell $\mathcal{T}(\mathcal{E})$ auch ein minimales Modell von $\mathcal{S}'$. Da weiter $rg(\mathcal{S}, \mathcal{T}(\mathcal{E})) > 0$ und φ nichtatomar ist, gilt nach Lemma 4.46 $rg(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rg(\mathcal{S}, \mathcal{T}(\mathcal{E}))$. Damit besitzt nach Induktionshypothese $(*)$ das Tableau T' einen endlichen offenen Zweig $\mathcal{B}'$, so daß bis auf Konstantenumbenennung, $Atoms(\cup\mathcal{B}') = \mathcal{E}$. Da aber T und T' im Knoten N und allen nachfolgenden Knoten bis auf das irrelevante Vorkommen von φ übereinstimmen, gilt die Behauptung auch für T.

Fall 2: Sei $\varphi = \forall\bar{x}(B(\bar{x}) \rightarrow F)$. Weiter sei $\mathcal{S}'$ die (eindeutige) einfache Erweiterung von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{E})$. Entlang eines Zweiges eines fairen EP-Tableau T für $\mathcal{S}$ kann die PUHR-Regel nun möglicherweise mehrfach auf die Formel φ angewandt werden. Deshalb ist das EP-Tableaux, dessen Wurzel der Kinderknoten N der Wurzel von T ist, nicht notwendigerweise ein EP-Tableau für $\mathcal{S}'$.

Im folgenden wird nun gezeigt wie Teile des EP-Tableau T für $\mathcal{S}$ als Teile eines EP-Tableau T' für $\mathcal{S}'$ betrachtet werden können. Für $m \in \mathbb{N}$ und einen Zweig $\mathcal{B}$ von T sei $\mathcal{B}^m$ das Präfix von $\mathcal{B}$ bis zu der (und ohne die) $(m+1)$-ste Anwendung der PUHR-Regel auf φ, falls die PUHR-Regel mehr als m-mal auf φ im Zweig $\mathcal{B}$ angewendet wird; sonst bezeichne $\mathcal{B}^m$ den ganzen Zweig $\mathcal{B}$. Es wird zunächst folgende Behauptung durch Induktion über die natürlichen Zahlen gezeigt: Für alle $m \in \mathbb{N}$ gilt, daß

$(**)$ T besitzt einen Zweig $\mathcal{B}$ derart, daß bis auf Konstantenumbenennung $Atoms(\mathcal{B}^m) \subseteq \mathcal{E}$ gilt.

$m = 1$: Der Kinderknoten N der Wurzel des Tableau T entsteht durch eine Anwendung der PUHR-Regel auf die Formel $\varphi = \forall\bar{x}(B(\bar{x}) \rightarrow F)$. Damit gibt es nach Definition 4.28 und 4.29 eine Menge $\mathcal{G}$ von Grundatomen in $\mathcal{S}$ und eine Substitution σ derart, daß $\mathcal{T}(\mathcal{G})$ ein Modell von $B\sigma$, aber nicht von $F\sigma$ ist. Weiter besteht der Knoten N aus $\mathcal{S} \cup \{F\sigma\}$. Da nach Voraussetzung $\mathcal{T}(\mathcal{E})$ ein Modell von $\mathcal{S}$ ist, ist die Menge $\mathcal{G}$ eine Teilmenge von $\mathcal{E}$. Nach Lemma 4.21 in Abschnitt 4.3 ist dann auch $\mathcal{T}(\mathcal{E})$ ein Modell von $B\sigma$. Da $\mathcal{T}(\mathcal{E})$ nach Voraussetzung ein Modell von φ ist, ist $\mathcal{T}(\mathcal{E})$ dann also auch ein Modell von $F\sigma$. Damit ist $F\sigma$ ein Element der (eindeutigen) einfachen Erweiterung $\mathcal{S}'$ von $\mathcal{S}$ bezüglich φ und $\mathcal{T}(\mathcal{E})$.

Es gibt also ein EP-Tableau T' für $\mathcal{S}'$, das mit T von dem Knoten N bis zur zweiten Anwendung der PUHR-Regel auf die Formel φ in allen Zweigen übereinstimmt, abgesehen von der Formel φ, die in T vorkommt aber nicht in T'. Da das Tableau T aber nur *zwischen* zwei Anwendungen der PUHR-Regel auf φ betrachtet wird, ist die Formel φ in den betrachteten Knoten irrelevant.

Da nun nach Lemma 4.43 $\mathcal{T}(\mathcal{E})$ auch ein minimales Modell von $\mathcal{S}'$ ist und weiter nach Lemma 4.46 $rg(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rg(\mathcal{S}, \mathcal{T}(\mathcal{E}))$ ist, kann die Induktionsannahme $(*)$ angewendet

werden: Im EP-Tableau T' gibt es einen Zweig $\mathcal{B}'$ mit $Atoms(\cup\mathcal{B}') = \mathcal{E}$ bis auf Konstantenumbenennung. Also sind die Grundatome in dem korrespondierenden Zweig $\mathcal{B}$ in T eine Teilmenge von $\mathcal{E}$ ($Atoms(\mathcal{B}^1) \subseteq \mathcal{E}$).

$m > 1$: Angenommen, die Behauptung $(**)$ gilt für alle $m' \leq m$. Seien nun $\mathcal{B}_1, \ldots, \mathcal{B}_k$ all diejenigen Zweige in T mit $Atoms(\mathcal{B}_i^m) \subseteq \mathcal{E},\ i = 1, \ldots, k$.
Gibt es ein $i \in \{1, \ldots, k\}$ mit $\mathcal{B}_i^m = \mathcal{B}_i^{m+1} = \mathcal{B}_i$, d.h. die PUHR-Regel wird höchstens m-mal auf die Formel φ entlang des Zweiges $\mathcal{B}_i$ angewendet, dann ist nach Induktionsannahme $(**)$ $Atoms(\mathcal{B}_i^{m+1}) \subseteq \mathcal{E}$.

Sonst, da nach Induktionsannahme $(**)$ $Atoms(\mathcal{B}_i^m) \subseteq \mathcal{E}$, ist nach Lemma 4.21 und nach Definition von $\mathcal{S}'$ jede Formel $F\sigma_i$, die durch die $(m+1)$-ste Anwendung der PUHR-Regel auf die Formel φ im Zweig $\mathcal{B}_i$ entsteht, in $\mathcal{S}'$ enthalten. Deshalb kann folgendermaßen ein EP-Tableau T' für $\mathcal{S}'$ aus dem Teiltableau von T konstruiert werden, dessen Wurzel der Knoten N ist:

Als erstes wird der Knoten N durch $\mathcal{S}'$ ersetzt. Als zweites wird von jedem Zweig $\mathcal{B}_i$ nur das Präfix $\mathcal{B}_i^{m+1}$ behalten und der Rest abgeschnitten. Drittens werden in jedem Zweig $\mathcal{B}_i^{m+1}$ all diejenigen Knoten eliminiert, die aufgrund einer Anwendung der PUHR-Regel auf die Formel φ entstanden sind. Schließlich werden alle anderen Zweige direkt vor der ersten Anwendung der PUHR-Regel auf die Formel φ abgeschnitten.

T' ist ein endliches EP-Tableau für $\mathcal{S}'$, das aber nicht notwendigerweise fair sein muß. Da T' endlich ist, kann aber ein faires EP-Tableau T'' für $\mathcal{S}'$ dadurch erhalten werden, daß T' weiter expandiert wird. Nach Lemma 4.46 ist $rg(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rg(\mathcal{S}, \mathcal{T}(\mathcal{E}))$. Deshalb hat nach Induktionsannahme $(*)$ das EP-Tableau T'' einen Zweig $\mathcal{B}'$ mit $Atoms(\cup\mathcal{B}') = \mathcal{E}$ bis auf Konstantenumbenennung. Nach Definition von $\mathcal{B}_1, \ldots, \mathcal{B}_k$ und T'' gibt es einen Zweig $\mathcal{B}_i$ in T derart, daß $Atoms(\mathcal{B}_i^{m+1}) = Atoms(\mathcal{B}'^{m+1}_i)$. Da aber $Atoms(\cup\mathcal{B}') = \mathcal{E}$ ist dann also $Atoms(\mathcal{B}_i^{m+1}) \subseteq \mathcal{E}$.

Damit wurde die Behauptung $(**)$ gezeigt. Also kann nun der Induktionsschritt für die Behauptung $(*)$ für die Formel $\varphi = \forall\bar{x}(B(\bar{x}) \to F)$ beendet werden.

Da nach Voraussetzung $\mathcal{T}(\mathcal{E})$ endlich und das EP-Tableau T fair ist, hat T einen endlichen Zweig $\mathcal{B}$ für den die Behauptung $(**)$ gilt. Dieser Zweig ist offen. Da T fair ist, ist nach Lemma 4.38 $\mathcal{T}(Atoms(\cup\mathcal{B}))$ ein Modell von $\mathcal{S}$. Da weiter $\mathcal{T}(\mathcal{E})$ ein minimales Modell ist, gilt dann bis auf Konstantenumbenennung $Atoms(\cup\mathcal{B}) = \mathcal{E}$.

■

Bemerkung 4.49 Die Vollständigkeit der Erfüllbarkeit im Endlichen gilt für die EP-Tableaux-Methode *bis auf Konstantenumbenennung*, wie Satz 4.48 zeigte. Dieser Zusatz ist aufgrund der $\exists$-Regel notwendig. Betrachten wir etwa die Formel $F = \exists x\ p(x)$. Ein minimales Termmodell von F ist z.B. das Termmodell $\mathcal{T}(\{p(a)\})$. Der einzige Zweig des EP-Tableau von F, das durch Anwendung der $\exists$-Regel gemäß Definition 4.28 auf F entsteht, repräsentiert jedoch das Termmodell $\mathcal{T}(\{p(c_{neu})\})$. Durch Umbenennung der Konstante c_{neu} zu a erhalten wir offensichtlich das Termmodell $\mathcal{T}(\{p(a)\})$. Der Name c_{neu} für die

mittels der $\exists$-Regel neu eingeführte Konstante ist natürlich willkürlich. Deshalb bedeutet der Zusatz „bis auf Konstantenumbenennung“ in Satz 4.48 keine Einschränkung.

Nachdem die Vollständigkeit der Erfüllbarkeit im Endlichen für die EP-Tableaux-Methode gezeigt werden konnte, sind an dieser Stelle bereits ein paar Bemerkungen hinsichtlich einer Implementierung angebracht. So stellt sich heraus, daß die implementierte Suchstrategie Konsequenzen hinsichtlich der Vollständigkeit für die Erfüllbarkeit im Endlichen hat.

Aus Satz 4.48 folgt, daß eine Implementierung der EP-Tableaux-Methode mit Breitensuche vollständig für die Erfüllbarkeit im Endlichen ist. Ist dagegen eine Tiefensuche implementiert, so gilt dies nicht mehr, wie das folgende Beispiel zeigt.

Beispiel 4.50 Betrachten wir die PRQ-Formeln

$F_1 = s(a, b)$
$F_2 = \forall xy\ (s(x, y) \rightarrow \exists z\ s(y, z))$
$F_3 = \forall xyz\ (s(x, y) \wedge s(y, z) \rightarrow s(x, z))$
$F_4 = \forall xy\ (s(x, y) \wedge s(y, x) \rightarrow \bot).$

Weiter sei $\mathcal{S}_3 = \{F_1, F_2, F_3, F_4\}$. Abbildung 4.3 zeigt, daß die Modelle von $\mathcal{S}_3$ unendlich sind.

Betrachten wir nun die Formel $G = (F_1 \wedge F_2 \wedge F_3 \wedge F_4) \vee p(a)$ und sei $\mathcal{S}_4 = \{G\}$. Abbildung 4.4 zeigt ein EP-Tableau für $\mathcal{S}_4$. Würde dieses EP-Tableau mittels Tiefensuche von links nach rechts expandiert, so würde zunächst der unendliche Zweig expandiert werden, der zu dem unendlichen Modell $\mathcal{T}(\{s(a, b), s(b, c_1), s(a, c_1), s(c_1, c_2), \dots\})$ führt. Das Verfahren würde also nicht terminieren, obgleich im EP-Tableau auf der rechten Seite ein endlicher Zweig existiert, der auf das endliche Modell $\mathcal{T}(\{p(a)\})$ führen würde.

Wie Beispiel 4.50 zeigte, verlieren wir durch eine Implementierung der EP-Tableaux-Methode mit Tiefensuche die Vollständigkeit für die Erfüllbarkeit im Endlichen. Die Widerlegungsvollständigkeit (siehe Satz 4.39) bleibt dagegen erhalten.

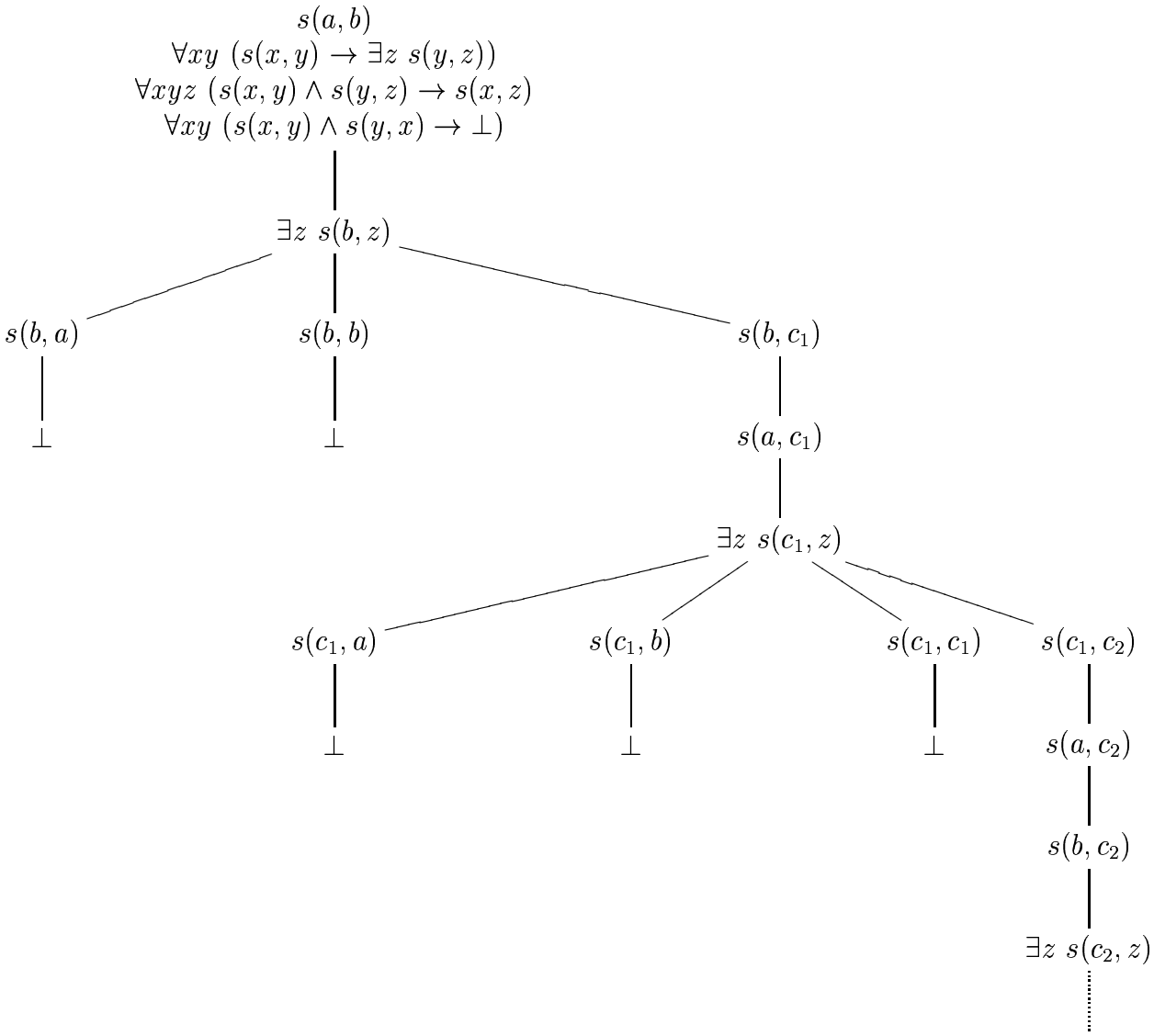


Abbildung 4.3: Ein EP-Tableau für $\mathcal{S}_3$ in Beispiel 4.50

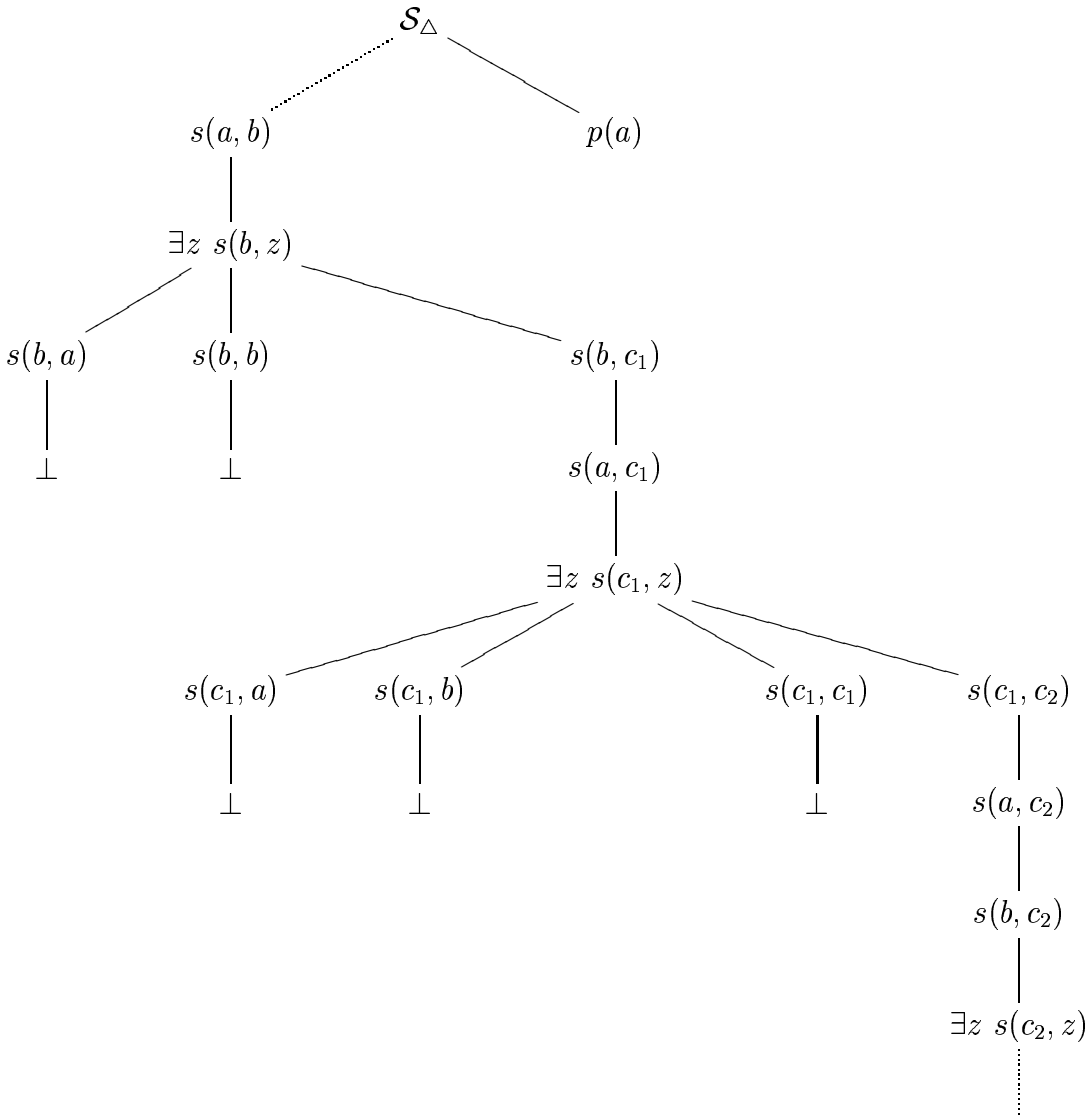


Abbildung 4.4: Ein EP-Tableau für $\mathcal{S}_4$ in Beispiel 4.50

Kapitel 5

Implementierung der EP-Tableaux-Methode in Prolog

In diesem Kapitel wird im ersten Abschnitt die Implementierung FINFIMO der EP-Tableaux-Methode vorgestellt. Der Name FINFIMO steht für FINd FInite MOdells. Bei FINFIMO handelt es sich um eine Implementierung in Prolog. FINFIMO expandiert die Zweige eines fairen EP-Tableau mittels Tiefensuche. Wie wir in Abschnitt 4.6 gesehen haben, geht bei der Tiefensuche die Vollständigkeit für die Erfüllbarkeit im Endlichen verloren. Deshalb ist es für Anwendungen wie etwa das System SIC [BEST98a, BEST98c, BEST98b], das in Kapitel 7.3 ausführlich beschrieben wird, unbedingt notwendig, die EP-Tableaux-Methode mit Breitensuche oder einer kontrollierten Suche zu implementieren.

Trotz dieses Nachteils ist die Verwendung der Tiefensuche zunächst sinnvoll, da die Prolog-eigene Tiefensuche ausgenutzt werden kann. Deshalb kann die EP-Tableaux-Methode sehr elegant implementiert werden. Die Implementierung der EP-Tableaux-Methode im System SIC stützt sich auf FINFIMO, benutzt aber ein anderes Suchverfahren.

Im zweiten Abschnitt dieses Kapitels wird eine Variante der PRQ-Syntax beschrieben. Diese Variante der PRQ-Syntax ist stark durch die Implementierung FINFIMO motiviert und soll deshalb im Anschluß an diese beschrieben werden.

Die Bezeichnung *Spezifikation* wird im folgenden immer für eine Menge von geschlossenen PRQ-Formeln verwendet.

5.1 Das Programm FINFIMO

Das Programm FINFIMO, welches in Abbildung 5.2 und 5.3 dargestellt ist, entspricht im Kern der fairen Version des SATCHMO-Programms (siehe Abschnitt 3.3). Aufgrund der erweiterten Syntax sind aber natürlich wesentlich mehr Fälle zu betrachten, was zu einer Verlängerung des Programms führt.

FINFIMO erzeugt die Zweige eines fairen EP-Tableau mittels Tiefensuche. Genau wie bei SATCHMO wird die Prolog-eigene Tiefensuche ausgenutzt, was auch hier zu einem sehr kurzen Programm führt.

Die wesentliche Idee, die dem Programm zugrunde liegt, ist folgende: Eine gegebene Spezifikation wird gemäß den EP-Tableaux-Expansionsregeln behandelt. Die Grundatome der Spezifikation und die bei der Expansion entstehenden Grundatome werden direkt in die dynamische Datenbasis des Prologsystems eingeführt. Auf diese Weise repräsentiert die Datenbasis zu jedem Zeitpunkt des Programmablaufs eine Terminterpretation.

Die Knoten eines EP-Tableau bestehen aus PRQ-Formeln. Diese werden folgendermaßen durch Prologterme dargestellt:

1. Gemäß der Prologkonvention werden Relationen und Konstanten durch Bezeichner repräsentiert, die mit Kleinbuchstaben beginnen. Variablen werden durch Bezeichner repräsentiert, die mit Großbuchstaben beginnen.

2. Gemäß der Prologkonvention wird die Konjunktion durch Komma und die Disjunktion durch Semikolon dargestellt.

3. Eine existentiell quantifizierte Formel $\exists x\ (p(x) \wedge q)$ wird durch den Prologterm `exists(X, (p(X), q))` dargestellt. Das zweite Argument dieses Prologterms ist eine Konjunktion in Prolog-Darstellung.

4. Eine universell quantifizierte Formel $\forall x_1 x_2 x_3\ (p \rightarrow q)$ wird durch den Prologterm `all([X1, X2, X3], p => q)` dargestellt. Dabei ist `=>` ein zweistelliger binärer Operator, der für die Darstellung der Implikation deklariert wurde. Wird nur eine einzige Variable quantifiziert, so kann die einelementige Variablenliste durch die Variable selbst ersetzt werden.

Nachdem die Darstellung der PRQ-Formeln beschrieben wurde, stellt sich jetzt die Frage nach der Repräsentation des gesamten EP-Tableau. In FINFIMO wird ein EP-Tableau durch Datenstrukturen und interne Steuerinformationen des Prologsystems dargestellt. So werden etwa Fallunterscheidungen durch Prolog-Choice-Points dargestellt.

Bevor die verwendeten Datenstrukturen genau beschrieben werden, muß zunächst geklärt werden, welche Komponenten eines EP-Tableau dargestellt werden müssen.

Die Wurzel eines EP-Tableau besteht aus einer Menge von geschlossenen PRQ-Formeln, also der Spezifikation. Für die Darstellung der Spezifikation wird zwischen Grundatomen, komplexen Formeln und Konstanten unterschieden. Die Grundatome werden gesondert behandelt, da sie zu jedem Zeitpunkt der Berechnung eine Terminterpretation repräsentieren. Die Konstanten sind von besonderer Bedeutung, da auf sie bei jeder Behandlung einer existentiell quantifizierten Formel zurückgegriffen werden muß.

Die Knoten eines EP-Tableau bestehen aus der Spezifikation und aus den hergeleiteten Formeln. Für die Darstellung der hergeleiteten Formeln wird unterschieden zwischen Grundatomen, universellen Formeln, sonstigen komplexen Formeln und Konstanten. Die hergeleiteten universellen Formeln haben deshalb eine Sonderrolle, weil ein einmaliges Erfüllen einer universellen Formel nicht gewährleistet, daß die Formel auch erfüllt bleibt. Deshalb

	Formeln der Spezifikation	Hergeleitete Formeln
Grundatom	`axiom` und Grundatom in der dynamischen Datenbasis	Grundatom in der dynamischen Datenbasis
universelle Formel	`axiom`	`axiom` und Prologsteuerung
sonst. kompl. Formel	`axiom`	Prologsteuerung
Konstante c in	`axiom(dom(c))` und `dom(c)` in der dynamischen Datenbasis	`dom(c)` in der dynamischen Datenbasis

Abbildung 5.1: Verwendete Datenstrukturen und Darstellungen in FINFIMO.

muß immer wieder getestet werden, ob eine universelle Formel noch erfüllt ist. Aus diesem Grund wird für die universellen Formeln eine andere Datenstruktur gewählt als für die anderen hergeleiteten Formeln.

Abbildung 5.1 gibt eine Übersicht über die verwendeten Datenstrukturen bzw. Darstellungen. Jede Formel F der Spezifikation wird durch einen Prologterm der Form `axiom(F)` repräsentiert. Dies gilt für Grundatome, universelle und sonstige Formeln gleichermaßen. Wenn wir uns aber an die Definition eines EP-Tableau erinnern, so besteht die Wurzel des EP-Tableau aus der Spezifikation. Weiter repräsentieren die Grundatome eines Zweiges, insbesondere auch die der Wurzel, eine Terminterpretation. In der Implementierung der EP-Tableaux-Methode wird diese Terminterpretation durch die Grundatome in der dynamischen Prologdatenbasis repräsentiert. Deshalb muß sichergestellt werden, daß alle Grundatome in einem aktuellen Zweig in der Prologdatenbasis enthalten sind. Dies gilt insbesondere auch für die Grundatome, die bereits in der Spezifikation enthalten sind. Aus diesem Grund werden die Grundatome der Spezifikation zusätzlich zu ihrer Repräsentation mittels `axiom` direkt in die Prologdatenbasis eingegeben.

Dagegen werden die hergeleiteten Grundatome nur direkt in die Prologdatenbasis eingegeben. Die hergeleiteten universellen Formeln haben, wie bereits erwähnt, deshalb eine Sonderrolle, weil gewährleistet werden muß, daß sie während der Expansion erfüllt bleiben. Angenommen, ein Zweig in einem EP-Tableau enthält die hergeleitete Formel $\forall x\ (p(x) \rightarrow q(x))$. Repräsentiert der Zweig die Terminterpretation $\mathcal{T}(\{p(a), q(a)\})$, so wird $\forall x\ (p(x) \rightarrow q(x))$ von dieser erfüllt. Wird bei der weiteren Expansion des EP-Tableau das Atom $p(b)$ erzeugt, so erfüllt die durch den Zweig repräsentierte Terminterpretation die Formel $\forall x\ (p(x) \rightarrow q(x))$ nicht mehr. Aus diesem Grund werden die hergeleiteten universellen Formeln wie Formeln der Spezifikation behandelt und entsprechend durch `axiom` dargestellt. Dadurch wird immer wieder getestet, ob sie noch erfüllt sind. Zusätzlich dazu werden sie wie alle anderen hergeleiteten Formeln durch interne Steuerinformationen des Prologsystems repräsentiert.

Um die Konstanten eines EP-Tableau zu verwalten, wird das Prologprädikat `dom` verwendet. Die in der Spezifikation auftretenden Konstanten c werden durch einen Prologterm `axiom(dom(c))` repräsentiert. Um auf die Konstanten in einem Zweig zurückzugreifen,

wird das Prologprädikat `dom` aufgerufen. Dazu ist es aber notwendig, daß alle im Zweig auftretenden Konstanten c zu jedem Zeitpunkt „abrufbar“ sind, d.h. `dom(c)` muß in der dynamischen Datenbasis enthalten sein. Dies gilt, ähnlich wie bei den Grundatomen, insbesondere auch für die Konstanten, die in der Spezifikation auftreten. Aus diesem Grund werden sie zusätzlich als Prologfaktum `dom(c)` in die dynamische Datenbasis eingegeben. Dagegen werden die aufgrund der Existenzregel neu generierten Konstanten c_{neu} nur direkt mittels der Prologfakten `dom(c_neu)` in der dynamischen Datenbasis verwaltet.

Zu der Konstantenverwaltung mittels des Prologprädikats `dom` sei noch folgendes erwähnt: Die Formeln `dom(c)`, die für jede Konstante c hinzugenommen werden, dienen allein dazu, die Konstanten in der Datenbasis zu verwalten. Es ist wichtig zu sehen, daß sie nur ein Hilfsmittel zur Konstantenverwaltung sind. Dies erlaubt eine einfache und auch sich selbst erklärende Möglichkeit der Konstantenverwaltung. In Abschnitt 5.2 wird eine alternative Definition der PRQ-Syntax vorgestellt, bei der die Konstantenverwaltung direkt in die Syntax integriert wird.

Beispiel 5.1 Für die Menge $\mathcal{S} = \{p(a),\ q(a),\ \forall x((p(x) \wedge q(x)) \rightarrow r(x) \vee \exists y(s(x,y) \wedge t(y)))\}$ erwartet FINFIMO folgende Eingabe:

```
axiom(p(a)).
axiom(q(a)).
axiom(all(X, (p(X), q(X)) => (r(X); exists(Y, (s(X,Y), t(Y)))))).
p(a).
q(a).
axiom(dom(a)).
dom(a).
```

Selbstverständlich ist hier auch eine Automatisierung möglich, so daß als Eingabe

```
p(a)
q(a)
all(X, (p(X), q(X)) => (r(X); exists(Y, (s(X,Y), t(Y)))))
```

ausreicht.

Nach dieser Vorarbeit kann das eigentliche Programm FINFIMO beschrieben werden.

Wie bereits erwähnt, erzeugt FINFIMO die Zweige eines fairen EP-Tableau. Den Kern des Programms bildet das Prologprädikat `satisfiable`. Durch Zurücksetzen über `satisfiable` werden Termmodelle der gegebenen Spezifikation erzeugt, die durch eine Menge $\mathcal{G}$ von Grundatomen in der dynamischen Datenbasis des Prologsystems repräsentiert werden. Die Elemente der Menge $\mathcal{G}$ werden durch Rücksetzen über das Prologprädikat `assume` sowohl in die dynamische Datenbasis eingefügt als auch wieder herausgenommen.

Die Fairness wird durch den Aufruf des Prolog built-in-Prädikates `setof` gewährleistet. Das Prologprädikat `violated_instance` gibt bei jedem Aufruf eine Instanz einer Formel oder Teilformel zurück, die von der momentanen Datenbasis nicht erfüllt ist und deshalb

erfüllt werden muß. Durch das Prologprädikat `setof` werden also alle zu diesem Zeitpunkt nicht erfüllten Instanzen von (Teil-)formeln zurückgegeben, und so wird die Fairness gesichert.

Das Prologprädikat `satisfy` veranlaßt das Einfügen von Grundatomen, um die verletzten Instanzen durch den so erhaltenen Datenbankzustand zu erfüllen. `satisfy` führt eine Fallunterscheidung bezüglich der syntaktischen Form der gegebenen Instanzen durch. Da in einem einzigen `satisfiable`-Schritt eine Menge von Instanzen erfüllt werden (dies ist notwendig, um die Fairness zu sichern), kann folgendes passieren: Durch das Erfüllen einer oder mehrerer verletzten Instanzen werden gleichzeitig auch andere vorher verletzte Instanzen miterfüllt. Damit diese miterfüllten Instanzen nicht ein zweites Mal erfüllt werden, ist die erste Prologklausel von `satisfy` ein Test, ob die betreffende Instanz tatsächlich noch unerfüllt ist. Im Fall einer Disjunktion oder einer existentiell quantifizierten Formel gibt es verschiedene Möglichkeiten, die Formel zu erfüllen. Das Prologprädikat `satisfy` gibt hier alle Möglichkeiten durch Zurücksetzen zurück. Im existentiellen Fall wird die Wiederverwendung alter Konstanten (wie in Definition 4.9 beschrieben) dadurch ausgeführt, daß das Prologprädikat `dom` aufgerufen wird, welches Zurücksetzen erlaubt. Die Einführung einer neuen Konstante wird durch das Prologprädikat `new_constant` ausgeführt. Für eine universelle Formel F wird mittels `assume` der Prologterm `axiom(F)` in die dynamische Datenbasis eingeführt. Damit werden die universellen Teilformeln, wie bereits erklärt, wie die Formeln der Spezifikation behandelt. Allerdings werden die hergeleiteten universellen Teilformeln (im Gegensatz zu Formeln der Spezifikation) auch mittels `assume` wieder aus der Datenbasis herausgenommen, falls der expandierte Zweig nicht zu einem Modell führt. Ein besonderer Fall ist der atomare Fall. Das Prologprädikat `satisfy` unterscheidet im atomaren Fall atomare (Teil-)formeln der Spezifikation von sogenannten `built_in`-Prädikaten. Die atomaren (Teil-)formeln der Spezifikation werden dadurch erfüllt, daß sie in die Datenbasis aufgenommen werden. Der Aufruf von `satisfy` scheitert, falls das Atom `false` erfüllt werden soll. Die `built_in`-Prädikate sind dagegen solche Prädikate, die eine vordefinierte Semantik wie etwa die Gleichheit besitzen. Sie können von FINFIMO an sich nicht behandelt werden können. Jedoch können sie getestet werden. Diese `built_in`-Prädikate werden also nicht erfüllt in dem Sinn, daß Instanzen in die Datenbasis eingefügt werden, sondern sie werden nur getestet. Schlägt der Test fehl, so scheitert der Aufruf von `satisfy`. Dies entspricht dem Vorgehen bei der Behandlung von „normalen“ atomaren Formeln. In diesem Fall scheitert der Aufruf von `satisfy`, wenn das Atom `false` erfüllt werden soll.

Auf diese Weise können Formeln wie etwa $\forall x \forall y \ (q(x, y) \rightarrow x = y)$ behandelt werden. Besonders im Fall der Gleichheit bleibt aber auf der theoretischen Ebene noch genau zu untersuchen, für welche Anwendungen diese Art von „eingeschränkter Gleichheit“ [BK98] sinnvoll oder sogar ausreichend ist (siehe Kapitel 8).

Nachdem das FINFIMO-Programm vorgestellt wurde, sollen an dieser Stelle einzelne Teile noch etwas genauer betrachtet und geringe Abwandlungen diskutiert werden.

Sehen wir uns noch einmal die Definition von `satisfiable` an. Um die Fairness zu gewährleisten, wird das Prologprädikat `setof` eingesetzt. Genauso könnte an dieser Stelle auch das Prologprädikat `findall` verwendet werden. `findall` gibt im Gegensatz zu `setof`

```
satisfiable :-
        setof(F, violated_instance(F), F_List),
        !,
        satisfy_list(F_List),
        satisfiable.
satisfiable.

violated_instance(A) :-
        axiom(A),
        atomic_formula(A),
        not A.
violated_instance(A) :-
        axiom((A, _B)),
        not A.
violated_instance(B) :-
        axiom((_A, B)),
        not B.
violated_instance((A ; B)) :-
        axiom((A ; B)),
        not (A ; B).
violated_instance(exists(Var, (R, F))) :-
        axiom(exists(Var, (R, F))),
        not (R, F).
violated_instance(F) :-
        axiom(all(_VarList, R => F)),
        R,
        not F.

satisfy_list([]).
satisfy_list([H | T]) :-
        satisfy(H),
        satisfy_list(T).

satisfy(F) :-
        F,
        !.
satisfy(exists(Var, (R, F))) :-
        dom(Var),
        satisfy(R),
        satisfy(F).
satisfy(exists(Var, (R, F))) :-
        !,
        new_constant(Var),
        assume(dom(Var)),
        satisfy(R),
        satisfy(F).
```

Abbildung 5.2: Das FINFIMO-Programm, Teil 1

```
satisfy(all(VarList, R => F)) :-
        !,
        assume(axiom(all(VarList, R => F))),
        findall(F, (R, not F), L),
        satisfy_list(L).
satisfy((E1, E2)) :-
        !,
        satisfy(E1),
        satisfy(E2).
satisfy((E1 ; E2)) :-
        !,
        (satisfy(E1) ; satisfy(E2)).
satisfy(Atom) :-
        not Atom = false,
        functor(Atom,F,A),
        (built_in(F,A) ->
            Atom
        ;   assume(Atom)).

built_in(=,2).

assume(Atom) :-
        Atom,
        !.
assume(Atom) :-
        asserta(Atom).
assume(Atom) :-
        once(retract(Atom)),
        fail.

new_constant(X1) :-
        retract(next_constant(X1)),
        X2 is X1 + 1,
        asserta(next_constant(X2)).

next_constant(1).

all(_VarList, R => F) :-
        not (R, not F).
exists(_Var, (R, F)) :-
        (R, F).

atomic_formula(E) :-
        not E = (R => F),
        not E = all(_VarList, R => F),
        not E = exists(_Var, (R, F)),
        not E = (A, B),
        not E = (A ; B).
```

Abbildung 5.3: Das FINFIMO-Programm, Teil 2

eine ungeordnete Liste von Instanzen zurück. Andererseits ist der Aufruf von `findall` auch dann erfolgreich, wenn die Liste der Instanzen leer ist. Deshalb müsste, falls `setof` durch `findall` ersetzt werden würde, zusätzlich getestet werden, ob die Liste nicht leer ist. Zudem kann `findall` auch eine Liste mit Duplikaten zurückgeben. Aus diesem Grund fiel die Entscheidung zugunsten des Prologprädikats `setof`.

Bei der Definition von `violated_instance` fällt auf, daß die Konjunktion und die Disjunktion nicht symmetrisch behandelt werden. Als erstes stellt sich die Frage, warum die Konjunktion nicht entsprechend wie die Disjunktion durch

```
violated_instance((A , B)) :-
        axiom((A , B)),
        not (A , B).
```

behandelt wird. Natürlich wäre dies möglich. Allerdings ist diese Variante für den Fall, daß nur `A` bzw. nur `B` vom momentanen Datenbankzustand nicht erfüllt ist, ungünstig. Durch `violated_instance` wird die vollständige Konjunktion `(A , B)` zurückgegeben. Im nachfolgenden Aufruf von `satisfy` werden dann also `A` und `B` erfüllt. Das bedeutet aber, daß das Konjunktionsglied, das bereits erfüllt ist, überflüssigerweise noch einmal getestet wird. Dieser Test kann dadurch eingespart werden, daß bei der Konjunktion nur das Konjunktionsglied zurückgegeben wird, welches verletzt ist:

```
violated_instance(A) :-
        axiom((A, _B)),
        not A.
violated_instance(B) :-
        axiom((_A, B)),
        not B.
```

Sind beide Konjunktionsglieder verletzt, so feuern beide Prologklauseln und beide Konjunktionsglieder werden in die Liste der verletzten Formeln aufgenommen.

Ist dagegen eine Disjunktion verletzt, so sind beide Disjunktionsglieder verletzt. Deshalb ist es in diesem Fall notwendig, immer beide Konjunktionsglieder im gleichen Aufruf von `violated_instance` zu testen und zurückzugeben:

```
violated_instance((A ; B)) :-
        axiom((A ; B)),
        not (A ; B).
```

5.2 Skizze einer alternativen Definition der PRQ-Syntax und ihrer Konsequenzen

Der folgende Abschnitt ist einer Variante der PRQ-Syntax gewidmet. Motiviert wurde diese Modifizierung der PRQ-Syntax durch die Implementierung der EP-Tableaux-Methode.

Die Veränderung der Syntax führt schließlich auch zu einer geringen Abänderung der Definition der EP-Tableaux.

Wir haben im letzten Abschnitt gesehen, daß in der Implementierung der EP-Tableaux-Methode ein Prologprädikat `dom` dazu verwendet wird, die Konstanten, die in einem Zweig des EP-Tableau auftreten, d.h. also die Domänenelemente der durch den Zweig repräsentierten Terminterpretation, zu verwalten. Ausgehend von der Implementierung stellt sich dann aber die Frage, ob eine Verwaltung der Domänenelemente nicht auch direkt in die zugrundegelegte Syntax integriert werden kann. Diese Überlegung führt zu der im folgenden beschriebenen Variante der PRQ-Syntax.

Die wesentliche Idee dieser Syntaxvariante besteht darin, mittels der Syntax der zugelassenen Formeln zu erzwingen, daß die Domäne eines Termmodells einer gegebenen Spezifikation explizit dargestellt wird.

Erinnern wir uns an die Transformation (siehe Abschnitt 4.3, Definition 4.23) einer beliebigen Formel der Prädikatenlogik erster Stufe in die PRQ-Syntax. Hier wurde ein neues, nicht in der zugrundeliegenden Sprache auftretendes, Relationssymbol eingeführt und dazu verwendet, die Domäne zu repräsentieren.

Die gleiche Idee wird nun für die Definition einer Variante der PRQ-Syntax verwendet. Auch hier wird ein neues Relationssymbol eingeführt, um die Domäne zu repräsentieren. Anstelle des allgemein definierten Bereichs einer Variable wird in existentiell quantifizierten Formeln nur die Form $\exists x\ (D(x) \wedge F)$ gefordert, wobei D das neu eingeführte Relationssymbol und F wieder eine Formel in der zugelassenen Syntax sei. Das Atom $D(x)$ übernimmt also die Rolle des Bereichs B in den existentiell quantifizierten PRQ-Formeln, ist aber eine Art Metasymbol. In allen anderen Fällen bleibt die Definiton die gleiche wie in Abschnitt 4.3. Hier stellt sich natürlich die Frage, weshalb nicht auch die Definition der universell quantifizierten Formeln abgeändert werden sollte zu $\forall x\ (D(x) \rightarrow F)$, so daß auch hier das Atom $D(x)$ die Rolle des Bereiches übernimmt. In diesem Fall bringt aber der Übergang von einem Bereich zu der Relation D in manchen Fällen mehr Tests mit sich. Gehen wir von der Form $\forall x\ (D(x) \rightarrow F)$ aus, so muß für alle Domänenelemente e getestet werden, ob $F\{e/x\}$ erfüllt ist. Gehen wir von Formeln der Form $\forall x\ (B(x) \rightarrow F)$ aus, wobei $B(x)$ ein Bereich für x sei, so ist die Menge der Elemente, für die $B(x)$ bereits erfüllt ist, möglicherweise nur eine Teilmenge der Domäne. In diesem Fall muß dann auch für entsprechend weniger Elemente e überprüft werden, ob $F\{e/x\}$ erfüllt ist. Im existentiellen Fall wird dagegen die Anzahl der möglichen Belegungen durch den Bereich nicht eingeschränkt. Es werden in jedem Fall alle zu diesem Zeitpunkt der Berechnung zur Verfügung stehenden Konstanten in Betracht gezogen (siehe $\exists$-Regel). Aus diesem Grund unterscheidet sich die Definition der existentiell quantifizierten und der universell quantifizierten Formeln.

Die Definition der EP-Tableaux wird nun folgendermaßen abgeändert: Im Gegensatz zu den EP-Tableaux, wie sie in Abschnitt 4.4 definiert wurden, besteht die Wurzel eines variierten EP-Tableau nicht nur aus der gegebenen Formelmenge $\mathcal{S}$ (jetzt in der variierten PRQ-Syntax), sondern zusätzlich aus der Menge $\mathcal{C}(\mathcal{S}) := \{D(c) \mid c \text{ Konstanten in } \mathcal{S}\}$, falls es Konstanten in $\mathcal{S}$ gibt, sonst $\mathcal{C}(\mathcal{S}) := \{D(c_0)\}$ für eine beliebige Konstante c_0. Aufgrund dieser Änderung wird sichergestellt, daß alle Konstanten, die in der gegebenen Formelmenge $\mathcal{S}$ auftreten, in D repräsentiert werden. Die einzigen Konstanten, die außer diesen

Konstanten im Tableau auftreten können, werden aufgrund einer Anwendung der ∃-Regel eingeführt. Da aber die existentiell quantifizierten Formeln in der variierten PRQ-Syntax die Form $\exists x\ (D(x) \wedge F)$ besitzen, ist auch für die neu eingefügten Konstanten sichergestellt, daß sie in D repräsentiert werden. Wir können nun die ∃-Regel folgendermaßen variieren:

variierte ∃-Regel:

$$\frac{\exists x(D(x) \wedge E(x))}{E(c_1) \mid \ldots \mid E(c_k) \mid \begin{array}{l} E(c_{neu}) \\ D(c_{neu}) \end{array}}$$

wobei $\{c_1, \ldots, c_k\}$ die Menge aller Konstanten ist, die im expandierten Knoten vorkommen und c_{neu} eine von c_i für $i = 1, \ldots, k$ verschiedene Konstante ist.

Streng genommen ist diese Variante der ∃-Regel nicht notwendig, aber in den Zweigen, in denen „alte“ Konstanten wiederverwendet werden, wird damit eine überflüssige Anwendung der ∧-Regel eingespart, da die „alten“ Konstanten bereits in der Relation D enthalten sind.

Nach dieser informellen Beschreibung der Variante der PRQ-Syntax stellt sich die Frage nach den Vor- und Nachteilen dieser Herangehensweise.

Gehen wir von der Implementierung aus, so ist die Variante der PRQ-Syntax aus folgenden Gründen näher an der Implementierung als die PRQ-Syntax. Betrachten wir noch einmal die in Abschnitt 4.4 in Definition 4.9 definierte ∃-Regel. Bei Anwendung dieser Regel muß auf Konstanten zurückgegriffen werden, „die im expandierten Knoten vorkommen“. Das heißt für die Implementierung aber, daß eine Menge von Konstanten verwaltet werden und ein Zugriff auf die Menge möglich sein muß. Wie wir in Abschnitt 5.1 gesehen haben, wird zu diesem Zwecke in der Implementierung das zusätzliche Prädikat `dom` verwendet, und jede Konstante wird mittels dieses Prädikats in der Datenbasis verwaltet. Wird im Falle der ∃-Regel eine neue Konstante benötigt, so wird auch diese mittels des Prädikats `dom` bei der Behandlung einer existentiell quantifizierten Formel in die Datenbasis eingefügt. Dies entspricht aber genau der oben beschriebenen variierten ∃-Regel.

Durch die Verwendung der alternativen PRQ-Syntax kann die Definition einer Terminterpretation ein wenig vereinfacht werden. Bei der Definition einer Terminterpretation in Kapitel 2, Abschnitt 2.3 wurde festgelegt, daß die Domäne zusätzlich zu den Konstanten, die in der Menge der erfüllten Grundatome auftreten, ein Element enthält. Wie in Beispiel 2.36 gezeigt wurde, war dies notwendig, um zu gewährleisten, daß alle Konstanten, die in einer Spezifikation auftreten, interpretiert werden. In der oben beschriebenen Variante wird die Forderung nach einem Zusatzelement überflüssig, da ohnehin alle Konstanten, die in der Spezifikation auftreten, auch in der Relation D und damit in der Menge der erfüllten Grundatome auftreten. Dies wird durch die zusätzlich zur Spezifikation betrachtete Menge $\mathcal{C}(\mathcal{S}) := \{D(c) \mid c \text{ Konstanten in } \mathcal{S}\}$, falls es Konstanten in $\mathcal{S}$ gibt, sonst $\mathcal{C}(\mathcal{S}) := \{D(c_0)\}$ für eine beliebige Konstante c_0, erzwungen. Eine zusätzliche Konstante kommt in diesem

Fall nur deshalb ins Spiel, um zu verhindern, daß eine Interpretation mit leerer Domäne betrachtet wird.

Schließlich erinnern wir uns noch einmal an die Transformation beliebiger Formeln der Logik der ersten Stufe in die PRQ-Syntax (siehe Abschnitt 4.3). Hier wird ein „Metarelationssymbol“ D eingeführt, um die Domäne der Interpretation zu repräsentieren. Dieses Metarelationssymbol D entspricht dem Prädikatssymbol `dom` in der Implementierung, d.h. durch die Transformation beliebiger Formeln der ersten Stufe entstehen PRQ-Formeln, die gleichzeitig auch Formeln in alternativer PRQ-Syntax sind.

Trotzdem spricht einiges gegen die alternative PRQ-Syntax. So ist die angegebene Transformation von Formeln der Logik erster Stufe in PRQ-Formeln eine erste Lösung. Sie dient vornehmlich dem Beweis der Ausdrucksstärke der PRQ-Syntax. Es sind andere Transformationen denkbar, mittels derer so wenig wie möglich an den gegebenen Formeln der ersten Stufe verändert würde. Dann sind aber die durch die Transformation gewonnenen PRQ-Formeln nicht notwendigerweise auch gleichzeitig in der alternativen PRQ-Syntax.

Zudem war das Ziel bei der Definiton der PRQ-Syntax, im Rahmen der Bereichsbeschränkung möglichst viel an syntaktischer Variation zuzulassen, damit Spezifikationen überhaupt nicht von der Logik der ersten Stufe in die geforderte Syntax transformiert werden müssen, sondern vielmehr sofort in der geforderten Syntax formuliert werden können. Deshalb wurde auch der Bereich einer Variable möglichst allgemein definiert. Eine Reduzierung des Bereichs auf Atome der Form $D(x)$ im existentiellen Fall würde dagegen wieder eine Einschränkung bedeuten.

Ein wesentlicher Nachteil des alternativen Herangehens liegt darin, daß zweierlei Arten von Relationssymbolen auftauchen. Einmal gibt es die Relationssymbole, die in der zugrundeliegenden Sprache auftreten, und zum anderen gibt es dieses „Metarelationssymbol“ D, durch welches die Domäne repräsentiert wird. Beide Arten von Relationssymbolen werden zwar gleich behandelt, haben aber verschiedene Funktionen. Rein aus Sicht der Theorie scheint es auf jeden Fall wesentlich eleganter zu sein, nur mit einer Art von Relationen zu arbeiten.

Schließlich spricht auch einiges dagegen, daß eine Theorie der Implementierung folgen sollte. Die vorliegende Implementierung ist eine mögliche Implementierung der EP-Tableaux-Methode. Natürlich sind auch ganz andere Implementierungen denkbar. Würde die Theorie jetzt ausgehend von der Implementierung geändert, so wäre dies nur dann gerechtfertigt, wenn dadurch eine deutliche Vereinfachung der Theorie erzielt werden würde. Da dies hier nicht gegeben ist, würde so dieser prototypischen Implementierung ein zu hoher Stellenwert beigemessen.

Kapitel 6

Anwendungen der EP-Tableaux-Methode

Wie bereits erwähnt, wurde der Entwurf der EP-Tableaux-Methode durch verschiedene Anwendungen motiviert. In diesem Kapitel wird nun genauer untersucht, wie die EP-Tableaux-Methode in der Diagnose, beim Datenbankschemaentwurf sowie im Zusammenhang mit der natürlichen Sprache angewendet werden kann. Für alle diese Anwendungen ist dabei die Vollständigkeit der EP-Tableaux-Methode für die Erfüllbarkeit im Endlichen wesentlich. Für die Anwendung beim Datenbankschemaentwurf sowie in der Linguistik ist zusätzlich die Vollständigkeit der EP-Tableaux-Methode für die Widerlegung notwendig.

6.1 Diagnose

Stellen wir uns folgendes Szenario vor: Es ist erstens eine Beschreibung eines Systems und seines Verhaltens gegeben, und zweitens ist zu beobachten, daß sich das System anders verhält, als aufgrund der Beschreibung zu erwarten wäre.

Die Suche nach einer Erklärung für das unerwartete Verhalten des Systems wird als *Diagnoseproblem* bezeichnet. Für die formale Vorgehensweise ist hierbei wesentlich, welche Art von Wissen vorliegt. Folgende Möglichkeiten sind denkbar:

1. Über ein gegebenes System ist bekannt,

   - aus welchen Komponenten das System zusammengesetzt ist und
   - wie die Komponenten korrekt arbeiten.

Dagegen ist nicht bekannt,

   - welche Fehler im System auftreten können und
   - wie sie sich zeigen.

2. Über ein gegebenes System ist bekannt,

   - welche Komponenten fehlerhaft sein können und
   - welche Gründe für welche Symptome in Frage kommen.

In einer Reihe von Arbeiten wurden logische Beschreibungen des Diagnoseproblems vorgestellt [Rei87, CP87, Poo89b, BFFN97b]. Dabei werden vor allem zwei Ansätze betrachtet, die *Konsistenzbasierte Diagnose* und die *Abduktive Diagnose*. Die Wahl für einen der Ansätze ist oft durch das gegebene Wissen motiviert. Der Unterschied zwischen den beiden Ansätzen hat jedoch nichts mit obiger Unterscheidung bezüglich des bekannten Wissens zu tun. In [Poo89b] wird sogar gezeigt, daß unabhängig vom Wissen beide Ansätze verwendet werden können. Im folgenden wird vorrangig auf die konsistenzbasierte Diagnose eingegangen.

6.1.1 Konsistenzbasierte Diagnose

Das konsistenzbasierte Diagnoseverfahren besteht daraus, Abweichungen vom korrekten Verhalten eines Systems zu bestimmen. Nicht korrekt arbeitende Komponenten werden im folgenden als *Abnormitäten* bezeichnet. Die folgende Definition geht zurück auf [Rei87].

Definition 6.1 (Konsistenzbasierte Diagnose) Eine *konsistenzbasierte Diagnose* ist eine minimale[1] Menge von Abnormitäten, so daß die Beobachtungen zusammen mit der Menge von korrekt arbeitenden Komponenten des Systems konsistent sind. □

Bemerkung 6.2 Beim konsistenzbasierten Diagnoseverfahren folgt die Diagnose oder eine Disjunktion von Diagnosen aus den Beobachtungen.

Im folgenden werden wir anhand von zwei Beispielen sehen, wie das Diagnoseproblem, abhängig vom gegebenen Wissen, formalisiert werden kann.

Verfügen wir über ein Wissen über das gegebene System, wie es oben in Punkt 1 beschrieben wurde, so wird zur Formalisierung des Diagnoseproblems folgende Methode angewendet [Rei87]:

1. Für jede Komponente k eines Systems, die sich inkorrekt verhalten kann, wird $\neg ab(k)$ angenommen[2].

2. Das System wird durch geschlossene Implikationen beschrieben, wobei immer vom korrekten Verhalten des Systems ausgegangen wird. So ist es möglich, aus den Beobachtungen eine Disjunktion von inkorrektem Verhalten zu folgern.

[1] bzgl. der Mengeninklusion
[2] ab steht für „Abnormität“

Beispiel 6.3 Wir betrachten ein System, dessen Komponenten jeweils eine Kontrollampe besitzen. Bei korrektem Verhalten einer Komponente leuchtet deren Kontrollampe grün:

$$\forall x(\neg ab(x) \rightarrow \exists y\ (lampe(x,y) \wedge farbe(y,gruen))).$$

Jede Komponente besitzt nur eine Kontrollampe:

$$\forall x\ y\ z(lampe(x,y) \wedge lampe(x,z) \rightarrow y = z).$$

Die Lampen können nur grün oder rot leuchten:

$$\forall x\ y\ z\ (lampe(x,y) \wedge farbe(y,z) \rightarrow z = gruen\ \vee z = rot).$$

Bilden zwei Komponenten einen „Verbund“, so besitzt der Verbund eine eigene Kontrollampe deren Farbe abhängig von der Farbe der Lampen der Komponenten ist:

$$\begin{array}{l}\forall x\ x_1\ x_2\ y\ z\ (verbund(x,x_1,x_2) \wedge lampe(x,y) \wedge farbe(y,z) \rightarrow \\ \quad \exists y_1\ (lampe(x_1,y_1) \wedge \exists y_2\ (lampe(x_2,y_2) \wedge \\ \qquad (z = gruen \rightarrow farbe(y_1,gruen) \wedge farbe(y_2,gruen)) \wedge \\ \qquad (z = rot \rightarrow farbe(y_1,rot) \vee farbe(y_2,rot)))).\end{array}$$

Beobachten wir nun, daß die Lampe l des Verbundes k, der aus den Komponenten k_1 und k_2 besteht, rot aufleuchtet

$$verbund(k,k_1,k_2) \wedge lampe(k,l) \wedge farbe(l,rot),$$

und nehmen wir $\neg ab(k_1)$ an, so erhalten wir die Diagnose $ab(k_2)$. Nehmen wir dagegen $\neg ab(k_2)$ an, so erhalten wir die Diagnose $ab(k_1)$.

Steht uns dagegen Wissen über das gegebene System wie oben in Punkt 2 beschrieben zur Verfügung, so ergibt die Methode von [Rei87]:

1. Es wird angenommen, daß keine fehlerhaften Komponenten existieren, solange deren Existenz nicht gezeigt werden kann.

2. Ausgehend von den Symptomen wird formalisiert, welche Gründe für ein Symptom in Frage kommen, d.h. Symptome implizieren eine Disjunktion von Gründen.

Beispiel 6.4 Stellen wir uns ein System mit zwei Komponenten $kom(a)$ und $kom(b)$ vor:

$$\forall x(kom(x) \rightarrow (x = a \vee x = b)).$$

Falls eine Komponente hupt, deutet das darauf hin, daß mindestens eine Komponente des Systems fehlerhaft arbeitet:

$$\forall x(hupt(x) \rightarrow \exists y(kom(y) \wedge ab(y))).$$

Wir beobachten, daß $hupt(a)$ gilt. Das Symptom $hupt(a)$ impliziert dann mögliche Gründe. In unserem Fall kommen zwei Gründe in Frage, und wir erhalten zwei Diagnosen $ab(a)$ oder $ab(b)$.

6.1.2 Abduktive Diagnose

Der Vollständigkeit halber wird an dieser Stelle auch das abduktive Diagnoseverfahren definiert und auf den Unterschied zum konsistenzbasierten Diagnoseverfahren hingewiesen.

Definition 6.5 (Abduktive Diagnose) Eine *abduktive Diagnose* ist eine minimale Menge von Annahmen, so daß die Beobachtungen aus der Vereinigung von Diagnose und Hintergrundwissen folgt [PGA87, Poo89a]. □

Bemerkung 6.6 Beim abduktiven Diagnoseverfahren sind die Beobachtungen logische Folgerungen aus der Diagnose im Gegensatz zum konsistenzbasierten Diagnoseverfahren (vgl. Bemerkung 6.2).

6.1.3 Diagnose und EP-Tableaux-Methode

In Beispiel 6.3 und Beispiel 6.4 haben wir gesehen wie das Diagnoseproblem durch Formeln der Prädikatenlogik der ersten Stufe formalisierbar ist. Die Suche nach einer Diagnose entspricht der Suche nach einem Modell der Regeln und Beobachtungen. Daher ist es naheliegend Modellgenerierungsverfahren, insbesondere Tableauxverfahren, in der Diagnose einzusetzen. Dies wurde u.a. bereits in [FN92] und [BFFN97b] beschrieben. In [FN92] wird das System MOMO vorgestellt, dessen Kern der Modellgenerator SATCHMO ist. Für die Diagnose ist es allerdings wesentlich, möglichst kleine Modelle zu finden, da einfache Erklärungen den Erklärungen mit überflüssiger Information vorzuziehen sind. Dieses Prinzip ist als „Occam'sches Prinzip“ bekannt.

Um möglichst kleine Modelle, insbesondere endliche Modelle zu finden, ist der Modellgenerierer SATCHMO nicht geeignet. In Kapitel 4 wurde auf diese Problematik ausführlich eingegangen. Die EP-Tableaux-Methode bietet sich dagegen an, in der Diagnose eingesetzt zu werden, da sie vollständig für die Erfüllbarkeit im Endlichen ist. Also wird eine endliche Erklärung gefunden, sofern eine existiert.

Gleichzeitig stellt die EP-Tableaux-Methode eine flexiblere Sprache zur Verfügung. Die Formeln in Beispiel 6.4 sind bereits in PRQ-Syntax, sehen wir einmal von der Gleichheit ab. Allerdings handelt es sich hier wieder um die eingeschränkte Gleichheit [BK98], wie sie in Abschnitt 5.1 beschrieben wurde. Sie kann von FINFIMO behandelt werden.

Die Formeln in Beispiel 6.4 können leicht in die PRQ-Syntax gebracht werden, falls ein Relationssymbol *kom* für alle Komponenten eingeführt wird. Statt der Formel

$$\forall x(\neg ab(x) \rightarrow \exists y\ (lampe(x,y) \wedge farbe(y,gruen)))$$

betrachten wir dann

$$\forall x(kom(x) \rightarrow ab(x) \vee \exists y\ (lampe(x,y) \wedge farbe(y,gruen))).$$

Wird die EP-Tableaux-Methode auf die beiden Beispiele zusammen mit den Beobachtungen angewendet, werden jeweils Modelle generiert, in denen die Diagnosen erfüllt sind.

Wie bereits erwähnt, sind bei der Diagnose möglichst einfache Erklärungen, d.h. minimale Modelle, von Interesse. Deshalb ist es besonders wünschenswert, die MM-Prozedur, wie sie in Abschnitt 3.4 beschrieben wurde, auch auf die EP-Tableaux-Methode zu übertragen. Mit dieser Erweiterung würde die EP-Tableaux-Methode ein äußerst geeignetes Werkzeug für die Diagnose darstellen. Aufgrund der Vollständigkeit für die Erfüllbarkeit im Endlichen werden endliche Erklärungen gefunden, sofern es sie gibt. Zusätzlich würden aufgrund der MM-Prozedur nur die einfachen Erklärungen erzeugt.

Auf einen weiteren Punkt sei an dieser Stelle noch hingewiesen. Erinnern wir uns an die Definition 6.1 der konsistenzbasierten Diagnose, in der die konsistenzbasierte Diagnose als minimale Menge von Abnormitäten definiert wurde. Wird diese Definition auf den Modellbegriff übertragen, so ist die Suche nach einer Diagnose die Suche nach einem minimalen Modell der Regeln und Beobachtungen. Die Minimalität bezieht sich dabei allein auf die Menge der Abnormitäten, d.h. auf die Relation *ab*. Der Minimalitätsbegriff, der dieser Arbeit zugrunde liegt, bezieht sich dagegen auf alle Relationen.

Diese Beobachtung deutet darauf hin, daß für verschiedene Anwendungen auch verschiedene Minimalitätsbegriffe [Eis98] sinnvoll sind. Diese Frage wurde noch nicht untersucht und geht über den Rahmen dieser Arbeit hinaus.

6.2 Datenbankschemaentwurf

Eine Datenbank besteht aus einem *Schema* und den eigentlichen Daten, dem *Zustand*. Das Schema besteht aus einer Definition der Struktur der Daten und den *Integritätsbedingungen*. Diese Integritätsbedingungen sind Formeln der Prädikatenlogik der ersten Stufe und müssen von jedem zulässigen Zustand der Datenbank erfüllt werden.

Integritätsbedingungen werfen zwei grundsätzlich verschiedene Fragen auf. Zum einen stellt sich die Frage der *Erfülltheit* der Integritätsbedingungen: Sind in einem gegebenen Datenbankzustand alle Integritätsbedingungen erfüllt? Zum anderen stellt sich die Frage der *Erfüllbarkeit* der Integritätsbedingungen: Gibt es überhaupt einen Datenbankzustand, der die Integritätsbedingungen erfüllt?

Diese zweite Frage sollte geklärt werden, bevor Daten in eine Datenbank eingefügt werden, also bereits während des Datenbankschemaentwurfs. Modellgenerierungsverfahren können hierbei als geeignetes Hilfsmittel eingesetzt werden. Da allerdings die zulässigen Datenbankzustände i.a. endlich sein müssen, muß das eingesetzte Modellgenerierungsverfahren nicht nur vollständig für die Unerfülltheit, sondern auch vollständig für die Erfüllbarkeit im Endlichen sein. Die EP-Tableaux-Methode bietet diese Eigenschaften.

In Kapitel 7 wird ausführlich auf diesen Themenbereich eingegangen und der Prototyp SIC (Satisfiable Integrity Constraints) [BEST98a, BEST98c, BEST98b] vorgestellt. SIC basiert auf der EP-Tableaux-Methode und wurde zur Unterstützung des Datenbankschemaentwurfs entworfen.

6.3 Natürliche Sprache

Verbindungen zwischen der natürlichen Sprache und der EP-Tableaux-Methode können auf verschiedenen Wegen hergestellt werden.
1. Zum einen wäre es wünschenswert, eine Art „natürlichsprachliche Schnittstelle“ für Implementierungen der EP-Tableaux-Methode zur Verfügung zu haben. Die Syntax, die die EP-Tableaux-Methode fordert, ist ein Fragment der Prädikatenlogik der ersten Stufe, die sogenannte PRQ-Syntax, wie sie in Abschnitt 4.3 vorgestellt wurde. Im Hinblick auf die oben erwähnten Anwendungen – die Diagnose und den Datenbankschemaentwurf – wäre dagegen eine natürlichsprachliche Eingabe wesentlich bequemer. Andererseits birgt die Verarbeitung der natürlichen Sprache aufgrund von Mehrdeutigkeiten und des Umfangs des Vokabulars sowie ihrer komplexen Grammatik schwierige Probleme. Diese Überlegungen führen zu den sogenannten kontrollierten Sprachen, insbesondere zu der kontrollierten englischen Sprache ACE (Attempto Controlled English) [FS96]. Der Einsatz der kontrollierten Sprache ACE scheint in diesem Zusammenhang aufgrund ihrer eingeschränkten Form möglich. Auch wenn die Verwendung einer kontrollierten Sprache eine Einschränkung auf Fachtexte in bekannten Bereichen oder einfache Beschreibungen mit sich bringt, erscheint sie doch in manchen Fällen wie etwa der Entscheidungsunterstützung bei Diagnosesystemen oder beim Datenbankschemaentwurf ausreichend. Ziel ist es also, eine Eingabe in kontrollierter englischer Sprache zu ermöglichen. In Abschnitt 6.3.1 wird diese Möglichkeit erörtert.
2. Zum anderen ist die Linguistik selbst ein Anwendungsgebiet für die Modellgenerierung im allgemeinen und die EP-Tableaux-Methode im speziellen [Kon98]. Bei der Interpretation natürlichsprachlicher Diskurse, d.h. von Sätzen sowie ganzen Texten, spielen Modelle eine wichtige Rolle. Natürlichsprachliche Diskurse können mittels logischer Formalismen repräsentiert werden. Die Anwendung eines Modellgenerators auf diese logischen Repräsentationen gibt Hinweise auf inkonsistente Diskurse, überflüssige Information in einem Diskurs sowie implizit angenommenes Wissen [Kon98]. In Abschnitt 6.3.2 wird die Anwendung der EP-Tableaux-Methode zur Interpretation natürlichsprachlicher Diskurse erläutert.

6.3.1 Die EP-Tableaux-Methode und die kontrollierte Sprache ACE

Die Sprache ACE (Attempto Controlled English) [FS96] ist eine Untermenge der natürlichen Sprache, die einerseits so formal ist, daß sie von einem Computer effizient bearbeitet werden kann, andererseits flexibel genug, um sie auf eine natürliche Art verwenden zu können. Es wird mit einem vorgegebenen Vokabular gearbeitet, das aber für jede Anwendung entsprechend erweitert werden kann. Ein Text in der Sprache ACE besteht aus einfachen deklarativen Sätzen der Form Subjekt-Verb-Objekt, aus zusammengesetzten Sätzen sowie aus Fragesätzen. Zusammengesetzte Sätze entstehen aus einfachen Sätzen mittels *and, or, either, if-then, who, which, that, not.* Verben dürfen nur aktiv im „simple present tense“ in der dritten Person Singular oder Plural verwendet werden. Die Sätze können Relativsätze,

anaphorische Referenzen (wie z.B. Personalpronomen), Ellipsen[3], die Negation eines Satzes oder eines Verbs (z.B. *does not*) sowie Abkürzungen und Synonyme enthalten.

Wie wir in den letzten beiden Abschnitten gesehen haben, kann die EP-Tableaux-Methode in der Diagnose und beim Datenbankschemaentwurf angewendet werden. Eine Formelmenge, auf die die EP-Tableaux-Methode angewendet wird, muß in der PRQ-Syntax, wie sie in Abschnitt 4.3 festgelegt wurde, formuliert werden. Für Personen, die wenig Übung im Umgang mit formaler Logik haben, ist dies jedoch verhältnismäßig unbequem. Deshalb wäre es äußerst wünschenswert, eine Formelmenge, auf die die EP-Tableaux-Methode angewendet werden soll, in ACE formulieren zu können. Gleichzeitig scheint die Sprache ACE, trotz ihrer Einschränkungen, für diese Anwendungen ausreichend zu sein. Also wird ein Hilfsmittel benötigt, welches eine Folge von Sätzen (im Sinne der natürlichen Sprache), die in ACE formuliert ist, in eine Menge von Formeln übersetzt, auf die die EP-Tableaux-Methode angewendet werden kann. Um dies zu erreichen, gibt es verschiedene Möglichkeiten.

Ein erster naheliegender Ansatz ist sicherlich, eine Übersetzung von Sätzen in ACE in PRQ-Formeln anzugeben. Eine andere Möglichkeit wird später in Kapitel 8 angedeutet. Einen wesentlichen Teil der Übersetzung von Sätzen in ACE in PRQ-Formeln kann das System Attempto [FS96, SF96] übernehmen.

Das System Attempto übersetzt eine Folge von Sätzen in ACE in eine Diskursrepräsentationsstruktur (kurz: DRS) [KR93] und diese wiederum, falls gewünscht, in Prologklauseln. Eine Diskursrepräsentationsstruktur ist eine semantische Repräsentation von einer Folge natürlichsprachlicher Sätze und kann als eine Formel der Prädikatenlogik der ersten Stufe „in einem anderen Gewand“ aufgefaßt werden. In [KR93] wird eine Übersetzung von Diskursrepräsentationsstrukturen in Formeln der Prädikatenlogik der ersten Stufe angegeben. Erste Untersuchungen haben ergeben, daß eine Anbindung von Attempto an FINFIMO oder das System SIC (siehe Kapiel 7.3) durchaus realisierbar erscheint. Damit stünde eine Art „natürlichsprachliche“ Schnittstelle für FINFIMO sowie SIC zur Verfügung. Dabei wird eine Eingabe in ACE formuliert und von Attempto in eine DRS übersetzt. Die so entstandene DRS muß dann natürlich noch in die PRQ-Syntax übersetzt werden, also in die Syntax, auf die die EP-Tableaux-Methode anwendbar ist. Da eine DRS als prädikatenlogische Formel der ersten Stufe aufgefaßt werden kann, ist diese Übersetzung auf jeden Fall möglich. Zumindest für einfache Beispiele erscheint sie auch unproblematisch.

Beispiel 6.7 Wir betrachten den einfachen Satz „Every student visits Rome or Paris.“. In ACE kann dieser Satz genauso formuliert werden.

ACE: Every student visits Rome or Paris.

[3]In *If the code is valid then... If not* [*the code is valid*] *then...* ist [*the code is valid*] eine Ellipse und kann weggelassen werden.

Das System Attempto erzeugt daraus folgende DRS:

DRS: $[B, D]$
$\text{named}(B, \text{Rome})$
$\text{named}(D, \text{Paris})$
IF:
$[A]$
$\text{student}(A)$
THEN:
OR:
$[C]$
$\text{event}(C, \text{visit}(A, B))$
OR:
$[E]$
$\text{event}(E, \text{visit}(A, D))$

Hierbei sind B und D die sogenannten Diskursreferenten. Die Teilformeln $\text{named}(B, \text{Rome})$ und $\text{named}(D, \text{Paris})$ können als Interpretation von B bzw. D durch die Konstanten Rome bzw. Paris aufgefaßt werden. Der untere Teil der DRS entspricht einer universell quantifizierten Formel in Implikationsform, deren rechte Seite aus einer Disjunktion von existentiell quantifizierten Formeln besteht. Werden $\text{event}(C, \text{visit}(A, B))$ und $\text{event}(E, \text{visit}(A, D))$ als prädikatenlogische Formeln aufgefaßt, so ist der Formelteil 'visit' ein Funktionssymbol. Da für die EP-Tableaux-Methode aus bekannten Gründen keine Funktionssymbole zugelassen werden, wird 'visit' in eine Relation übersetzt. Die gesamte DRS kann in der PRQ-Syntax folgendermaßen repräsentiert werden:

$$\begin{aligned}\text{PRQ:}\quad & \forall A\ (\text{student}(A) \rightarrow \\ & \quad \exists C\ (\text{event}(C) \wedge \text{visit}(C, A, \text{Rome})) \vee \\ & \quad \exists E\ (\text{event}(E) \wedge \text{visit}(E, A, \text{Paris})))\end{aligned}$$

Um eine vollständige Übersetzung von Beschreibungen in ACE in PRQ-Formeln zu liefern, muß eine Transformation von den von Attempto erzeugten DRS in PRQ-Formeln angegeben werden. Diese Untersuchungen gehen jedoch über den Rahmen dieser Arbeit hinaus. Aufgrund ihrer Relevanz für die Anwendungen der EP-Tableaux-Methode einerseits, und aufgrund der anhand von Beispielübersetzungen genährten Überzeugung, daß die von Attempto erzeugten DRSen in natürlicher Weise PRQ-Formeln entsprechen, andererseits, sollten diese Untersuchungen jedoch zügig vorangetrieben werden.

6.3.2 Die EP-Tableaux-Methode in der Linguistik

Natürlichsprachliche Diskurse, also natürlichsprachliche Sätze oder ganze Texte, können durch Formeln der Prädikatenlogik der ersten Stufe repräsentiert werden [KR93, Kon98]. Die Anwendung eines Modellgenerierungsverfahrens auf die logischen Repräsentationen von natürlichsprachlichen Diskursen gibt dann Hinweise auf inkonsistente Diskurse, überflüssige

Information in einem Diskurs sowie implizit angenommenes Wissen [Kon98], da die Modelle einer solchen logischen Repräsentation die Bedeutung des natürlichsprachlichen Diskurses darstellen.

Beispiel 6.8 Der folgende Diskurs

Anna trinkt keinen Alkohol. Sie trinkt Whiskey. Whiskey ist Alkohol.

ist inkonsistent. Eine mögliche Formalisierung ist

$$\forall x(trinkt(anna, x) \rightarrow (alkohol(x) \rightarrow \bot)),$$
$$trinkt(anna, whiskey),$$
$$alkohol(whiskey).$$

Ein besonders schönes Beispiel für einen inkonsistenten Diskurs ist das folgende bekannte Gedicht.[4]

Beispiel 6.9

Dunkel war's, der Mond schien helle,
schneebedeckt die grüne Flur,
als ein Wagen blitzeschnelle
langsam um die Ecke fuhr.

Drinnen saßen stehend Leute,
schweigend ins Gespräch vertieft,
...

Die Anwendung eines widerlegungsvollständigen Modellgenerierungsverfahrens auf die Formalisierung in Beispiel 6.8 oder auf eine Formelisierung von Beispiel 6.9 zeigt die Inkonsistenz der Diskurse. Um dies zu erkennen, würde auch die Anwendung eines Refutationsbeweisers genügen. Wird ein Refutationsbeweiser dagegen auf einen konsistenten Diskurs angewendet, so würde er möglicherweise nicht halten. Da jedoch konsistente Diskurse in der natürlichen Sprache weit häufiger auftreten als inkonsistente Diskurse, ist die Anwendung von Refutationsbeweisern hier nicht sinnvoll. Statt dessen bietet sich die Anwendung der EP-Tableaux-Methode an. Die EP-Tableaux-Methode ist widerlegungsvollständig, d.h. inkonsistente Diskurse werden als solche nach endlicher Zeit erkannt. Weiter ist die EP-Tableaux-Methode vollständig für die Erfüllbarkeit im Endlichen. Für endlich erfüllbare Diskurse werden also endliche Modelle nach endlicher Zeit gefunden. Die Einschränkung auf endlich erfüllbare Diskurse ist nicht einschneidend, da ein konsistenter endlicher Diskurs im allgemeinen endlich erfüllbar ist[5]. Die gefundenen Modelle der logischen Repräsentation eines Diskurses können dann dazu dienen, überflüssige Information in einem Diskurs zu erkennen.

[4] Unbekannter Verfasser. Aus Ludwig Reiners: 'Der ewige Brunnen', Verlag C. H. Beck München

[5] Eine Ausnahme bilden Diskurse wie 'Es gibt unendlich viele Primzahlen.'

Beispiel 6.10 In dem Diskurs *Anna trinkt Whiskey. Whiskey ist Alkohol. Anna trinkt Alkohol.* bringt der letzte Satz keinerlei neue Information.

Wir gehen davon aus, daß die Modelle einer logischen Repräsentation eines Diskurses die Bedeutung des Diskurses darstellen. Der dritte Satz in Beispiel 6.10 enthält nur überflüssige Information. Auf die Modelle übertragen kann dies folgendermaßen formuliert werden: Jedes Modell einer Formalisierung der ersten beiden Sätze ist auch ein Modell einer Formalisierung des dritten Satzes, d.h. der dritte Satz folgt aus den ersten beiden Sätzen. Also entspricht die Frage nach überflüssiger Information in einem Satz bezüglich eines Diskurses der Frage, ob die logische Formalisierung des Satzes, etwa F, aus der logischen Formalisierung des Diskurses, etwa M folgt. Diese Frage ist entscheidbar, falls M ausschließlich endliche Modelle besitzt. Der Grund dafür liegt in folgendem Lemma (siehe etwa [EFT92]):

Lemma 6.11 *Für eine geschlossene Formel F und eine Menge M von geschlossenen Formeln gilt:*

$$M \models F \text{ genau dann, wenn } M \cup \{\neg F\} \text{ nicht erfüllbar ist.}$$

Besitzt die Menge M nur endliche Modelle, so reduziert sich die Frage, ob $M \models F$ darauf, entweder die Unerfüllbarkeit oder die endliche Erfüllbarkeit von $M \cup \{\neg F\}$ zu erkennen. Beides kann z.B. durch eine Anwendung der EP-Tableaux-Methode in endlicher Zeit erkannt werden. Damit ist also die Frage, ob ein Satz bezüglich eines gegebenen Diskurses zusätzliche Information enthält, entscheidbar, vorausgesetzt der gegebene Diskurs hat nur endliche Modelle. Die EP-Tableaux-Methode ist ein geeignetes Hilfsmittel, um diese Frage zu beantworten.

Schließlich können Modelle einer logischen Repräsentation eines Diskurses auch Hinweise auf implizit angenommenes Wissen oder allgemein bekanntes, sogenanntes Weltwissen, geben.

Beispiel 6.12 Betrachten wir folgenden Diskurs:

Anna trinkt Whiskey. Anna trinkt ein nichtalkoholisches Getränk.

Wird die EP-Tableaux-Methode auf die Formalisierung

$$\begin{array}{l} \mathit{trinkt}(\mathit{anna}, \mathit{whiskey}) \\ \exists x(\mathit{trinkt}(\mathit{anna}, x) \wedge (\mathit{alkohol}(x) \rightarrow \bot)) \end{array}$$

angewendet, so wird ein einziges Modell erzeugt. In diesem Modell gilt $\mathit{trinkt}(\mathit{anna}, \mathit{whiskey})$. Das ist überraschend, da wir erwarten, daß Anna ein nichtalkoholisches Getränk trinkt und annehmen, daß Whiskey ein alkoholisches Getränk ist. Dies ist jedoch Weltwissen und wurde nicht explizit formalisiert. Nehmen wir zu obiger Formalisierung

$$\mathit{alkohol}(\mathit{whiskey})$$

hinzu und wenden die EP-Tableaux-Methode auf die erweiterte Formalisierung an, so werden zwei Modelle erzeugt. Im ersten Modell gilt $trinkt(anna, anna)$. Dies ist wieder ein Hinweis auf implizites Wissen. Wir gehen stillschweigend davon aus, daß *anna* eine Person und *whiskey* ein Getränk ist. Dies wurde jedoch nicht explizit formalisiert. Deshalb kann $trinkt(anna, anna)$ erzeugt werden. Im zweiten Modell gibt es ein weiteres Getränk, etwa $getränk$, und es gilt $trinkt(anna, getränk)$. Zusätzlich gilt auch $\neg alkohol(getränk)$. Diese Information ist jedoch (entsprechend der Definition eines Termmodells, siehe Definition 2.35) implizit, da $alkohol(getränk)$ im erzeugten Modell nicht gilt.

In Beispiel 6.12 wurde erläutert, wie Modelle einer Formalisierung eines Diskurses Hinweise auf implizites Wissen im Diskurs oder zusätzlich angenommenes Weltwissen geben können. Bei genauer Betrachtung des obigen Beispiels besteht das zusätzlich angenommene Wissen darin, daß die im Diskurs auftretenden Objekte – hier *anna* und *whiskey* – gewissen Klassen von Objekten zugeordnet werden, die möglicherweise disjunkt sind. So wird *anna* der Klasse der Personen und *whiskey* der Klasse der alkoholischen Getränke zugeordnet. Statt von Klassen kann hier auch von Sorten gesprochen werden. In dem Modell, in dem $trinkt(anna, anna)$ gilt, wird nicht berücksichtigt, daß die Sorten 'Personen' und 'Getränke' sinnvollerweise disjunkt sind. In obigem Beispiel wird die Generierung von $trinkt(anna, anna)$ als Hinweis auf dieses implizite Wissen, das nicht formalisiert wurde, angesehen. Sicherlich ist es in gewissen Fällen hilfreich, solche Hinweise zu erhalten. Gleichzeitig könnte jedoch durch Unterstützung von Sorten direkt vermieden werden, daß diese Art von implizitem Wissen zu mangelhaften Formalisierungen führt. Die Formalisierung dieses impliziten Wissens kann in der Logik über Sorten oder, wie in Programmiersprachen, über Typen realisiert werden. Deshalb scheint eine Erweiterung der EP-Tableaux-Methode auf Sorten oder Typen sinnvoll. Eine solche Erweiterung der EP-Tableaux-Methode wäre auch für Anwendungen in anderen Bereichen wie der Diagnose oder dem Datenbankschemaentwurf hilfreich (siehe Kapitel 8).

Kapitel 7

Überprüfung der Korrektheit von Integritätsbedingungen

In Abschnitt 6.2 wurde bereits kurz angedeutet, wie Modellgenerierungsverfahren unterstützend während des Datenbankschemaentwurfs eingesetzt werden können. Die EP-Tableaux-Methode bietet sich hier besonders an, da sie nicht nur vollständig für die Unerfüllbarkeit, sondern auch für die Erfüllbarkeit im Endlichen ist. Da realisierbare Datenbankzustände im allgemeinen endlich sind, ist die Vollständigkeit für die Erfüllbarkeit im Endlichen für diese Anwendung von besonderer Bedeutung. In Abschnitt 7.1 wird ausführlich darauf eingegangen, warum und wie die EP-Tableaux-Methode zur Überprüfung der Korrektheit von Datenbankintegritätsbedingungen angewendet werden kann. Diese Überlegungen werden in Abschnitt 7.2 anhand von Beispielen verdeutlicht. In Abschnitt 7.3 wird schließlich das System SIC (Satisfiable Integrity Constraints) vorgestellt. Hierbei handelt es sich um ein Werkzeug zur Überprüfung der Korrektheit von Datenbankintegritätsbedingungen, dessen Inferenzmaschine eine Implementierung der EP-Tableaux-Methode ist.

7.1 Eine Anwendung der EP-Tableaux-Methode

Wie bereits erwähnt, besteht eine Datenbank aus zwei Teilen, dem *Schema* und den eigentlichen Daten, dem *Zustand*. Das Schema beinhaltet die Definition der Struktur der Daten und die *Integritätsbedingungen*. Die Integritätsbedingungen sind Formeln der Prädikatenlogik der ersten Stufe, die von jedem zulässigen Zustand der Datenbank, d.h. von der durch den Datenbankzustand repräsentierten Terminterpretation, erfüllt werden müssen. Bei der Entwicklung einer Datenbank sind der Datenbankschemaentwurf und das Füllen der Datenbank mit Daten zwei verschiedene Phasen, die oft von verschiedenen Personen durchgeführt werden [AHV95, Dat95].

Hinsichtlich der Integritätsbedingungen stellen sich zwei grundsätzlich verschiedene Fragen, erstens die Frage nach der *Erfülltheit* und zweitens die Frage nach der *Erfüllbarkeit* der Integritätsbedingungen.

Hinter der Frage nach der Erfülltheit der Integritätsbedingungen verbirgt sich folgende Problematik: Die Integritätsbedingungen müssen von jedem zulässigen Zustand der Datenbank erfüllt werden. Insbesondere müssen sie nach einer Änderungstransaktion (etwa dem Einfügen oder Löschen von Daten) auch von dem neu entstandenen Datenbankzustand erfüllt werden. Die Frage ist also, ob eine gegebene Interpretation (repräsentiert durch einen Datenbankzustand) ein Modell der Integritätsbedingungen ist. Selbstverständlich kann dies erst nach dem Füllen der Datenbank mit Daten überprüft werden. Wie diese Frage, d.h. die Erfülltheit von Integritätsbedingungen, überprüft werden kann, wurde bereits eingehend untersucht [Nic82, LST84, BM86, KSS87, BDM88a, BMM91].

Die Frage nach der Erfüllbarkeit von Integritätsbedingungen muß unabhängig von einem gegebenen Datenbankzustand geklärt werden. Hier muß vielmehr überprüft werden, ob es für eine Menge von Integritätsbedingungen überhaupt einen Datenbankzustand gibt, der die Integritätsbedingungen erfüllt. Die Frage ist also, ob es für eine gegebene Menge von Integritätsbedingungen ein Modell gibt, d.h. ob die Integritätsbedingungen erfüllbar sind. Darauf wurde in [BM86, BM87, BDM88b, Man90, NML93, DTU96, MP96] hingewiesen. Die Frage nach der Erfüllbarkeit der Integritätsbedingungen kann bereits in der ersten Phase der Entwicklung einer Datenbank, dem Schemaentwurf, überprüft werden. Dies ist nicht nur möglich, sondern sogar notwendig, denn die Erfüllbarkeit der Integritätsbedingungen ist eine wesentliche Voraussetzung für das Füllen einer Datenbank mit Daten. Sind die entworfenen Integritätsbedingungen unerfüllbar, dann wird jede mögliche Änderung des Datenbankzustandes zurückgewiesen. Insbesondere können keine Daten in die Datenbank eingetragen werden. Für die Anwender gibt es aber keine Hinweise, daß der Grund dafür nicht in unerlaubten Änderungen des Datenbankzustandes sondern in fehlerhaft entworfenen Integritätsbedingungen liegt. Da zudem oft verschiedene Personen am Schemaentwurf und am Füllen der Datenbank beteiligt sind, kann sich die Suche nach dem Fehler als schwieriges Unterfangen herausstellen.

Allerdings reicht die bloße Information, daß Integritätsbedingungen erfüllbar sind, in manchen Fällen nicht aus, um eine Datenbank aufzubauen. Einmal gibt es Fehler im Datenbankschema, die dazu führen, daß die Integritätsbedingungen nur vom leeren, also dem trivialen, Modell erfüllt werden. Dies führt zu den gleichen Schwierigkeiten wie bei unerfüllbaren Integritätsbedingungen. Es können keine Daten in die Datenbank eingefügt werden. Werden die Integritätsbedingungen nur von solchen Modellen, in denen eine oder mehrere bestimmte Relationen immer leer sind, erfüllt, so kann ebenfalls von Fehlern im Datenbankschema ausgegegangen werden. Diese partiell leeren Modelle sind (ähnlich wie das leere Modell) als mögliche Datenbankzustände wenig aussagekräftig. Zudem können beim Einfügen von Daten in die Datenbank die gleichen Schwierigkeiten auftreten wie bei Integritätsbedingungen, die nur vom leeren Modell erfüllt werden. Andere Fehler im Datenbankschema erwirken, daß die Integritätsbedingungen zwar erfüllt werden können, aber nur von unendlichen Modellen, d.h. durch unendlich große Datenbankzustände. Offensichtlich kann kein tatsächlicher Datenbankzustand solche Integritätsbedingungen erfüllen. In diesem Fall stellen die Integritätsbedingungen ein sogenanntes *Unendlichkeitsaxiom* dar. So schließt die Erfüllbarkeitsfrage für Integritätsbedingungen auch die Frage nach „prakti-

kablen“ Modellen, also Datenbankzuständen, ein. „Praktikable“ Datenbankzustände sind im allgemeinen endliche, weder leere noch partiell leere Datenbankzustände.

Die Erfüllbarkeitsfrage für Integritätsbedingungen ist in keiner Weise eine unrealistische Fragestellung. Es gibt erstaunlich einfache und realistische Beispiele, in denen einer der erwähnten Fehler im Schemaentwurf auftritt (vgl. Abschnitt 7.2). Wenn mehrere Personen an verschiedenen Teilen der Integritätsbedingungen arbeiten, sind solche Entwurfsfehler nahezu unvermeidbar.

Aus diesem Grund sollten bereits beim Schemaentwurf Hilfsmittel zur Verfügung stehen, um Fehler der oben beschriebenen Art zu entdecken und zu vermeiden. Idealerweise sollten es diese Hilfsmittel ermöglichen, während der Schemaentwurfsphase zu überprüfen, ob die Integritätsbedingungen

1. unerfüllbar,

2. endlich erfüllbar durch nichtleere oder nicht partiell leere Modelle oder

3. nur durch unendliche Modelle (d.h. durch unrealisierbare unendlich große Datenbankzustände) erfüllbar

sind. Im Falle von unerfüllbaren Integritätsbedingungen sollten zusätzlich Mittel zur Verfügung stehen, die die Suche nach den Integritätsbedingungen unterstützen, die die Ursache für die Unerfüllbarkeit sind.

Der dritte Wunsch der obigen Liste muß sicherlich ein Wunsch bleiben. Integritätsbedingungen sind Formeln der Prädikatenlogik erster Stufe. Deshalb sind durch die Unentscheidbarkeitsergebnisse theoretische Grenzen gesetzt. Aufgrund der Unentscheidbarkeit der Erfüllbarkeit im Endlichen [Tra50] kann von einem System nicht erwartet werden, daß es auf die Frage, ob eine Menge von Integritätsbedingungen ein Unendlichkeitsaxiom darstellt, immer nach endlicher Zeit antwortet. Trotzdem steht mit der EP-Tableaux-Methode ein Verfahren zur Verfügung, das sehr gut für die Überprüfung der Korrektheit von Datenbankintegritätsbedingungen geeignet ist. Ist eine Menge von Integritätsbedingungen unerfüllbar, so wird dies nach endlicher Zeit erkannt, da die EP-Tableaux-Methode vollständig für die Unerfüllbarkeit ist. Ist eine Menge von Integritätsbedingungen endlich erfüllbar, so wird dies ebenfalls nach endlicher Zeit erkannt, da die EP-Tableaux-Methode vollständig für die Erfüllbarkeit im Endlichen ist. Zusätzlich kann durch systematische Hinzunahme weiterer Axiome zu den Datenbankintegritätsbedingungen getestet werden, ob die Integritätsbedingungen nur durch leere oder partiell leere Modelle erfüllt werden. Gegebenenfalls können dann die Integritätsbedingungen verbessert werden. Da die EP-Tableaux-Methode ein Modellgenerierungsverfahren ist, werden bei ihrer Anwendung Modelle, also mögliche Datenbankzustände, generiert. Dies geschieht durch die Expansion eines EP-Tableau. Ist eine Menge von Integritätsbedingungen nur unendlich erfüllbar, so kann dies mittels einer schrittweisen Expansion des zugehörigen EP-Tableau und durch Analyse seiner Struktur von den Benutzenden des Systems erkannt werden.

7.2 Beispiele von problematischen Integritätsbedingungen

In diesem Abschnitt zeigen wir anhand von Beispielen, daß Schemaentwurfsfehler, wie sie in Abschnitt 7.1 beschrieben wurden, bereits in einfachen Beispielen auftreten können. Das erste Beispiel zeigt eine Menge von Integritätsbedingungen, die nur von einem leeren Modell, also einer Terminterpretation, die von der leeren Menge von Grundatomen repräsentiert wird, erfüllt wird. Dies entspricht einem leeren Datenbankzustand.

Beispiel 7.1 Die folgende Menge von Integritätsbedingungen ist eine Variante eines Beispiels aus [Man90].

(1) Abteilungen und Angestellte sind verschieden voneinander:
$\forall x(\mathit{abteilung}(x) \wedge \mathit{angestellte}(x) \rightarrow \bot)$

(2) Die Leitung einer Abteilung gehört zu dieser Abteilung:
$\forall x\ \forall y(\mathit{leitet}(x, y) \rightarrow \mathit{mitglied}(x, y))$

(3) Eine Person, die zu einer Abteilung gehört, arbeitet für die Leitung dieser Abteilung:
$\forall x\ \forall y\ \forall z(\mathit{mitglied}(x, z) \wedge \mathit{leitet}(y, z) \rightarrow \mathit{arbeitet_für}(x, y))$

(4) Jede angestellte Person gehört zu einer Abteilung:
$\forall x(\mathit{angestellte}(x) \rightarrow \exists y\ (\mathit{abteilung}(y) \wedge \mathit{mitglied}(x, y)))$

(5) Jede Abteilung hat eine angestellte Person als Leitung:
$\forall x(\mathit{abteilung}(x) \rightarrow \exists y\ (\mathit{angestellte}(y) \wedge \mathit{leitet}(y, x)))$

(6) Niemand arbeitet für sich selbst:
$\forall x(\mathit{arbeitet_für}(x, x) \rightarrow \bot)$

Diese Menge von Integritätsbedingungen kann erfüllt werden und zwar vom leeren Datenbankzustand. Somit scheinen die Integritätsbedingungen gut entworfen worden zu sein. Der leere Datenbankzustand ist aber ein uninteressanter Sonderfall, da sich keine Daten in der Datenbank befinden. Gibt es dagegen eine Angestellte, etwa $\mathit{angestellte}(\mathit{anna})$, in der Datenbank, so ist die Menge der Integritätsbedingungen unerfüllbar.

Wegen $\mathit{angestellte}(\mathit{anna})$ verlangt Bedingung (4) die Existenz einer Abteilung y, zu der anna gehört. Wenn y gleich anna ist, erhalten wir $\mathit{abteilung}(\mathit{anna})$, was mit (1) zu einem Widerspruch führt.

Ist y ein anderes Element, etwa d_1, so gilt $\mathit{angestellte}(\mathit{anna})$ und $\mathit{abteilung}(d_1)$ und $\mathit{mitglied}(\mathit{anna}, d_1)$. Nun verlangt (5) die Existenz einer leitenden Person y von d_1. Falls y gleich anna ist, so ist die einzige neue Konsequenz $\mathit{leitet}(\mathit{anna}, d_1)$. Daraus folgt zusammen mit $\mathit{mitglied}(\mathit{anna}, d_1)$ und (3) $\mathit{arbeitet_für}(\mathit{anna}, \mathit{anna})$. Mit (6) führt dies zu einem Widerspruch. Falls y gleich d_1 ist, so entsteht wieder ein Widerspruch zu (1).

Also muß y ein weiteres Element sein, etwa a_2, so daß $\mathit{angestellte}(\mathit{anna})$ und $\mathit{abteilung}(d_1)$ und $\mathit{mitglied}(\mathit{anna}, d_1)$ und $\mathit{angestellte}(a_2)$ und $\mathit{leitet}(a_2, d_1)$ gelten. Aus letzterem folgt

mitglied(a_2, d_1) mit (2), dann *arbeitet_für*(a_2, a_2) mit (3), und schließlich ein Widerspruch mit (6).

Sobald es also eine angestellte Person gibt, ist es nicht mehr möglich, die Integritätsbedingungen zu erfüllen. Die Integritätsbedingungen sind so entworfen, daß jedes Einfügen von Daten zurückgewiesen wird. Das folgende Beispiel zeigt, wie ein Unendlichkeitsproblem entstehen kann. Die Integritätsbedingungen sind so entworfen, daß sie nur von einer unrealisierbaren unendlich großen Datenbank erfüllt werden können.

Beispiel 7.2

(1) Jede angestellte Person arbeitet für jemanden:
$\forall x \;\; \mathit{angestellte}(x) \rightarrow \exists y \; \mathit{angestellte}(y) \wedge \mathit{arbeitet_für}(x, y)$

(2) Keine angestellte Person arbeitet für sich selbst:
$\forall x \;\; \mathit{arbeitet_für}(x, x) \rightarrow \bot$

(3) Die Relation „arbeitet_für“ ist transitiv:
$\forall x \; \forall y \; \forall z \;\; \mathit{arbeitet_für}(x, y) \wedge \mathit{arbeitet_für}(y, z) \rightarrow \mathit{arbeitet_für}(x, z)$

Solange die Datenbank leer ist, entstehen keine Schwierigkeiten. Die Menge der Integritätsbedingungen wird vom leeren Datenbankzustand erfüllt. Wird allerdings eine Angestellte *angestellte*(*anna*) in die Datenbank eingefügt, so passiert folgendes.

Wegen *angestellte*(*anna*) verlangt (1) die Existenz einer angestellten Person y, für die *anna* arbeitet. Falls y gleich *anna* ist, erhalten wir *arbeitet_für*(*anna*, *anna*) und mit (2) einen Widerspruch. Somit muß y ein anderes Element a_1 sein, derart daß *angestellte*(*anna*) und *angestellte*(a_1) und *arbeitet_für*(*anna*, a_1) gelten.

Als nächstes verlangt (1) die Existenz eines y, für das a_1 arbeitet. Falls y gleich *anna* ist, erhalten wir *arbeitet_für*(a_1, *anna*), dann *arbeitet_für*(*anna*, *anna*) mit (3) und einen Widerspruch mit (2). Falls y gleich a_1 ist, erhalten wir *arbeitet_für*(a_1, a_1) und einen Widerspruch mit (2). Somit ist y ein weiteres Element a_2.

Wieder verlangt (1) die Existenz eines y, für das a_2 arbeitet. Wieder führen die Fälle, in denen y eines der vorherigen Elemente ist, zu Widersprüchen, und y muß gleich einem weiteren Element a_3 sein, und so weiter.

Die Integritätsbedingungen können erfüllt werden, indem unendlich viele Angestellte eingeführt werden, aber nicht mit einer endlichen Anzahl von Angestellten. Dieses Beispiel gehört zur Klasse der Unendlichkeitsaxiome, die von keiner Prozedur alle als solche erkannt werden können.

7.3 Die Implementierung: SIC

Der folgende Abschnitt ist dem System SIC gewidmet. Das System SIC (Akronym von *Satisfiable Integrity Constraints*) ist ein Prototyp eines interaktiven Deduktionssystems, das den Entwurf endlich erfüllbarer Integritätsbedingungen für Datenbanken unterstützt.

Es behandelt das Erfüllbarkeitsproblem für Integritätsbedingungen während der Schemaentwurfsphase einer Datenbank.

Das System SIC wurde ausgehend von dem System SNARKS [Ket97, KE97] entwickelt. SNARKS ist ein graphisches Werkzeug für die Analyse von Beweisen, wie sie von Tableauxverfahren und ähnlichen Deduktionsmethoden erzeugt werden. Die Inferenzmaschine von SNARKS ist eine an SATCHMO [MB88] angelehnte Implementierung der PUHR-Tableaux-Methode. Um das System SIC zu erhalten, wurde die Inferenzmaschine von SNARKS durch eine an FINFIMO angelehnte Implementierung der EP-Tableaux-Methode ersetzt. Die Implementierung der EP-Tableaux-Methode wurde so vorgenommen, daß die Möglichkeiten, die die graphische Oberfläche von SNARKS bietet, übernommen werden konnten.

In Kapitel 4 wurde die Vollständigkeit für die Unerfüllbarkeit sowie die Vollständigkeit für die Erfüllbarkeit im Endlichen der EP-Tableaux-Methode gezeigt. Diese Eigenschaften sind von zentraler Bedeutung für die Anwendung der EP-Tableaux-Methode zur Überprüfung der Korrektheit von Datenbankintegritätsbedingungen. Allerdings wurde in Abschnitt 4.6 darauf hingewiesen, daß die Vollständigkeit für die Erfüllbarkeit im Endlichen bei einer Implementierung der EP-Tableaux-Methode mit Tiefensuche verlorengeht. Aus diesem Grund ist die EP-Tableaux-Methode im System SIC mit einer kontrollierten Breitensuche, wie sie unten beschrieben wird, implementiert. SIC bietet aber zusätzlich die Möglichkeit auch andere Suchstrategien wie die Tiefensuche oder eine kontrollierte Tiefensuche zu verwenden.

Die EP-Tableaux-Methode ist in SIC in Prolog implementiert, die graphische Oberfläche, die von SNARKS übernommen wurde, ist in Java implementiert. SIC kann von geeigneten WWW-Browsern aufgerufen werden.

Wird SIC auf eine Menge von Integritätsbedingungen angewendet, so wird ein EP-Tableaux für die Integritätsbedingungen expandiert. Aufgrund der internen Expansion des EP-Tableau kann ein Ableitungsbaum auf dem Bildschirm erzeugt werden, der in seiner Struktur dem EP-Tableau entspricht, dessen Knoten aber nicht exakt mit den Knoten des EP-Tableau übereinstimmen. Für die Visualisierung des Ableitungsbaumes gibt es viele Optionen. Um die Möglichkeiten von SIC vorzustellen, betrachten wir noch einmal die Integritätsbedingungen aus Beispiel 7.1. Zur Erinnerung sei die Formalisierung hier wieder gegeben:

(1) $\forall x(\mathit{abteilung}(x) \wedge \mathit{angestellte}(x) \rightarrow \bot)$

(2) $\forall x\ \forall y(\mathit{leitet}(x, y) \rightarrow \mathit{mitglied}(x, y))$

(3) $\forall x\ \forall y\ \forall z(\mathit{mitglied}(x, z) \wedge \mathit{leitet}(y, z) \rightarrow \mathit{arbeitet_für}(x, y))$

(4) $\forall x(\mathit{angestellte}(x) \rightarrow \exists y\ (\mathit{abteilung}(y) \wedge \mathit{mitglied}(x, y)))$

(5) $\forall x(\mathit{abteilung}(x) \rightarrow \exists y\ (\mathit{angestellte}(y) \wedge \mathit{leitet}(y, x)))$

(6) $\forall x(\mathit{arbeitet_für}(x, x) \rightarrow \bot)$

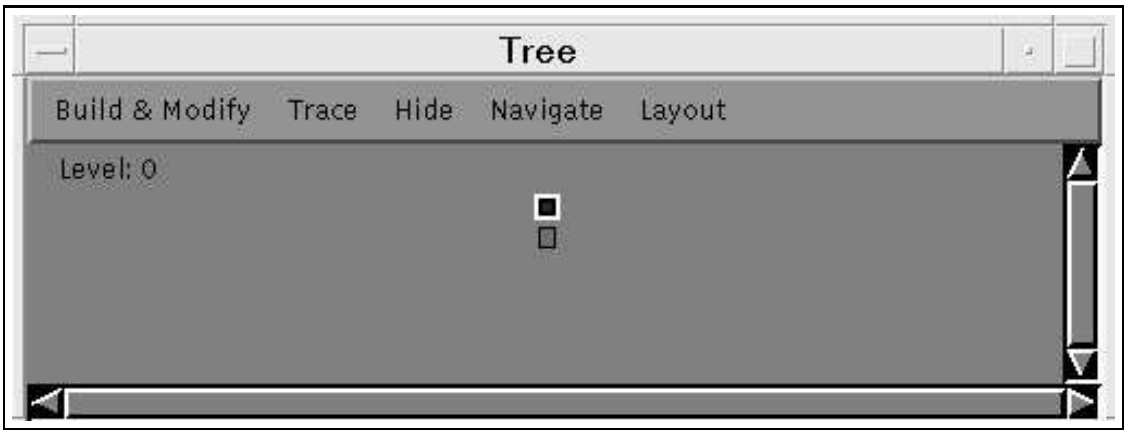


Abbildung 7.1: Ableitungsbaum für Beispiel 7.1 ausgegeben von SIC

Diese Integritätsbedingungen werden vom leeren Modell erfüllt. Was in Abbildung 7.3 zu sehen ist, wird von SIC erzeugt, wenn das System auf die Integritätsbedingungen in Beispiel 7.1 angewendet wird.

Das hell umrandete Viereck stellt die Datenbank dar. Sie ist in diesem Fall leer. Das Viereck darunter ist das „Modellsymbol“. Damit wird angezeigt, daß die leere Datenbank ein Modell der Integritätsbedingungen repräsentiert. Um nun zu testen, ob die Integritätsbedingungen auch durch nichtleere Modelle erfüllt werden können, ermöglicht SIC die manuelle Eingabe von Daten. In unserem Beispiel wurde manuell eine Angestellte namens Anna (`ang(anna)`) und zur Konstantenverwaltung zusätzlich `dom(anna)` in die Datenbank eingegeben. Abbildung 7.3 zeigt einen Teilbaum, nachdem die Daten in die Datenbank eingegeben wurden. Die Wurzel des abgebildeten Baumes zeigt jetzt die modifizierte Datenbank, die aus dem Atom `ang(anna)` und der Konstantenverwaltung dom(anna) besteht. Der Balken rechts der Wurzel zeigt an, daß die Datenbank manuell verändert wurde.

Um den Ableitungsbaum vollständig zu expandieren, stehen folgende Möglichkeiten zur Verfügung:

- schrittweise Expansion, d.h. Knoten für Knoten
- Expansion bis zu einer vorgegebenen Baumtiefe
- Expansion solange, bis ein vorgegebenes Atom erfüllt ist
- Expansion solange, bis der Baum sich verzweigt
- Expansion des ganzen Baumes in einem Schritt

Während der Expansion eines Baumes kann zwischen den verschiedenen Möglichkeiten auch gewechselt werden. Jeder Knoten repräsentiert eine Menge von Grundatomen, die zu diesem Zeitpunkt der Berechnung bereits in der Datenbank enthalten sind. Diese

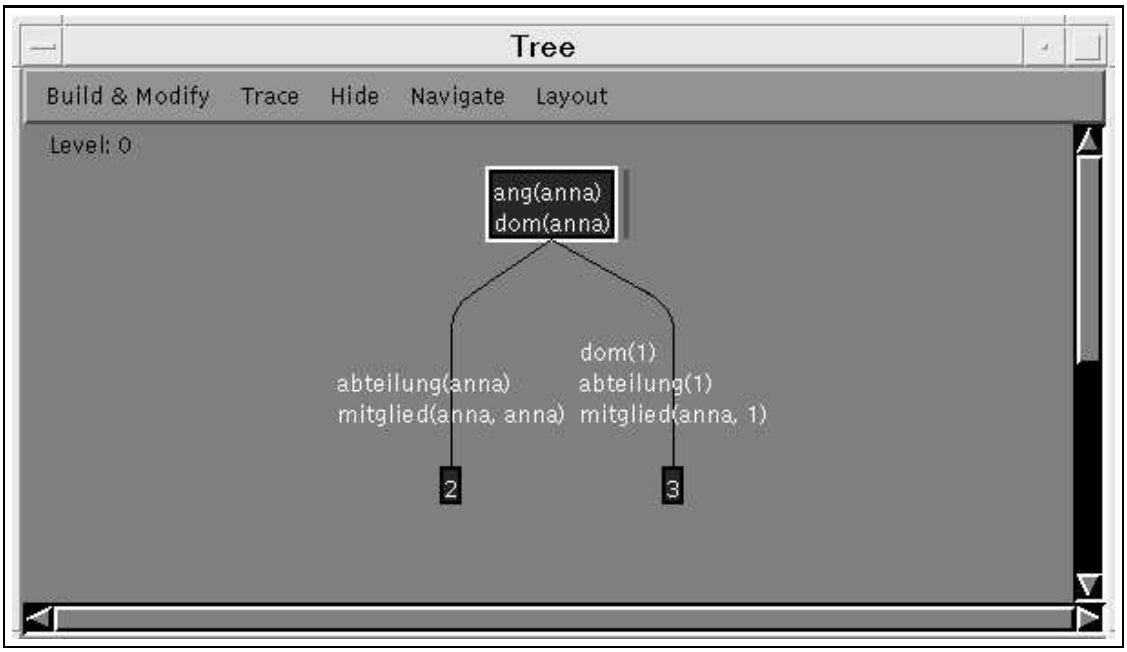


Abbildung 7.2: Ableitungsbaum für Beispiel 7.1 mit modifizierter Datenbank

Grundatome repräsentieren wiederum eine Terminterpretation. So kann für jeden Knoten die Menge der von dieser Terminterpretation unerfüllten Integritätsbedingungen angezeigt werden. Aus dieser Menge können Integritätsbedingungen ausgewählt werden, die im nächsten Berechnungsschritt dann erfüllt werden. Es ist also sogar möglich, einzelne Ableitungsschritte auszuwählen und so die Expansion des Baumes zu steuern. In Abbildung 7.3 wurde die schrittweise Expansion gewählt. In einem ersten Schritt wurden die Kinderknoten der Wurzel expandiert.

Die Knoten in Abbildung 7.3 sind verschieden beschriftet. Die Wurzel zeigt die Datenbank zu Beginn der Berechnung, die anderen Knoten sind nur numeriert. Allerdings kann die Beschriftung der Knoten genauso wie auch die Beschriftung der Kanten in vielfacher Weise verändert werden. Knoten und Kanten können mit

- Nummern
- allen in der Datenbank enthaltenen atomaren Formeln
- den im letzten Schritt in die Datenbank eingefügten atomaren Formeln
- den unerfüllten Integritätsbedingungen
- den zuletzt erfüllten Integritätsbedingungen
- einer Erklärung für das Einfügen eines Atoms

beschriftet werden. Bei den Knoten wird zwischen dem sogenannten aktiven Knoten, also dem Knoten, der im Moment „angeklickt“ wurde (in Beispiel 7.3 die Wurzel), und den anderen Knoten unterschieden. Der aktive Knoten wird durch eine helle Umrahmung gekennzeichnet. Er und die anderen Knoten können verschieden beschriftet werden (siehe Abbildung 7.3). Die Beschriftung der Knoten und Kanten kann jederzeit auch während der Expansion des Baumes verändert werden.

Aus Abschnitt 7.2 ist bereits bekannt, daß die Integritätsbedingungen aus Beispiel 7.1 zusammen mit dem Atom angestellte($anna$) unerfüllbar sind. Aufgrund der Vollständigkeit für die Unerfüllbarkeit der EP-Tableaux-Methode ist also jeder Baum für diese Menge von Formeln geschlossen. Abbildung 7.3 zeigt einen vollständig expandierten Ableitungsbaum für die Integritätsbedingungen aus Beispiel 7.1 zusammen mit dem Atom angestellte($anna$). Alle Zweige des Baumes sind geschlossen. Dies wird durch das Symbol $\perp$ unter den Blättern der Zweige angezeigt. Das Blatt des dritten Zweiges von links ist in diesem Ableitungsbaum der aktive Knoten. Er ist mit den atomaren Formeln, die dem Datenbankzustand entsprechen, beschriftet. Die Formel $\perp$ wird hier durch `false` repräsentiert.

In diesem Fall ist es nun natürlich von Interesse zu erkennen, warum die entworfenen Integritätsbedingungen unerfüllbar sind. Zu diesem Zweck kann eine Abhängigkeitsanalyse angefordert werden, wie in Abbildung 7.3 zu sehen ist. Hier wurde im mittleren Zweig die Formel `false`, welches den Widerspruch im Zweig darstellt, duch ein helles Quadrat markiert. SIC hat nach unten zeigende Dreiecke der gleichen Farbe hinzugefügt, um zu zeigen, wie es zu dem Widerspruch in dem Zweig kam: Die neu eingefügte Konstante 1 (d_1 in Abschnitt 7.2) ist sowohl eine Abteilung als auch eine angestellte Person. Dies führt mit Integritätsbedingung (1) zum Widerspruch. Genauso kann der rechte Zweig analysiert werden. Hier wurde die Formel `false` durch ein dunkleres Rechteck markiert. Da SIC daraufhin die Atome `leitet(2,1)`, `mitglied(2,1)` und `a_f(2,2)`[1] markiert hat, kann der Grund für den Widerspruch ermittelt werden. Aufgrund von Bedingung (3) wurde `a_f(2,2)` in die Datenbank eingefugt, was mit Bedingung (6) zu einem Widerspruch führt. Stimmen also in Bedingung (3) die Variablen x und y überein, wird ein Widerspruch erzeugt. Durch eine Abänderung der Integritätsbedingung (3) zu $\forall x\ \forall y\ \forall z\ \mathit{mitglied}(x,z) \wedge \mathit{leitet}(y,z) \rightarrow x{=}y \vee \mathit{arbeitet_fuer}(x,y)$ kann das vermieden werden. Genauso wie für die Formel `false` kann für jede andere Formel der Datenbank eine Abhängigkeitsanalyse angefordert werden.

Abbildung 7.3 zeigt einen Ableitungsbaum für die verbesserten Integritätsbedingungen. Auch in diesem Fall wurden die Atome `ang(anna)` und `dom(anna)` manuell in die Datenbank eingeführt. Jetzt sind die Integritätsbedingungen durch ein nichtleeres Modell erfüllbar. Zwei Zweige des Baumes repräsentieren Modelle, d.h. mögliche Datenbankzustände, die die Integritätsbedingungen erfüllen. Sie sind durch ein helles Rechteck unter ihrem Blatt gekennzeichnet.

Gleichzeitig zeigt Abbildung 7.3 eine weitere Eigenschaft von SIC. Es ist möglich, gewisse Knoten des Baumes bzw. ganze Teilbäume zu „verstecken“, um so den Baum in einer kompakteren Form darzustellen. Mit „verstecken“ ist hier das Zusammenschieben eines

[1] `a_f(2,2)` steht für arbeitet_für$(2,2)$.

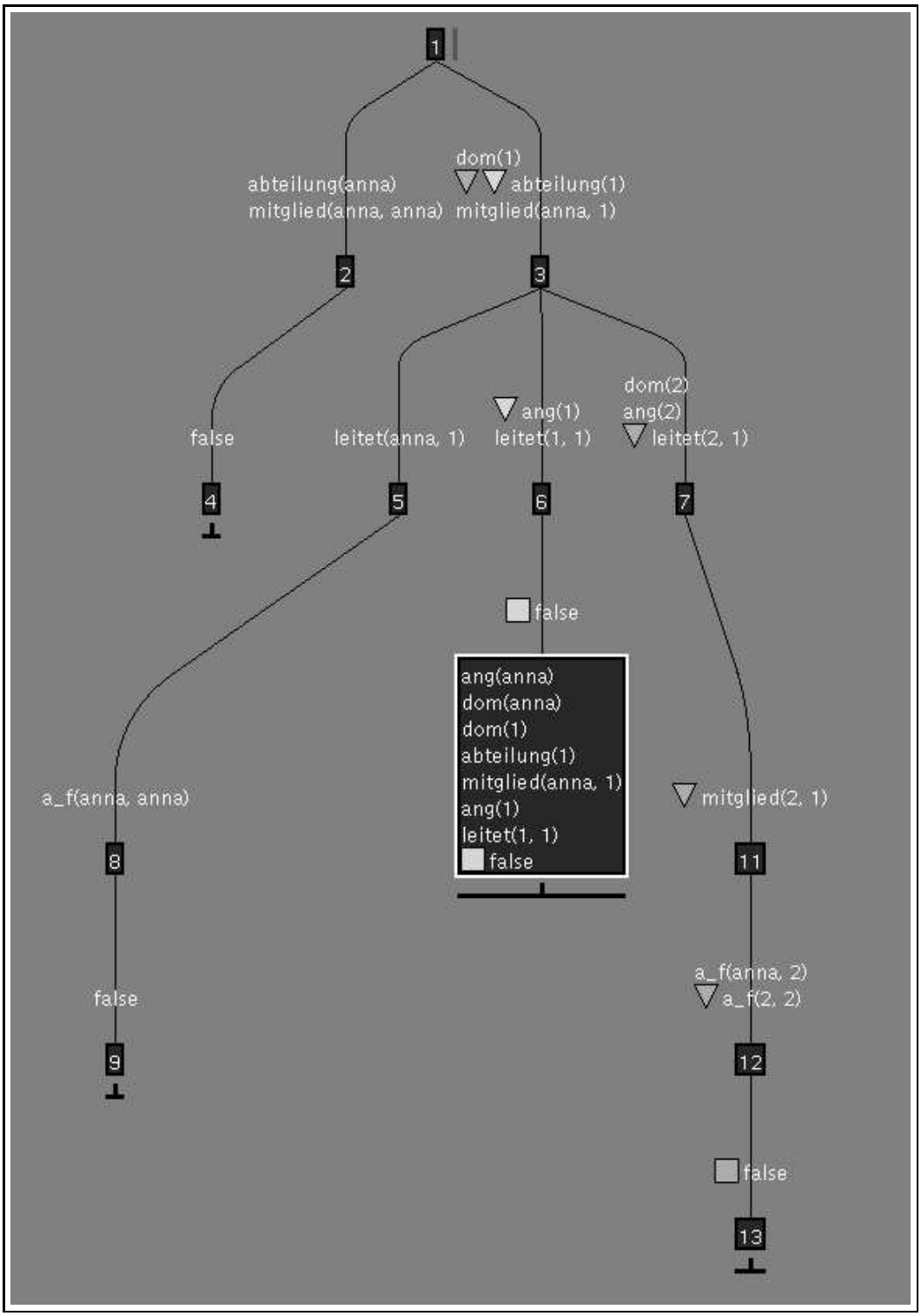


Abbildung 7.3: Ableitungsbaum für Beispiel 7.1 ausgegeben von SIC

Knotens mit seinem Elternknoten gemeint. Um Teilbäume zu verstecken werden alle Knoten des zu versteckenden Teilbaumes in seine Wurzel geschoben. Gekennzeichnet werden versteckte Knoten dadurch, daß der Elternknoten bzw. die Wurzel des Teilbaums aus einem Dreieck besteht. Um Knoten zu verstecken, können verschiedene Bedingungen angegeben werden. So können etwa alle Knoten bis auf die Blätter versteckt werden. In Abbildung 7.3 wurden alle Knoten, die zu einem Widerspruch führen, d.h. die Formel `false` enthalten, versteckt. So bleiben nur noch die Zweige übrig, die ein Modell repräsentieren.

Besonders bei größeren Mengen von Integritätsbedingungen ist dieses Hilfsmittel von erheblicher Bedeutung. Dadurch können unübersichtliche Bäume auf die wesentlichen Teile reduziert werden, um so eine Analyse zu erleichtern. Sind die Integritätsbedingungen nur unendlich erfüllbar, so können durch geschicktes Verstecken von Teilbäumen wiederkehrende Strukturen bei der Expansion des Ableitungsbaumes erkannt werden. Zusammen mit einer Abhängigkeitsanalyse können dann möglicherweise Unendlichkeitsaxiome erkannt werden.

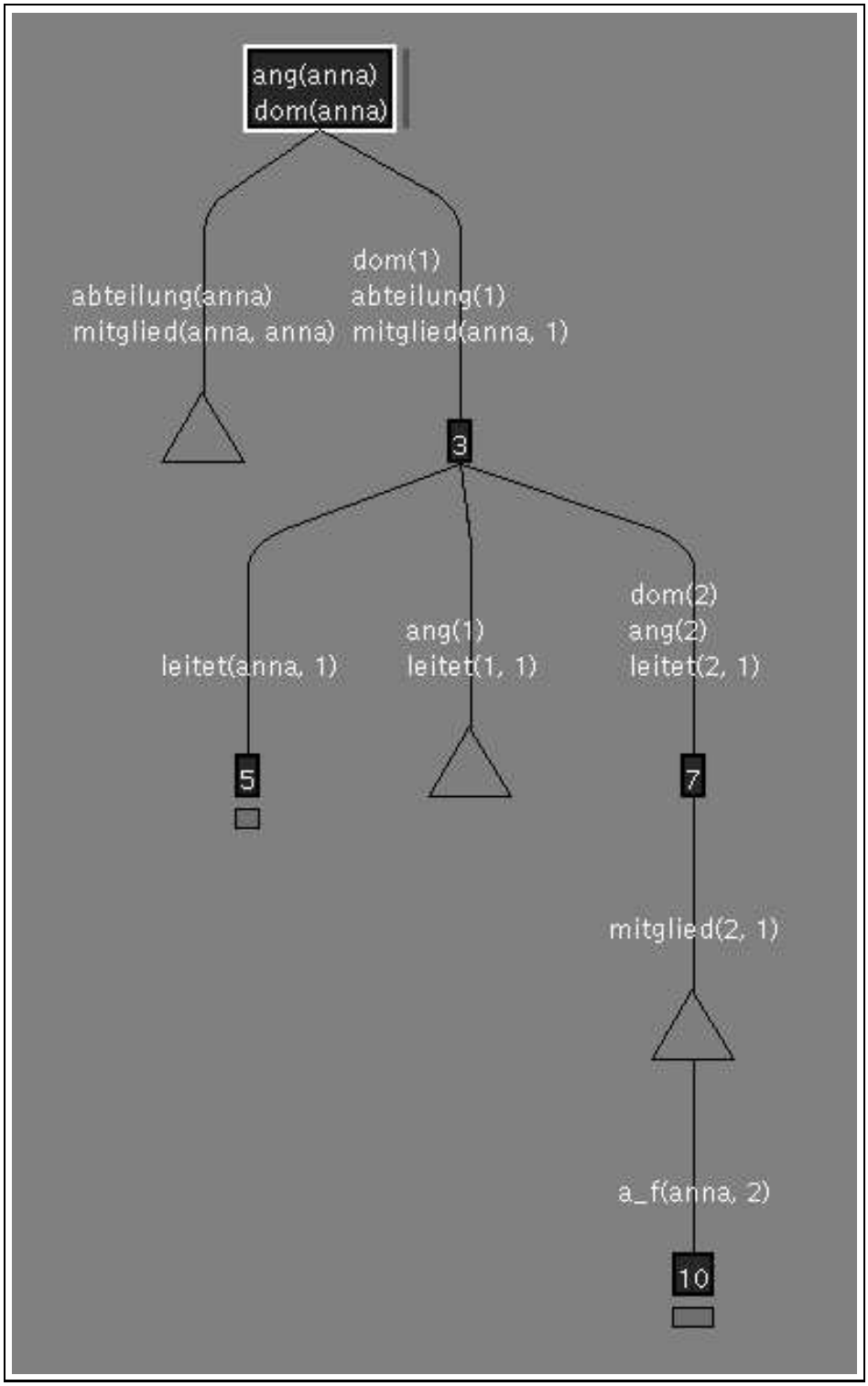


Abbildung 7.4: Ableitungsbaum für die verbesserten Integritätsbedingungen aus Beispiel 7.1 ausgegeben von SIC

Kapitel 8

Abschließende Bemerkungen

8.1 Zusammenfassung

Mit dieser Arbeit wurde die Extended-Positive-Tableaux-Methode (kurz EP-Tableaux-Methode) eingeführt. Die EP-Tableaux-Methode basiert auf der Methode der Positive-Unit-Hyper-Resolution-Tableaux (kurz PUHR-Tableaux) [BY96] und erweitert diese.

Die PRQ-Syntax, die von der EP-Tableaux-Methode gefordert wird, erweitert das Prinzip der Bereichsbeschränkung auf Formeln mit universellen und existentiellen Quantoren. Sie ist ein sehr weitreichendes Fragment einer Sprache der ersten Stufe ohne Funktionssymbole. Insbesondere läßt die PRQ-Syntax explizite Existenzquantoren zu. Dies ist eine deutliche Erweiterung gegenüber der PUHR-Tableaux-Methode, wo die Klauselform gefordert wird. In dieser Arbeit konnte gezeigt werden (siehe Satz 4.24), daß die durch die PRQ-Syntax definierte Sprache gleichmächtig ist wie eine Sprache der ersten Stufe ohne Funktionssymbole.

Für die EP-Tableaux-Methode konnte die Korrektheit und Vollständigkeit für die Unerfüllbarkeit, die Modellkorrektheit sowie die Vollständigkeit für die Erfüllbarkeit im Endlichen gezeigt werden. Besonders die Vollständigkeit für die Erfüllbarkeit im Endlichen (Satz 4.48, Seite 80) ist im Hinblick auf Anwendungen ein wichtiges Ergebnis.

Die EP-Tableaux-Methode wurde in Prolog implementiert (siehe Kapitel 5). Die Implementierung namens FINFIMO wurde vorgestellt und erläutert. In diesem Zusammenhang wurde auch eine Variante der PRQ-Syntax diskutiert (siehe Abschnitt 5.2).

Die Eigenschaft, gleichzeitig vollständig für die Unerfüllbarkeit und für die Erfüllbarkeit im Endlichen zu sein, macht die EP-Tableaux-Methode für verschiedene Anwendungen besonders interessant. In Kapitel 6 wurde erläutert, wie die EP-Tableaux-Methode in der Diagnose, beim Datenbankschemaentwurf und in der Linguistik angewendet werden kann.

Beim Datenbankschemaentwurf kann die EP-Tableaux-Methode zur Überprüfung der Korrektheit von Datenbankintegritätsbedingungen eingesetzt werden (siehe Kapitel 7). Zu diesem Zweck wurde das System SIC entwickelt, dessen Inferenzmaschine aus einer Implementierung der EP-Tableaux-Methode besteht. Das System SIC wurde ausführlich

beschrieben. Zudem wurde die Möglichkeit einer natürlichsprachlichen Schnittstelle für FINFIMO und SIC diskutiert.

8.2 Verwandte Arbeiten

Gerade weil für viele Anwendungen besonders endliche Modelle von Bedeutung sind, gibt es bereits die verschiedensten Ansätze, um endliche Modelle zu erzeugen. Dabei sind folgende Klassen von Methoden oder Verfahren zu unterscheiden:

- Modellgenerierungsverfahren für eine gegebene (oder bis zu einer gegebenen) Kardinalität der gesuchten Modelle
- Kombinierte Verfahren, die aus einem Modellgenerierungsverfahren für eine gegebene (oder bis zu einer gegebenen) Kardinalität und einem Refutationsverfahren bestehen
- Tableaux-Verfahren, die die δ^*- oder $\exists$-Regel integrieren
- Tableaux-Verfahren zur Generierung minimaler Modelle
- Tableaux-Verfahren, die die klassische γ-Regel verwenden anstelle der integrierten Unifikation wie in der PUHR- oder EP-Tableaux-Methode
- Modellgenerierungsverfahren für endlich repräsentierbare Modelle

Vermutlich eines der ersten Modellgenerierungsverfahren für eine vorgegebene Kardinalität wurde in [Hör81] beschrieben. Inzwischen zählen FINDER [Sla92] und SEM [ZZ95] sicherlich zu den bekanntesten Modellgeneratoren dieser Art. Ihre Stärke liegt in einer ausgeklügelten und sehr effizienten Implementierung einer „erschöpfenden“ Suche nach Modellen der vorgegebenen Kardinalität für eine Formelmenge. Mit den meisten dieser Modellgenerierungsverfahren kann die Suche mit einer höheren Kardinalität fortgesetzt werden, falls kein Modell mit der zuerst vorgegebenen Kardinalität gefunden werden kann. Die Angabe einer oberen Schranke ist aber für manche Anwendungen wie etwa der Überprüfung der Korrektheit von Integritätsbedingungen möglicherweise eine zu starke Forderung. Zudem können Modellgenerierungsverfahren wie FINDER und SEM im Gegensatz zur EP-Tableaux-Methode die Unerfüllbarkeit einer Formelmenge nicht erkennen.

Um diese Eigenschaft ebenfalls zur Verfügung zu haben, können Systeme wie FINDER oder SEM mit Refutationsbeweisern gekoppelt werden, wie es z.B. in [PP98] beschrieben wird. Ein solches kombiniertes System ist dann sowohl vollständig für die Unerfüllbarkeit als auch für die Erfüllbarkeit im Endlichen. Trotzdem scheint ein Verfahren wie die EP-Tableaux-Methode für viele Anwendungen geeigneter zu sein, da ein einziges Deduktionssystem besser zu handhaben ist als eine Kopplung zweier Systeme. Dies gilt besonders für interaktive Systeme.

Die δ^*- oder $\exists$-Regel, die in der EP-Tableaux-Methode verwendet wird, wurde bereits in früheren Arbeiten vorgeschlagen [BM87, Hin88, Lor94]. In [BM87] wurde sie jedoch nicht direkt in einem Tableaux-Kontext beschrieben, sondern im Rahmen eines Verfahrens, mittels dessen (ähnlich wie im Satz von Herbrand) das Herbranduniversum aufgezählt wird und so Modelle der betrachteten Formeln gesucht werden. Dagegen wird in [Hin88, Lor94] ein tableaux-basiertes Beweisverfahren für eine nichtmonotone Folgerungsrelation vorgestellt, die eng mit McCarthys „domain circumscription" [McC80] zusammenhängt. Bei diesem Beweisverfahren werden, genau wie bei der EP-Tableaux-Methode, existentiell quantifizierte Formeln mittels der $\exists$-Regel behandelt, um möglichst kleine Modelle zu generieren. Universell quantifizierte Formeln werden aber, im Gegensatz zur EP-Tableaux-Methode, mittels der klassischen γ-Regel behandelt. Ein Vergleich des Verfahrens von [Hin88, Lor94] mit der EP-Tableaux-Methode ist allerdings schwierig und könnte wenig aussagekräftig sein, da das Verfahren von [Hin88, Lor94] für nichtmonotone Logiken entworfen wurde.

Die Behandlung der existentiell quantifizierten Formeln mittels der δ^*- oder $\exists$-Regel in der EP-Tableaux-Methode führt zu möglichst kleinen Modellen bezüglich der Domäne des erzeugten Modells. Genauso ist es aber auch möglich, Modelle hinsichtlich der Menge der erfüllten Grundatome zu minimieren. Dieser Minimalitätsbegriff wurde ebenfalls in dieser Arbeit verwendet und die Vollständigkeit der Erfüllbarkeit im Endlichen der EP-Tableaux-Methode bezieht sich auf diesen Minimalitätsbegriff. Allerdings werden durch Anwendung der EP-Tableaux-Methode auch nichtminimale Modelle bezüglich dieses Minimalitätsbegriffs erzeugt. Eine Anwendung der PUHR-Tableaux-Methode zusammen mit der MM-Prozedur [BY96], wie sie in Abschnitt 3.4 beschrieben wurde, führt zu ausschließlich minimalen Modellen, falls die gegebene Klauselmenge erfüllbar ist. Zusätzlich werden bei dieser Methode auch keine Duplikate erzeugt. Die Tableaux-Methode von Niemelä [Nie96b] für die Aussagenlogik erzwingt ebenfalls die Generierung ausschließlich minimaler Modelle. Im Gegensatz zu [BY96] werden die erzeugten Modelle in [Nie96b] nicht miteinander verglichen. Deshalb ist die Erzeugung von Duplikaten möglich. In [Nie96a] wurde die Methode von [Nie96b] auf die Logik der ersten Stufe ausgedehnt. Da es für manche Anwendungen sicherlich sinnvoll ist, die erzeugten Modelle sowohl bezüglich der Domäne als auch bezüglich der Menge der erfüllten Grundatome zu minimieren, sollte die Möglichkeit einer Kombination der EP-Tableaux-Methode mit der Methode von [Nie96a] untersucht werden. In [Oli92] wird eine Tableaux-Methode für die Aussagenlogik vorgestellt, mittels der durch einen Vergleich der erzeugten Atome in einem Zweig mit vorher erzeugten Modellen, die generierten Modelle minimiert werden. Allerdings scheint diese Methode vor allem von theoretischem Interesse und bezüglich Anwendungen eher problematisch zu sein.

Der Unterschied zwischen der EP-Tableaux-Methode und den meisten anderen Tableaux-Verfahren liegt in der Behandlung der universell quantifizierten Formeln. Wie oben bereits beschrieben wurde, verwendet die EP-Tableaux-Methode anstelle der klassischen γ-Regel eine Art von Unifikation. Dagegen wird z.B. in dem Verfahren von [Lor94] die klassische γ-Regel eingesetzt. In der Implementierung dieses Verfahrens wird dann die unkontrollierte Instantiierung aufgrund der γ-Regel dadurch kontrolliert, daß eine obere Schranke für

die Anzahl der Anwendungen der γ-Regel vorgegeben wird. Dies kann sich jedoch in der Praxis als schwierig erweisen.

Schließlich seien noch einige Erweiterungen von Tableaux-Verfahren erwähnt, deren Ziel die Generierung einer endlichen Repräsentation eines (möglicherweise unendlichen) Modells einer Formelmenge ist [CZ93, Tam91, FL96, Pel97]. Die Methode, die in [Pel97] beschrieben wird, dient dazu, mittels „Gleichheits-Constraints“ Modelle einer Formelmenge aus möglicherweise unendlichen Zweigen eines Tableau zu extrahieren. Die Ansätze in [Tam91, FL96] basieren auf Resolution und sind wesentlich effizienter als die Verfahren, die auf der δ-Regel des klassischen Tableau basieren. Allerdings kann die Methode, die in [Tam91] vorgestellt wird, im Gegensatz zur EP-Tableaux-Methode, nur auf bestimmte Klassen von Formeln (nämlich der monadischen und der Ackermann-Klasse) angewendet werden. Die Methode von [FL96], die wie die EP-Tableaux-Methode auf der positiven Hyperresolution basiert, vermeidet die Verzweigung bei der Behandlung von Disjunktionen, wie es sowohl im klassischen Tableau als auch bei der EP-Tableaux-Methode der Fall ist. Dies führt in manchen Fällen zu einer Effizienzsteigerung und erlaubt zudem die endliche Repräsentation unendlicher Modelle von Formelmengen, die nicht endlich erfüllbar sind. Diese Eigenschaft ist jedoch für die Anwendungen, die den Entwurf der EP-Tableaux-Methode motivierten, nicht hilfreich, denn das Interesse gilt den erzeugten endlichen Modellen.

8.3 Ausblick

Einige Bereiche, in denen die EP-Tableaux-Methode angewendet werden kann, wurden in den Kapiteln 6 und 7 bereits vorgestellt. Darüberhinaus sind jedoch noch eine ganze Reihe von Anwendungen denkbar, die im folgenden Abschnitt 8.3.1 kurz angedeutet werden. Allerdings ist für manche Anwendungen eine Erweiterung der EP-Tableaux-Methode wünschenswert oder sogar notwendig. Welche Erweiterungen der EP-Tableaux-Methode dabei von Interesse sind, wird im Anschluß in Abschnitt 8.3.2 beschrieben.

8.3.1 Weitere Anwendungen

Aktualisierung von Datenbanksichten („View Updates“)

Eine einfache sogenannte „Sicht“ oder auch „View“ einer Datenbank kann als geschlossene PRQ-Formel der Form $\forall \bar{x}\ (P_1 \wedge \ldots \wedge P_n \rightarrow C)$ definiert werden, wobei $P_1, \ldots, P_n$ und C Atome sind. Mittels einer solchen „Sicht“ können Instanzen von C aus Instanzen der P_i berechnet werden statt sie explizit abzuspeichern. Auf diese Weise wird die Datenbank nicht unnötig durch „C-Daten“ aufgebläht.

Soll die „Sicht“, also die Menge der ableitbaren „C-Daten“, aktualisiert werden, so müssen entsprechend der gewünschten Aktualisierung der „Sicht“ die notwendigen Änderungen an den P_i-Daten bestimmt werden. Dieses Problem wurde bereits als Modellgenerierungsproblem formuliert [Bry90, AB97].

Da sinnvolle Lösungen des „View Update“-Problems aber endlich sein müssen, ist hier die Anwendung eines Modellgenerierungsverfahrens notwendig, das vollständig für die Erfüllbarkeit im Endlichen ist. Deshalb bietet sich die Anwendung der EP-Tableaux-Methode an. Allerdings werden bei dem Ansatz, wie er in [Bry90] beschrieben wurde, Funktionssymbole in der zugrundeliegenden Sprache benötigt. Vermutlich ist eine Übertragung dieses Ansatzes auf eine Sprache ohne Funktionssymbole möglich. Stünde jedoch die EP-Tableaux-Methode auch für Sprachen mit Funktionssymbolen zur Verfügung, könnte der Ansatz direkt übernommen werden.

Planung

Planungsprobleme können ebenfalls als Modellgenerierungsprobleme beschrieben werden [KS92]. Das Planungsproblem besteht dann aus einer Formalisierung des Anfangszustands, möglicherweise mehrerer Endzustände sowie einer Menge von möglichen Aktionen in der Prädikatenlogik der ersten Stufe, so daß jedes Modell der Formalisierung einen gültigen Plan repräsentiert. Da ausführbare Pläne aus höchstens endlich vielen Aktionen bestehen und wir annehmen können, daß es nur endlich viele Endzustände gibt, sind die Modelle, die ausführbare Pläne repräsentieren, endlich. Deshalb muß auch hier ein Modellgenerierungsverfahren angewendet werden, das vollständig für die Erfüllbarkeit im Endlichen ist und es scheint sinnvoll, die EP-Tableaux-Methode anzuwenden.

Ähnlich wie bei der Diagnose (vgl. Abschnitt 6.1) sind auch beim Planungsproblem möglicherweise nicht immer alle erzeugten Modelle, d.h. gültigen Pläne, von Interesse. So ist es vorstellbar, daß Pläne gesucht werden, in denen möglichst wenige Aktionen ausgeführt werden. Dies entspricht der Suche nach minimalen Modellen, wobei (genau wie bei der Diagnose) zunächst präzisiert werden muß, worauf sich der Minimalitätsbegriff bezieht [EG98]. Abhängig davon, wie eine Aktion modelliert wurde, kann sich der Minimalitätsbegriff hier auf eine bestimmte Relation oder auf eine bestimmte Teilmenge der Domäne beziehen. Aufgrund dieser Überlegungen scheint es sinnvoll, zu untersuchen, welche Minimalitätsbegriffe in welchem Zusammenhang sinnvoll sind. Dies könnte dann zu einer Erweiterung der EP-Tableaux-Methode dahingehend führen, daß ausschließlich minimale Modelle generiert werden. In Abschnitt 8.3.2 wird diese Thematik ausführlicher beschrieben.

Programmverifikation

Auch in der Programmverifikation wird Modellgenerierung eingesetzt. Die Idee dabei ist, Modellgenerierungsverfahren auf eine logische Repräsentation eines Programms anzuwenden. Die erzeugten Modelle repräsentieren dann mögliche Abläufe des Programms. Anhand dieser Modelle können Fehler im Programm erkannt werden. Die Modelle, die einen fehlerhaften Ablauf des Programms repräsentieren, sind sozusagen Gegenbeispiele für die Korrektheit des Programms. Existieren sogar endliche Gegenbeispiele, so ist es sicher hilfreich, wenn diese auch erzeugt werden. Dies spricht für eine Anwendung der EP-Tableaux-Methode in der Programmverifikation. Andererseits macht es die Beschreibung komplexer

Datenstrukturen, wie etwa „arrays“, unabdingbar, Funktionssymbole in der zugrundeliegenden Sprache zur Verfügung zu stellen. Ein „array“ etwa entspricht einer Funktion und Elemente eines „array“ werden durch zusammengesetzte Terme modelliert. Zudem stellt sich die Frage, ob für die Modellierung von Programmabläufen der Verwendung von Temporallogik, wie in [Krö87, BBC+96] beschrieben wurde, nicht besser geeignet ist. Genauso könnte natürlich auch an eine Erweiterung der EP-Tableaux-Methode auf Modal- oder Temporallogik [Brü98] gedacht werden.

Theorembeweisen

Um zu beweisen, daß eine Behauptung C aus einer Menge $\mathcal{S}$ von Formeln folgt, wird im allgemeinen die Unerfüllbarkeit von $\mathcal{S} \cup \{\neg C\}$ gezeigt. Dazu werden Refutationsmethoden eingesetzt. Ist jedoch die Vermutung, daß C aus $\mathcal{S}$ folgt, falsch, so ist die Anwendung eines Refutationsverfahrens wenig hilfreich. Ein Modellgenerierungsverfahren, angewendet auf $\mathcal{S} \cup \{\neg C\}$, erzeugt dagegen ein Gegenbeispiel für die Behauptung, also ein Modell von $\mathcal{S} \cup \{\neg C\}$. Dieses Modell kann dann dazu verwendet werden, die Behauptung zu korrigieren. Auch in diesem Falle sind möglichst einfache, also „kleine“ Gegenbeispiele wünschenswert („Occam'sches Prinzip“). Die Anwendung der EP-Tableaux-Methode scheint hier deshalb sinnvoll, da sie sowohl widerlegungsvollständig ist (und deshalb als Refutationsverfahren eingesetzt werden kann) als auch vollständig für die Erfüllbarkeit im Endlichen ist (also möglichst kleine Gegenbeispiele erzeugt).

8.3.2 Erweiterungen der EP-Tableaux-Methode

Wie wir gesehen haben, ist es für einige Anwendungen wünschenswert und teilweise sogar notwendig, die EP-Tableaux-Methode in verschiedene Richtungen zu erweitern. Im folgenden werden verschiedene Erweiterungsmöglichkeiten beschrieben.

Funktionssymbole

Die der EP-Tableaux-Methode zugrundeliegende Sprache ist eine Sprache ohne Funktionssymbole. Das ist für manche Anwendungen wie die Überprüfung der Korrektheit von Datenbankintegritätsbedingungen (siehe Kapitel 7) natürlich, für andere Anwendungen wie die Programmverifikation ist die Verwendung von Funktionssymbolen dagegen essentiell. Da auch in der natürlichen Sprache manches wie „... die Eltern von ...“ funktional ausgedrückt wird, wäre eine Sprache mit Funktionssymbolen in diesem Sinn natürlicher. Wie wir jedoch in Kapitel 4 gesehen haben, ist die Vollständigkeit der EP-Tableaux-Methode für die Erfüllbarkeit im Endlichen nicht mehr gewährleistet, falls die EP-Tableaux-Methode auf Formelmengen angewendet wird, in denen Funktionssymbole vorkommen. Deshalb muß nach anderen Möglichkeiten gesucht werden, Funktionen zu behandeln.

Eine Möglichkeit, Funktionen zu behandeln, ist ihre Darstellung durch ihren Graphen. Eine Funktion wird also mittels einer Relation repräsentiert.

Beispiel 8.1 Statt der Formel $\forall x\ (p(x) \rightarrow p(f(x)))$ wird die Formel

$$\forall x\ (p(x) \rightarrow \exists y\ (p(y) \wedge wert_f(x, y)))$$

betrachtet. Um die Funktion f zu repräsentieren, wird also ein zusätzliches Relationssymbol $wert_f$ eingeführt.

Um sicherzustellen, daß eine Relation auch wirklich eine Funktion darstellt, muß die Wohldefiniertheit und die Totalität formalisiert werden. Um die Totalität einer Funktion zu beschreiben, werden Existenzquantoren eingesetzt, die in der EP-Tableaux-Methode bereits zur Verfügung stehen. Die Formel $\forall x\ \exists y\ wert_f(x, y)$ beschreibt die Totalität der Funktion f in Beispiel 8.1. Mit Hilfe eines zusätzlichen Prädikats D zur Repräsentation der Domäne eines potentiellen Modells, wie in Abschnitt 4.3 beschrieben wurde, kann diese Formel in die geforderte PRQ-Syntax gebracht werden: $\forall x\ (D(x) \rightarrow \exists y\ wert_f(x, y))$.

Um die Wohldefiniertheit zu beschreiben, ist in eingeschränktem Maß, nämlich zum Testen, die Gleichheit notwendig. Die Formel $\forall x \forall y \forall z (wert_f(x, y) \wedge wert_f(x, z) \rightarrow y = z)$ beschreibt die Wohldefiniertheit der Funktion f in Beispiel 8.1. Die volle Gleichheit steht in der EP-Tableaux-Methode bisher nicht zur Verfügung. Allerdings haben wir bereits in Abschnitt 5.1 gesehen, daß eine „eingeschränkte Gleichheit“ behandelt werden kann. Trotzdem muß die Erweiterung der EP-Tableaux-Methode um die Gleichheit untersucht werden (siehe unten).

Eine andere Möglichkeit, Funktionssymbole zu behandeln, könnte die Kombination der EP-Tableaux-Methode mit Constraint-Techniken sein, wie sie bereits zur Behandlung von Existenzquantoren eingesetzt wurden [AS97].

Gleichheit

Noch wichtiger als die Integration der Funktionssymbole in die EP-Tableaux-Methode scheint für viele Anwendungen sowie auch hinsichtlich einer möglichst natürlichen Sprache die Integration der Gleichheit zu sein.

Eine Möglichkeit wäre hier, die EP-Tableaux-Methode mit bereits bestehenden Methoden für die Gleichheit [Her93, DdS94, Rud98] zu kombinieren.

Ein wesentlich einfacherer Ansatz dagegen nützt die Bereichsbeschränkung der zugrundeliegenden Sprache aus. Aufgrund der Bereichsbeschränkung reduziert sich die Gleichheit in der PRQ-Syntax in bestimmten Fällen auf die Gleichheit zwischen Grundtermen.

Beispiel 8.2 Betrachten wir noch einmal die Formalisierung der Wohldefiniertheit der Funktion f aus Beispiel 8.1:

$$\forall x \forall y \forall z (wert_f(x, y) \wedge wert_f(x, z) \rightarrow y = z).$$

Durch die Instantiierung der Variablen x, y und z reduziert sich die Auswertung der Gleichheit auf die Auswertung der Gleichheit zwischen Grundtermen.

Die Gleichheit zwischen Grundtermen scheint nun aber wesentlich einfacher zu behandeln und auch zu implementieren zu sein als die volle Gleichheit [BK98]. In diesem Zusammenhang stellen sich also zwei Fragen. Einmal muß genau untersucht werden, wie die oben beschriebene eingeschränkte Gleichheit in die EP-Tableaux-Methode zu integrieren ist und zweitens sollte geklärt werden, für welche Anwendungen die eingeschränkte Gleichheit ausreichend ist. Möglicherweise eröffnen sich durch die Behandlung der eingeschränkten Gleichheit bereits Anwendungsgebiete, die bisher aufgrund der fehlenden Gleichheit verschlossen waren.

Sorten und Typen

Die PRQ-Syntax, die der EP-Tableaux-Methode zugrunde liegt, erlaubt keine Sorten oder Typen. Die Einführung von Sorten oder Typen ist aber in mehrerlei Hinsicht wünschenswert.

Einmal sind Typen in der natürlichen Sprache völlig „natürlich“. So kann etwa in „alle Kinder spielen“ die Bezeichnung „Kind“ oder in „es gibt eine Person, die ...“ die Bezeichnung „Person“ als Typ aufgefaßt werden. Hinsichtlich der Formalisierung von natürlichsprachlichen Diskursen stellt die Verwendung von Sorten oder Typen sogar eine Vereinfachung dar (wie in Abschnitt 6.3.2 beschrieben wurde). Wird etwa über Personen und Getränke gesprochen, so ist es einfacher, das implizite oder Weltwissen, daß Personen und Getränke i.a. zwei disjunkte Mengen sind, über Sorten oder Typen zu formalisieren anstatt weitere Formeln für die Formalisierung der Disjunktheit hinzuzunehmen. Deshalb wäre die Einführung von Typen in die PRQ-Syntax eine Erweiterung in Richtung „natürlichere“ Syntax.

Dies würde zum zweiten eine „natürlichere“ Formulierung der Spezifikationen in Anwendungen ermöglichen, was für Anwendungen wie das System SIC im Datenbankenbereich oder für Diagnosesysteme von Bedeutung ist. Wird zudem das Ziel einer „natürlichsprachlichen“ Schnittstelle, wie in Abschnitt 6.3.1 beschrieben, verfolgt, so bietet es sich an, zusätzlich zu untersuchen, ob auch die Sprache ACE Typen unterstützen könnte.

Drittens kann durch die Einführung von Typen möglicherweise auch eine Effizienzsteigerung des Verfahrens erreicht werden, da bei der Quantifizierung nur noch Elemente des passenden Typs in Betracht gezogen werden.

Zu diesem Themenkreis kann auf einen reichen Fundus an Ergebnissen zu mehrsortigen Logiken für den Resolutionskalkül zurückgegriffen werden [SS89].

Eine objekt-orientierte Sprache

Die Erweiterung der EP-Tableaux-Methode auf eine objekt-orientierte Sprache würde weitere Anwendungen erleichtern.

Einmal könnte die Überprüfung der Korrektheit von Integritätsbedingungen unmittelbar auf objekt-orientierte Datenbanken ausgeweitet werden. Die Überprüfung der Korrektheit von Integritätsbedingungen in objekt-orientierten Datenbanken wurde bereits in [MP96, BS98a, BBLS98] behandelt. Das Verfahren von [BBLS98] wurde jedoch für eine eingeschränkte Sprache entworfen. In den Arbeiten von [MP96, BS98a] wurde ein Modell-

generierungsverfahren verwendet, das nicht vollständig für die Erfüllbarkeit im Endlichen ist, obwohl diese Eigenschaft hier wesentlich ist. Das System von [BS98a] verwendet eine Implementierung des klassischen Tableaux-Verfahren, wie es in [Fit90] beschrieben wurde. Deshalb könnte eine Einbettung der EP-Tableaux-Methode, erweitert auf eine objekt-orientierte Sprache, in das System von [BS98a] durchaus möglich und sinnvoll sein.

Zum zweiten würde sich durch eine Erweiterung der EP-Tableaux-Methode auf eine objekt-orientierte Sprache eine andere Möglichkeit einer Anbindung der kontrollierten Sprache ACE [FS96] ergeben. Anstatt über die von Attempto [SF96] ausgegebenen Diskursrepräsentationsstrukturen zu gehen (siehe Abschnitt 6.3.2), könnte versucht werden, direkt eine Übersetzung von ACE in die objekt-orientierte Sprache der erweiterten EP-Tableaux-Methode anzugeben.

Modallogik

Da es bereits Vorarbeiten für eine Erweiterung der PUHR-Tableaux-Methode auf die Modallogik gibt [Brü98], würde es sich anbieten, das Verfahren von [Brü98] mit der EP-Tableaux-Methode zu kombinieren. Ein solches kombiniertes Verfahren könnte möglicherweise zur Überprüfung der Korrektheit „dynamischer Integritätsbedingungen“ [Pac97] oder in der Programmverifikation angewendet werden.

Minimale Modelle

Für manche Anwendungen wie etwa die Diagnose ist es nicht nur wichtig, daß die erzeugten Modelle endlich sind. Darüberhinaus sollten sie auch minimal sein. Im Bereich der Diagnose etwa sind „überflüssige“ Erklärungen sicher nicht hilfreich, auch wenn sie nicht widersprüchlich sind [BFFN97a]. Springt etwa ein Auto nicht an, so ist es dienlich zu wissen, daß die Zündkerzen kaputt sind. Die zusätzliche Erklärung, daß die Motorhaube verrostet ist, hilft dagegen nicht weiter.

Für die Generierung ausschließlich minimaler Modelle bietet es sich an, bereits existierende Methoden, wie sie etwa in den Arbeiten von Niemelä [Nie96b, Nie96a] beschrieben wurden oder wie die MM-Prozedur (siehe Abschnitt 3.4), in die EP-Tableaux-Methode zu integrieren. Wie wir gesehen haben, besteht die MM-Prozedur aus dem Complement-Splitting und der „beschränkten Suche“. Das Prinzip der „beschränkten Suche“ auf die EP-Tableaux-Methode zu übertragen ist vermutlich unproblematisch. Die Übertragung des Complement-Splitting erfordert dagegen etwas mehr Anstrengung. So wird beim Complement-Splitting eine Disjunktion $F_1 \vee F_2$ dadurch behandelt, daß sich das Tableau verzweigt und auf der linken Seite die Formel F_1 sowie die Negation der Formel F_2 in das Tableau eingefügt wird. Auf der rechten Seite wird wie üblich die Formel F_2 in das Tableau eingefügt. Solange die Formel F_2, wie im Falle der PUHR-Tableaux-Methode, immer ein Atom ist, bringt dies keine Schwierigkeiten, da auch die Negation $\neg F_2$ als Klausel $F_2 \rightarrow \bot$ darstellbar ist und durch die PUHR-Tableaux-Methode oder die EP-Tableaux-Methode behandelt werden kann. Bei der EP-Tableaux-Methode ist aber eine wesentlich komplexere Syntax zugelassen. So sind Disjunktionen wie $p(a) \vee \forall x\ (q(x) \rightarrow r(x))$ erlaubt. Durch

Negation der Teilformel $\forall x\ (q(x) \rightarrow r(x))$ entsteht die Formel

$$\forall x\ (q(x) \rightarrow r(x)) \rightarrow \bot.$$

Diese Formel ist aber nicht mehr in der PRQ-Syntax, da universell quantifizierte Formeln keine positiven Formeln sind. Ein Lösungsansatz für dieses Problem scheint die Propagation der Negation zu sein. Die EP-Tableaux-Expansionsregeln würden um folgende Regeln erweitert werden:

$$\frac{\neg\forall\bar{x}\ F}{\exists\bar{x}\neg F} \qquad \frac{\neg\exists x\ F}{\forall x\ \neg F} \qquad \frac{\neg(F_1 \wedge F_2)}{\neg F_1 \vee \neg F_2} \qquad \frac{\neg(F_1 \vee F_2)}{\neg F_1 \wedge \neg F_2} \qquad \frac{\neg E}{E \rightarrow \bot} \quad \text{für atomare Formeln } E$$

Doch dieser Problemkreis muß noch genauer untersucht werden, da z.B. durch die Propagation der Negation Formeln der Form $\exists\bar{x}F$ entstehen, wobei $\bar{x}$ ein Tupel von Variablen ist. Für existentiell quantifizierte Formeln läßt die PRQ-Syntax aber keine mehrfache Quantifizierung zu.

Zusätzlich zur Integration des Complement-Splitting in die EP-Tableaux-Methode ist auch eine Ausweitung des Complement-Splitting auf die $\exists$-Regel denkbar, da existentiell quantifizierte Formeln eigentlich wie eine Disjunktion mit einer variablen Anzahl von Disjunktionsgliedern behandelt werden. Allerdings könnte gerade diese variable Anzahl der Disjunktionsglieder problematisch sein.

Alle diese Überlegungen beziehen sich jedoch immer auf den in dieser Arbeit zugrunde gelegten Minimalitätsbegriff (siehe Definition 2.39). In Abschnitt 6.1 wurde jedoch bereits erwähnt, daß für verschiedene Anwendungen möglicherweise auch verschiedene Minimalitätsbegriffe sinnvoll sind. Der Minimalitätsbegriff in Definition 2.39 basiert auf der Mengeninklusionsrelation für die Mengen der in verschiedenen Modellen erfüllten Grundatome. Genauso könnte ein Minimalitätsbegriff aber auch bezüglich der Mächtigkeit der Domäne verschiedener Modelle definiert werden. Im Bereich der Diagnose scheint sogar ein Minimalitätsbegriff sinnvoll, der sich nur auf bestimmte Relationen bezieht. Möglicherweise könnte hier die Methode der „circumscription“ [McC80] angewendet werden. Im Bereich der Planung könnte dagegen ein Minimalitätsbegriff, der sich nur auf die Mächtigkeit einer Teilmenge der Domäne, etwa die Anzahl der Aktionsschritte bezieht, angebracht sein [BK98, Eis98, EG98].

Nach der Klärung dieser Fragen, wäre die Anpassung der MM-Prozedur auf den jeweils passenden Minimalitätsbegriff der nächste Schritt.

Anhang A

CS-SATCHMO

Das CS-SATCHMO-Programm findet sich auf der nächsten Seite.

```
cs_satisfiable :-
        findall(K, violated_instance(K), Set),
        not (Set = []),
        !,
        cs_satisfy_all(Set),
        cs_satisfiable.
cs_satisfiable.

violated_instance(B => H) :-
        (B => H),
        B,
        not H.

cs_satisfy_all([]).
cs_satisfy_all([_B => H | T]) :-
        H,
        !,
        cs_satisfy_all(T).
cs_satisfy_all([_B => H | T]) :-
        cs_satisfy(H),
        cs_satisfy_all(T).

cs_satisfy(E) :-
        cs_component(Atom, Suffix, E),
        not (Atom = false),
        assume(Atom),
        assume_neg(Suffix).

cs_component(Atom, Suffix, (Atom; Suffix)).
cs_component(Atom, Suffix, (_Atom ; Rest)) :-
        !,
        cs_component(Atom, Suffix, Rest).
cs_component(Atom, false, Atom).

assume(Atom) :-
        asserta(Atom).
assume(Atom) :-
        once(retract(Atom)),
        fail.

assume_neg(false) :-
        !.
assume_neg(E) :-
        assume(E => false).
```

Abbildung A.1: Das CS-SATCHMO-Programm

Anhang B

MM-SATCHMO

```
minimal_model :-
        mm(true).

mm(K1) :-
        findall(K, violated_instance(K), Set),
        not (Set = []),
        !,
        mm_satisfy_all(Set, K1, K2),
        mm(K2).
mm(K) :-
        asserta(K => false).

violated_instance(B => H) :-
        (B => H),
        B,
        not H.

mm_satisfy_all([], K, K).
mm_satisfy_all([_B => H | T], K1, K3) :-
        H,
        !,
        mm_satisfy_all(T, K1, K3).
mm_satisfy_all([_B => H | T], K1, K3) :-
        mm_satisfy(H, A),
        and_merge(A, K1, K2),
        mm_satisfy_all(T, K1, K2).
```

Abbildung B.1: Das MM-SATCHMO-Programm, Teil 1

```
mm_satisfy(E, Atom) :-
        cs_component(Atom, Suffix, E),
        not (Atom = false),
        assume(Atom),
        assume_neg(Suffix).

and_merge(Atom, true, Atom) :-
        !.
and_merge(Atom, Conj, (Atom, Conj)).

cs_component(Atom, Suffix, (Atom; Suffix)).
cs_component(Atom, Suffix, (_Atom ; Rest)) :-
        !,
        cs_component(Atom, Suffix, Rest).
cs_component(Atom, false, Atom).

assume(Atom) :-
        asserta(Atom).
assume(Atom) :-
        once(retract(Atom)),
        fail.

assume_neg(false) :-
        !.
assume_neg(E) :-
        assume(E => false).
```

Abbildung B.2: Das MM-SATCHMO-Programm, Teil 2

Literaturverzeichnis

[AB97] Chandrabose Aravindan und Peter Baumgartner. A Rational and Efficient Algorithm for View Deletion in Databases. In *Proc. Int. Logic Programming Symposium*. MIT Press, 1997.

[AHV95] Serge Abiteboul, Richard Hull und Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[AS97] Slim Abdennadher und Heribert Schütz. Model Generation with Existentially Quantified Variables and Constraints. In *Proc. 6th Int. Conf. on Algebraic and Logic Programming*, LNCS 1298. Springer-Verlag, 1997.

[BA93] M. Ben-Ari. *Mathematical Logic for Computer Science*. Prentice Hall, 1993.

[BBC+96] Nikolaj S. Bjørner, Anca Browne, Eddie Chang, Michael Colón, Arjun Kapur, Zohar Manna, Henny B. Sipma und Thomás E. Uribe. STeP: Deductive-Algorithmic Verification of Reactive and Real-time Systems. In *8th Int. Conf. on Computer-Aided Verification*, LNCS 1102. Springer-Verlag, 1996.

[BBE+96] Thomas Brüggemann, François Bry, Norbert Eisinger, Tim Geisler, Sven Panne, Heribert Schütz, Sunna Torge und Adnan Yahya. Satchmo: Minimal Model Generation and Compilation (System Description). In *Proc. V-èmes Journées Francophones de Programmation en Logique et Programmation par Contraintes (JFPLC)*, 1996.

[BBLS98] Domenico Beneventano, Sonia Bergamaschi, Stefano Lodi und Claudio Sartori. Consistency Checking in Complex Object Database Schemata with Integrity Constraints. *Journal of IEEE Transaction on Knowledge and Data Engineering*, 10(4), 1998.

[BDM88a] François Bry, Hendrik Decker und Rainer Manthey. A Uniform Approach to Constraint Satisfaction and Constraint Satisfiability in Deductive Databases. In *Proc. of 1st Int. Conf. on Extending Database Technology*, 1988.

[BDM88b] François Bry, Hendrik Decker und Rainer Manthey. A Uniform Approach to Constraint Satisfaction and Constraint Satisfiability in Deductive Databases. In *Proc. Int. Conf. Extending Data Base Technology (EDBT)*, LNCS 303, Venice, Italy, March 1988. Springer-Verlag.

[BEST98a] François Bry, Norbert Eisinger, Heribert Schütz und Sunna Torge. SIC: An Interactive Tool for the Design of Integrity Constraints (System Description). In *Proc. EDBT'98 Demo Session*, 1998.

[BEST98b] François Bry, Norbert Eisinger, Heribert Schütz und Sunna Torge. SIC: Ein interaktives Deduktionssystem zur Überprüfung der Erfüllbarkeit von Integritätsbedingungen in Datenbanken (Systembeschreibung). In *In: Proc. Ïnference-Mechanisms in Knowledge-Based Systems: Theory and Applications", Workshop im Rahmen der 22. Jahrestagung Künstliche Intelligenz (KI-98), Bremen*, 1998.

[BEST98c] François Bry, Norbert Eisinger, Heribert Schütz und Sunna Torge. SIC: Satisfiability Checking for Integrity Constraints. In *Proc. Deductive Databases and Logic Programming (DDLP), workshop at JICSLP*, 1998.

[Bet55] E. W. Beth. Semantic Entailment and Formal Derivability. *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, N. R.*, 18(13), 1955. Zitiert in [BS98b], Nachdruck in: Karel Berka und Lothar Kreiser, Herausgeber. *Logik-Texte. Kommentierte Auswahl zur Geschichte der der modernen Logik.* Akademie-Verlag, Berlin, 1986.

[BF98] Peter Baumgartner und Ulrich Furbach. Variants of Clausal Tableaux. In Wolfgang Bibel und Peter H. Schmitt, Hrsg., *Automated Deduction – A Basis for Applications*, Band I, Kapitel 3. Kluwer Academic Publishers, 1998.

[BFFN97a] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach und Wolfgang Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. In *Proc. 15th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1997.

[BFFN97b] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach und Wolfgang Nejdl. Tableaux for Diagnosis Applications. In *Proc. 6th Workshop on Theorem Proving with Tableaux and Related Methods*, LNAI. Springer-Verlag, 1997.

[BFK+98] Peter Baumgartner, Ulrich Furbach, Michael Kohlhase, William McCune, Wolfgang Reif, Mark Stickel und Tomás Uribe, Hrsg. *Problem-solving Methodologies with Automated Deduction, Workshop at CADE-15*, Lindau, Germany, Juli 1998.

[BFN96] Peter Baumgartner, Ulrich Furbach und Ilkka Niemelä. Hyper Tableaux. In *Proc. 5th Europ. Workshop on Logics in AI (JELIA)*, LNCS 1126. Springer-Verlag, 1996.

[Bib92] Wofgang Bibel. *Deduktion.* R. Oldenbourg Verlag München, 1992.

[BK98] François Bry und Karsten Konrad. Über die Herbrandgleichheit. Persönliche Mitteilung, Mai 1998.

[BM86] François Bry und Rainer Manthey. Checking Consistency of Database Constraints: A Logical Basis. In *Proc. 12th Int. Conf. on Very Large Data Bases (VLDB)*, 1986.

[BM87] François Bry und Rainer Manthey. Proving Finite Satisfiability of Deductive Databases. In *Proc. 1st Workshop on Computer Science Logic (CSL)*, LNCS 329, Seiten 44–55. Springer-Verlag, 1987.

[BMM91] François Bry, Rainer Manthey und Bernd Martens. Integrity Verification in Knowledge Bases. In *Proc. of 2nd Russian Conf. on Logic Programming*, 1991. (invited paper).

[Brü98] Thomas Brüggemann. Zwei Modellgenerierungsverfahren für die Modallogik. Technical Report PMS-FB-1998-7, Institut für Informatik, LMU München, 1998.

[Bry90] François Bry. Intensional Updates: Abduction via Deduction. In *Proc. 7th Int. Conf. on Logic Programming*, Seiten 561–575, 1990.

[BS98a] Véronique Benzaken und Xavier Schaefer. Static Management of Integrity in Object-Oriented Databases: Design and Implementation. In *Proc. 6th Int. Conf. on Extending Database Technology*, LNCS 1377, 1998.

[BS98b] Wolfgang Bibel und Peter H. Schmitt. *A Basis for Application*, Band 1, Foundations – Calculi and Methods. Kluwer Academic Publishers, 1998.

[BT98] François Bry und Sunna Torge. A Deduction Method Complete for Refutation and Finite Satisfiability. In *Proc. 6th Europ. Workshop on Logics in AI (JELIA)*, LNAI. Springer-Verlag, 1998.

[BY96] François Bry und A. Yahya. Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux. In *Proc. 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, LNAI 1071. Springer-Verlag, 1996.

[CDHM64] T.J. Chinlund, M. Davis, P.G. Hineman und M.D. McIlroy. Theorem Proving by Matching. Technical report, Bell Laboratory, 1964.

[Cha72a] C.L. Chang. Theorem Proving by Generation of Pseudo-Semantic Trees. Technical report, Nat. Inst. of Health, Bethesda, Maryland, 1972.

[Cha72b] C.L. Chang. Theorem Proving with Variable-Constrained Resolution. *Information Science*, 4, 1972.

[CP87] Philip Cox und Tomasz Pietrzykowski. General Diagnosis by Abductive Inference. In *Proc. Symposium of Logic Programming*, 1987.

[CZ93] Ricardo Caferra und N. Zabel. Building Models by using tableaux extended by equational problems. *Journal of Logic and Computation*, 3:3–25, 1993.

[Dat95] C.J. Date. *An Introduction to Database Systems.* Addison-Wesley, 6. Auflage, 1995.

[Dav63] Martin Davis. Eliminating the Irrelevant from Mechanical Proofs. In *Proc. Symp. Applied Mathematics*, Band 15, 1963.

[DdS94] Marc Denecker und Danny de Schreye. On the Duality of Abduction and Model Generation in a Framework for Model Generation with Equality. *Theor. Comp. Sc.*, 122:225–262, 1994.

[DTU96] Hendrik Decker, Ernest Teniente und Toni Urpí. How to Tackle Schema Validation by View Updating. In *Proc. 5th Int. Conf. on Extending Database Technology (EDBT'96)*, LNCS 1057, Avignon (France), 1996. Springer-Verlag.

[EFT92] Heinz-Dieter Ebbinghaus, Jörg Flum und Wolfgang Thomas. *Einführung in die mathematische Logik.* BI-Wissenschafts-Verlag, 1992. 3., vollständig überarbeitete und erweiterte Auflage.

[EG98] Norbert Eisinger und Tim Geisler. Problem Solving with Model-Generation Approaches based on PUHR Tableaux. Technical Report PMS-FB-1998-8, Institut für Informatik, LMU München, 1998. In [BFK+98].

[Eis98] Norbert Eisinger. Über verschiedene Minimalitätsbegriffe. Persönliche Mitteilung, 1998.

[Fit90] Melvin Fitting. *First Order Logic and Automated Theorem Proving.* Springer-Verlag, 1990.

[FL96] Christian Fermüller und Alexander Leitsch. Hyperresolution and Automated Model Building. *Journal of Logic and Computation*, 6(2):173–203, 1996.

[FN92] Gerhard Friedrich und Wolfgang Nejdl. MOMO - Model-Based Diagnosis for Everybody. In *Proc. IEEE Conf. on Artificial Intelligence Applications*, 1992.

[FS96] Norbert E. Fuchs und Rolf Schwitter. Attempto Controlled English (ACE). In *Proc. 1st Int. Workshop on Controlled Language Applications (CLAW)*, Leuven, Belgium, March 1996.

[Fur98] Ulrich Furbach. *Automated Deduction – A Basis for Applications.* Tableaux and Connection Calculi. Part I. Kluwer Academic Publishers, 1998. To appear.

[GHR93] Dov M. Gabbay, C. J. Hogger und J. A. Robinson, Hrsg. *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 1: Logical Foundations.* Oxford University Press, 1993.

[Gil60] P.C. Gilmore. A Proof Method for Quantification Theory; its Justification and Realization. *IBM Journal Res. Develop.*, 1960.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 1930.

[Her93] Laurent Herr. E-SATCHMO: introduction de l'égalité dans le démonstrateur automatique SATCHMO. Technical report, ECRC, 1993.

[Hin55] Jaakko Hintikka. Form and Content in Quantification Theory. *Acta Philosophica Fennica*, 8, 1955. Zitiert in [BS98b].

[Hin88] Jaakko Hintikka. Model Minimization - An Alternative to Circumscription. *Journal of Automated Reasoning*, 4:1–14, 1988.

[Hör81] K. M. Hörnig. Generating Small Models of First Order Axiom. In *GWAI-81*, Informatik-Fachberichte 47. Springer-Verlag, 1981.

[Hue76] Gérard Huet. Résolution d'Équations dans les Langages d'Ordre $1, 2, \ldots, \omega$. Technical report, Université Paris VII, 1976. Thèse de Doctorat d'État (Habilitationsschrift).

[KE97] Mathias Kettner und Norbert Eisinger. The Tableau Browser SNARKS (System Description). In William McCune, Hrsg., *Proc. 14th Int. Conf. on Automated Deduction (CADE)*, LNAI 1249. Springer-Verlag, 1997.

[Ket97] Mathias Kettner. Ein System zur Konstruktion, Modifikation und Untersuchung von PUHR-Tableaux. Diplomarbeit, Ludwig-Maximilians-Universität München, 1997.

[KK71] Robert Kowalski und Donald Kuehner. Linear Resolution with Selection Function. *Artificial Intelligence*, 2(3-4), 1971. Zitiert in [Bib92].

[Kon98] Karsten Konrad. Modellgenerierung in der Linguistik. Persönliche Mitteilung, Mai 1998.

[KR93] Hans Kamp und Uwe Reyle. *From Discourse to Logic.* Kluwer Academic Publishers, 1993.

[Krö87] Fred Kröger. *Temporal Logic of Programs.* Springer-Verlag, 1987.

[KS92] Henry Kautz und Bart Selman. Planning as Satisfiability. In *Proc. 10th European Conf. on Artificial Intelligence*, 1992.

[KSS87] Robert Kowalski, Fariba Sadri und Paul Soper. Integrity Checking in Deductive Databases. In *Proc. of 13th Int. Conf. on Very Large Databases*, 1987.

[Let98] Reinhold Letz. Clausal Tableaux. In Wolfgang Bibel und Peter H. Schmitt, Hrsg., *Automated Deduction – A Basis for Applications*, Band I, Kapitel 2. Kluwer Academic Publishers, 1998.

[Lor94] Sven Lorenz. A Tableau Prover for Domain Minimization. *Journal of Automated Reasoning*, 13:375–390, 1994.

[Lov68] Donald W. Loveland. Mechanical Theorem Provingby Model Elimination. *Journal of the ACM*, 15(2), 1968.

[Lov72] Donald W. Loveland. A Unifying View of Some Linear Herbrand Procedures. *Journal of the ACM*, 19(2), 1972.

[LST84] John Lloyd, E.A. Sonenberg und R.W. Topor. Integrity Constraint Checking in Stratified Databases. *Journal of Logic Programming*, 11, 1984.

[Man90] Rainer Manthey. Satisfiability of Integrity Constraints: Reflections on a Neglected Problem. In *Proc. 2nd Int. Workshop on Foundations of Models and Languages for Data and Objects*, Aigen, Austria, September 1990.

[MB88] Rainer Manthey und François Bry. SATCHMO: A theorem prover implemented in Prolog. In *Proc. 9th Int. Conf. on Automated Deduction (CADE)*, LNCS 310. Springer-Verlag, 1988.

[McC80] John McCarthy. Circumscription – A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.

[MP96] Sophie Monties und Pierre-Yves Policella. Checking Consistency in OODBMS. Technischer Bericht, unveröffentlicht, 1996.

[Nic82] Jean-Marie Nicolas. Logic for Improving Integrity Checking in Relational Databases. *Acta Informatica*, 18(3), 1982.

[Nie96a] Ilkka Niemelä. Implementing Circumscription Using a Tableau Method. In *Proc. 12th European Conf. on Artificial Intelligence*. John Wiley, 1996.

[Nie96b] Ilkka Niemelä. A Tableau Calculus for Minimal Model Reasoning. In *Proc. 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, LNAI 1071. Springer-Verlag, 1996.

[NML93] Andrea Neufeld, Guido Moerkotte und Peter C. Lockemann. Generating Consistent Test Data for a Variable Set of General Consistency Constraints. *VLDB Journal*, 2(2):173–213, 1993.

[Oli92] Nicola Olivetti. Tableaux and Sequent Calculus for Minimal Entailment. *Journal of Automated Reasoning*, 9, 1992.

[Pac97] M. A. Pacheco e Silva. Dynamic Integrity Constraints Definition and Enforcement in Databases: A Classification Framework. In *1st Working Conf. on Integrity and Internal Control in Information Systems*, Band 1. Chapman & Hall, 1997.

[Pel97] Nicolas Peltier. Simplifying and Generalizing Formulae in Tableaux. Pruning the Search Space and Building Models. In William McCune, Hrsg., *Proc. 6th Workshop on Theorem Proving with Tableaux and Related Methods*, LNAI 1249, Seiten 408–411. Springer-Verlag, 1997.

[PGA87] David Poole, Randy Goebel und Romas Aleliunas. *The Knowledge Frontier: Essays in the Representation of Knowledge*, Kapitel „Theorist: A Logical Reasoning System for Defaults and Diagnosis". Springer-Verlag, 1987.

[Poo89a] David Poole. Explanation and Prediction: An Architecture for Default and Abductive Reasoning. *Computational Intelligence*, 5(2):97–110, 1989.

[Poo89b] David Poole. Normality and Faults in Logic-Based Diagnosis. In *Proc. 11th Int. Joint Conf. on Artificial Intelligence*, 1989.

[PP98] M. Paramasivam und David A. Plaisted. Automated Deduction Techniques for Classification in Description Logic Systems. *Journal of Automated Reasoning*, 20(3), 1998.

[Pra60] D. Prawitz. An Improved Proof Procedure. *Theoria*, 26, 1960.

[Pra69] D. Prawitz. Advances and problems, in Mechanical Proof Procedures. In *Machine Intelligence*, Band 4, Seiten 59–71. American Elsevier, 1969.

[Rei87] R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–95, 1987.

[Rob65a] John Alan Robinson. Automatic Deduction With Hyper-Resolution. *Int. Journal of Computational Mathematics*, 1, 1965.

[Rob65b] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Rud98] Thomas Rudlof. *SHR Tableaux. A Framework for Automated Model Generation.* Dissertation, Ludwigs-Maximilians-Universität München, Insitut für Mathematik, 1998.

[Sch56] Kurt Schütte. Ein System des verknüpfenden Schließens. *Archiv für mathematische Logik und Grundlagenforschung*, 2(2-4), 1956. Zitiert in [BS98b].

[SF96] Rolf Schwitter und Norbert E. Fuchs. Attempto – From Specifications in Controlled Natural Language towards Executable Specifications. In *Proc. EMISA Workshop „Natürlichsprachlicher Entwurf von Informationssystemen – Grundlagen, Methoden, Werkzeuge, Anwendungen*, Seiten 163–177. Evang. Akademie Tutzing, Germany, 1996.

[Sla92] John Slaney. Finder (finite domain enumerator): Notes and Guides. Technical report, Australian National University, Canberra, 1992. Automated Reasoning Project.

[Smu68] Raymond Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

[SS89] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*. LNAI 395. Springer-Verlag, 1989.

[Tam91] Tannel Tammet. Using Resolution for Deciding Solvable Classes and Building Finite Models. *Baltic Computer Science*, 1991.

[Tra50] B. A. Trakhtenbrot. Impossibility of an Algorithm for the Decision Problem in Finite Classes. In *Dokl. Acad. Nauk.*, SSSR 70, 1950. in Russisch, Englische Übersetzung in Amer. Soc. Trans., Series 2, 23, 1963.

[ZZ95] Jian Zhang und Hantao Zhang. SEM: a System for Enumerating Models. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Band 1, 1995.

Abbildungsverzeichnis

2.1 Einteilung in α-, β-, γ- und δ-Formeln 23
2.2 Ein Tableau für die Formel $\forall x\ p(x) \vee \exists y\ (q(a,y) \wedge r(y,b))$. 24

3.1 Ein PUHR-Tableau für Beispiel 3.8 32
3.2 Das SATCHMO-Programm 40
3.3 Ein faires PUHR-Tableau für Beispiel 3.22 42
3.4 Ein PUHR-Tableau für Beispiel 3.28 44
3.5 Ein Ablauf der MM-Prozedur für Beispiel 3.34. 47

4.1 Ein EP-Tableau für Beispiel 4.33 71
4.2 Ein EP-Tableau für Beispiel 4.34 72
4.3 Ein EP-Tableau für $\mathcal{S}_3$ in Beispiel 4.50 84
4.4 Ein EP-Tableau für $\mathcal{S}_4$ in Beispiel 4.50 85

5.1 Verwendete Datenstrukturen und Darstellungen in FINFIMO. 89
5.2 Das FINFIMO-Programm, Teil 1 92
5.3 Das FINFIMO-Programm, Teil 2 93

7.1 Ableitungsbaum für Beispiel 7.1 ausgegeben von SIC 117
7.2 Ableitungsbaum für Beispiel 7.1 mit modifizierter Datenbank 118
7.3 Ableitungsbaum für Beispiel 7.1 ausgegeben von SIC 120
7.4 Ableitungsbaum für die verbesserten Integritätsbedingungen aus Beispiel 7.1 ausgegeben von SIC 122

A.1 Das CS-SATCHMO-Programm 134

B.1 Das MM-SATCHMO-Programm, Teil 1 135
B.2 Das MM-SATCHMO-Programm, Teil 2 136

Index

$\top$, 8
$\bot$, 8
frei(F), 9
frei(t), 9
$\models$, 14, 15
$\leq$, 20
$Neg(A_1 \vee \ldots \vee A_n)$, 42
$Neg(\{A_1, \ldots, A_n\})$, 45

Abnormität, 100
allgemeingültig, 14
Alphabet, 7
Atom, 9
    Grund-, 10

Baum, 21
Bereich, 59

Diagnose, 99
    abduktive, 102
    konsistenzbasierte, 100

endlich erfüllbar, 14
EP-Tableaux, 68
erfüllbar, 14
erfüllbarkeitsäquivalent, 15
Erweiterung
    einfache, 76
Expansionsregeln
    EP-, 68
    PUHR-, 30
    $\exists$-Regel, 68
    $\wedge$-Regel, 68
    $\vee$-Regel, 30, 68
    Complement-Splitting-Regel, 42
    Erweiterte PUHR-Regel, 68
    PUHR-Regel, 30

Fairness
    EP-Tableaux, 74
    PUHR-Complement-Tableaux, 43
    PUHR-Tableaux, 36
finitär, 45
Formel, 8
    atomare, 9
    geschlossene, 10
    in Implikationsform, 11
    universelle, 9
    zusammengesetzte, 9

Gleichheit
    eingeschränkte, 91
Graph, 21

Herbrandinterpretation, 17
Hyperresolution, 24

Implikationsform
    Formel in, 11
    Klausel in, 10
Instanz, 12
    Grund-, 12
Interpretation, 13
    Herbrand-, 17
    Term-, 19

Klassische Tableaux, 22
Klausel, 10
    -kopf, 10
    -rumpf, 10
    bereichsbeschränkt, 27
    in Implikationsform, 10
konsistent, 15

Literal, 10

negatives, 10
positives, 10
logisch äquivalent, 15

Matching, 12
MM-Prozedur, 41, 45
Modell, 13
Herbrand-, 17
minimales, 20
Term-, 19

Positive Bedingung, 59
Prädikatenlogik der ersten Stufe, 7
Programm
FINFIMO-Programm, 92, 93
SATCHMO-Programm, 40
CS-SATCHMO-Programm, 133, 134
MM-SATCHMO-Programm, 135, 136
PRQ-Formeln, 59
PUHR-Tableaux, 30

Rang, 78
Redukt, 16
Resolution, 24

Satz, 10
Semantik
Herbrand-, 16
Logik der ersten Stufe, 13
Term-, 18
Spezifikation, 87
Substitution, 12
Subsumption, 12
Suche
beschränkte, 41
Syntax
erweiterte, 53
Logik der ersten Stufe, 7
PRQ-, 59

Tableaux
EP-, 68
Erweiterte PUHR-, 54
Klassische, 22
Positive-Unit-Hyper-, 31
PUHR-, 30, 31
PUHR-Complement-, 43
-Expansionsregeln
EP-, 68
PUHR-, 30
$\exists$-Regel, 68
$\wedge$-Regel, 68
$\vee$-Regel, 30, 68
-Complement-Splitting-Regel, 42
Erweiterte PUHR-Regel, 68
PUHR-Regel, 30
Term, 8
Grund-, 10
Terminterpretation, 19
Transformation, 28, 62

unerfüllbar, 14
Unifikation, 12

Variablen
freie, 9
Variablenbelegung, 13

Danke!

... an François Bry für die Vergabe des schönen Themas, das sehr vielseitige Untersuchungen zuließ und zu einer eigenständigen Arbeit führte, für seine engagierte Betreuung und nicht zuletzt für seine Unterstützung meines Wechsels in die Informatik,

... an Rainer Manthey für die Begutachtung dieser Arbeit und für seine Unterstützung während der Zeit der Promotion,

... an Norbert Eisinger für viele wertvolle Anmerkungen zu einer ersten Version dieser Arbeit und zahlreiche interessante und „nicht fruchtlose“ Diskussionen, die oft weit über meine Arbeit hinausgingen und für die er klaglos so manche Stunde opferte,

... an Slim Abdennadher für seine hilfreichen Hinweise zu Teilen dieser Arbeit sowie seine Unterstützung bei den organisatorischen Details rund um die Promotion,

... an Freia Torge für ihren Einsatz als „Duden“,

... an Heribert Schütz und Thomas Brüggemann für ihre Hinweise zu Teilen der Implementierung, an Martin Josko für seine freundliche und hilfsbereite Art, mir bei technischen Schwierigkeiten zu helfen und an Ellen Lilge für ihre Unterstützung in allen Verwaltungsangelegenheiten,

... an das Graduiertenkolleg SIL und die DFG für die Vergabe des Stipendiums, insbesondere an Franz Guenthner und Klaus Schulz für die Unterstützung und das Interesse an meiner Arbeit,

... und vor allem auch an meine Familie, meine Freunde und alle, die mich aus nah und fern auf die verschiedenste Weise hartnäckig unterstützt haben.

Lebenslauf

Persönliche Daten:

Name:	Sunna Torge
Geburtsdatum:	4. August 1967
Geburtsort:	Heidenheim/Brenz

Ausbildung:

Seit Nov. 1995	Promotion in Informatik an der Ludwig-Maximilians-Universität München und Stipendiatin im Graduiertenkolleg SIL (Sprache, Information, Logik)
WS 87/88 bis SS 1989 und SS 1992 bis SS 1995	Studium der Mathematik und Physik an der Universität Freiburg
September 1995	Diplom in Mathematik
Juli 1989	Baccalaureus in Mathematik
WS 86/87 bis SS 1987	Studium der Musikwissenschaft und Mathematik an der Universität Freiburg
Juni 1986	Abitur am Peutinger-Gymnasium Ellwangen

Tätigkeiten außerhalb der Universität:

1991 bis 1992	Praktikantin im Lektorat des B.I.-Wissenschaftsverlag, Mannheim
1990 bis 1991	Privatstudium Violoncello und Klavier
1989 bis 1990	Praktikantin in einer Lebensgemeinschaft (Camp Hill) mit geistig behinderten Erwachsenen in Lausanne, Schweiz