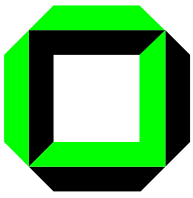
Vergleich und Analyse von Ordnungseinschränkungen für freie Variablen Tableau*

Christian Pape
pape@ira.uka.de



Universität Karlsruhe
Institut für Logik, Komplexität und
Deduktionsysteme

*Dies ist eine überarbeitete Version meiner Diplomarbeit vom 31.Mai 1996.

Inhalt

KAPITEL 1

Einführung

Der Einsatz automatischer Theorembeweiser stellt sich in der Praxis selbst nach ca. 30 Jahren seit der Vorstellung des Resolutionsprinzips (Robinson, 1965b) oftmals als unbefriedigend heraus. Die wesentliche Ursache ist der immense Suchraum, den ein Beweissystem bei vielen Problemen bewältigen muß. Darüber hinaus werden je nach Kalkül meist eine große Anzahl redundanter Ableitungen berechnet, welche oft nur mit hohem Aufwand zu vermeiden sind.

Um diese Probleme anzugehen, wurden für den Resolutionskalkül schon sehr früh nach dessen Veröffentlichung einige Einschränkungen und Verbesserungen beschrieben. Beispielsweise werden bei Hyperresolution (Robinson, 1965a) einzelne Resolutionsschritte zu Makro-Resolutionsschritten zusammengefaßt, ohne daß die Zwischenergebnisse für weitere Ableitungen benutzt werden dürfen. Zusammen mit der Subsumtion von Klauseln, welche redundante Ableitungen verhindert, ergibt dies eine Suchraumeinschränkung, die für die Effizienz existierender Implementierungen hauptsächlich verantwortlich ist.

Bei geordneter Resolution (Maslov, 1971; Kowalski & Hayes, 1969) wird eine Ordnung auf Literalen benutzt, um die Resolventenbildung einzuschränken: Anstatt zwischen beliebigen Literalen zweier Klauseln zu resolvieren, müssen die Literale bezüglich einer bestimmten Ordnung maximal sein, um eine gültige Resolvente beider Klauseln zu erzeugen. Dieser Kalkül ist von theoretischem Interesse, da geordnete Resolution zusammen mit einer geeigneten Ordnungseinschränkung ein Entscheidungsverfahren für viele Formelklassen darstellt. Neue umfangreiche Darstellungen über dieses Thema sind in (Leitsch, 1996) bzw. (Fermüller *et al.*, 1993) zu finden. In letzterem wird weiterhin gezeigt, daß die theoretischen Ergebnisse auch für effiziente Implementierungen ausgenutzt werden können.

Die positiven Erfahrungen mit geordneter Resolution lassen es sinnvoll erscheinen, Ordnungseinschränkungen auch für andere Kalküle als Resolution zu nutzen. Einen vollständigen Kalkül von Ordnungseinschränkungen für semantische Tableaus (Smullyan, 1995) für Formeln in konjunktiver Normalform wurde erstmals in (Klingenbeck & Hähnle, 1994) vorgestellt. In (Hähnle & Klingenbeck, 1996) wird dieser Kalkül für Formeln in Negations-Normal-Form erweitert.

Das wesentliche Ziel dieser Diplomarbeit ist es herauszufinden, wie Ordnungseinschränkungen sinnvoll im Tableaukalkül benutzt werden können. Insbesondere sollen damit Probleme aus der Programmverifikation des Karlsruhe Interactive Verifiers (Reif, 1992) besser behandelt werden.

In Kapitel 3 zeigen wir, wie sich die Kalküle aus (Klingenbeck & Hähnle, 1994) und (Hähnle & Klingenbeck, 1996) noch verallgemeinern lassen: Anstatt einer A-Ordnung können totale Funktionen auf Klauselmengen benutzt werden und die Substitutivitätseigenschaft der A-Ordnung kann abgeschwächt werden. Da wir später Probleme behandeln, die auch Gleichheit enthalten, zeigen wir, wie sich Theorien in A-geordnete Tableaus einbauen lassen.

Im prädikatenlogischen Fall, können Ordnungsconstraints benutzt werden, um bei A-geordneten Tableaus eine möglichst große Einschränkung zu erzielen. Probleme, die sich aus der Verwendung von Ordnungsconstraints ergeben, erörtern wir in Kapitel 4.

Wie haben Ordnungseinschränkungen als Teil des tableau-basierten Theorembeweiser ${}_3T^AP$ (Beckert *et al.*, 1996a; Beckert *et al.*, 1996b) implementiert. In Kapitel 5 berichten wir kurz über die Implementierung und zeigen einige Anwendungen A-geordneter Tableaus.

KAPITEL 2

Bezeichnungen

> "When I use a word", Humpty Dumpty said, in rather a scornful tone, "it means just what I choose it to mean — neither more nor less."
>
> Lewis Carroll —
>
> Through the Looking Glass

Im folgenden definieren wir einige für die nächsten Kapitel grundsätzliche Begriffe, welche normalerweise in jeder umfassenderen Abhandlung über formale Logik oder automatisches Beweisen zu finden sind. Ein Leser, der mit diesen Begriffen vertraut ist, kann deswegen getrost zu den nächsten Abschnitten wechseln, ohne Wesentliches zu verpassen. Der Index und das Symbolverzeichnis im Anhang sollen helfen, im Zweifelsfall die entsprechende Definition zu einem unklaren Begriff schnell nachschlagen zu können. Für Begriffe, die benutzt aber hier nicht eingehender erklärt werden, verweisen wir auf die umfangreiche Literatur (z.B. (Fitting, 1996)).

2.1. Terme und Literale

Das Folgende ist im wesentlichen aus (Fermüller *et al.*, 1993) übernommen.

$\mathbf{V}$ sei eine Menge von Variablensymbolen, welche wir meist mit x, y, z, u, v, w bezeichnen. Die Menge aller Funktionssymbole geben wir mit $\mathbf{F}$, Funktionssymbole mit f, g, h oder im Fall von Konstanten mit c, d, e an. $arity(f)$ sei die Stelligkeit einer Funktion f. So ist z.B. $\mathbf{K} = \{c \in \mathbf{F} | arity(c) = 0\}$ die Menge aller Konstanten.

Definition 2.1 (Terme).

1. Jede Variable und jede Konstante ist ein Term.
2. Sind $t_1, \ldots, t_n$ Terme und f eine Funktion mit $arity(f) = n$, dann ist auch $f(t_1, \ldots, t_n)$ ein Term.
3. Nichts anderes ist ein Term.

Wir bezeichnen Terme mit kleinen Buchstaben wie s, t und die Menge aller Terme mit $\mathbf{T}$. Für die Menge aller Variablen eines Terms t schreiben wir $\mathbf{V}(t)$.[1]

[1] $\mathbf{V}$ wird also mehrdeutig verwendet: Einmal als *Menge* aller Variablen und zum zweiten als *Funktion*!.

DEFINITION 2.2 (Termtiefe $\tau(t)$ eines Terms t).

1. Ist t eine Variable oder eine Konstante, dann gilt $\tau(t) = 0$.
2. Ist $t = f(t_1, \ldots, t_n)$, dann gilt $\tau(t) = 1 + \max\{\tau(t_i) | 1 \leq i \leq n\}$.

DEFINITION 2.3 (Bewertung). Eine *Bewertung* ist eine Abbildung $\# : \mathbf{F} \to \mathbb{N}$.

Wird $\#$ auf Variablen ausgedehnt, so darf $\#(x)$ für alle $x \in \mathbf{V}$ höchstens so groß wie die kleinste Bewertung eines Funktionssymbols sein.

Wir schreiben meist einfach $\#t$ statt $\#(t)$.

DEFINITION 2.4 (Positionen eines Terms t). Die Menge der *Positionen in* t $Pos(t)$ ist eine Menge von Folgen natürlicher Zahlen, für die gilt:

1. ϵ (die *leere Folge*) $\in Pos(t)$.
2. Gilt $p \in Pos(t_i)$, dann auch $i.p \in Pos(t)$, wobei f ein n-stelliges Funktionssymbol, $t = f(t_1, \ldots, t_n)$ und $1 \leq i \leq n$ ist.
3. Nichts anderes ist in $Pos(t)$.

BEISPIEL 2.1.

$$Pos(f(a, f(g(b), a, g(g(b))), g(b))) =$$
$$\{\epsilon, 1, 2, 3, 2.1, 2.2, 2.3, 2.1.1, 2.3.1, 2.3.1.1, 3.1\}$$

Ist die genaue Position für einen bestimmten Sachverhalt uninteressant, so bedeutet die Schreibweise $s[t]$, daß $s \in \mathbf{T}$ eine Position p mit $s_{|p} = t$ besitzt.

Die *Länge* $|p|$ einer Position p ist die Anzahl natürlicher Zahlen von p.

DEFINITION 2.5 (Teilterm). Für den *Teilterm* $t|_p$ eines Terms t an Position p gilt:

1. $t|_\epsilon = t$ und
2. $t|_{i.p} = t_i|_p$, falls $t = f(t_1, \ldots, f_n)$ für ein n-stelliges Funktionssymbol f gilt.

Kommt ein Term s an der Position p in einem Term t vor, dann ist $\tau(p, t) = |p|$ die Tiefe von s in t.

DEFINITION 2.6 (Grundterm). Ein *Grundterm* t ist ein Term, der keine Variablen enthält. Es existiert also keine Position p in t mit $t|p \in \mathbf{V}$.

DEFINITION 2.7 (Maximale Tiefe eines Terms).

$$\tau_{\max}(s, t) = \max\{\tau(p, t) | |t|_p = s, p \in Pos(t)\} \quad .$$

$\mathbf{P}$ sei eine Menge von *Prädikatensymbolen*. Die Elemente von $\mathbf{P}$ geben wir üblicherweise mit P, Q, R an. Die Stelligkeit eines Prädikatensymbols geben wir ebenfalls mit *arity* an.

DEFINITION 2.8 (Atome). Sind $t_1, \ldots, t_n$ Terme und ist P ein n-stelliges Prädikatensymbol, dann heißt $P(t_1, \ldots, t_n)$ *Atom* ($n = 0$ ist möglich).

Für Atome verwenden wir großen Buchstaben wie A, B, C, D.

2.2. Substitutionen

DEFINITION 2.9 (Substitution). Eine *Substitution* ist eine Abbildung $\sigma : \mathbf{V} \to \mathbf{T}$ mit $\sigma(x) = x$ für fast alle $x \in \mathbf{V}$.

Wir bezeichnen die Menge aller Substitutionen mit $\mathbf{S}$ und einzelne Substitutionen mit kleinen griechischen Buchstaben wie σ, τ, μ.

DEFINITION 2.10. Eine Substitution $\sigma : \mathbf{V} \to \mathbf{T}$ wird wie folgt zu einer Abbildung $\sigma : \mathbf{T} \to \mathbf{T}$ erweitert:

1. Für alle $t \in \mathbf{K}$ ist $\sigma(t) = t$.
2. $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.

Wir benutzen die Schreibweise $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ für eine Substitution. $t\sigma$ bezeichne die Anwendung einer Substitution σ auf einen Term t.

DEFINITION 2.11 (Instanzen). Ein Term t ist eine *Instanz* eines Terms s, wenn es eine Substitution σ mit $t = s\sigma$ gibt.

Zwei Terme s und t heißen *äquivalent*, wenn s eine Instanz von t und t eine Instanz von s ist.

s heißt *strikte* Instanz von t, wenn s eine Instanz von t und t keine Instanz von s ist.

Ist t ein Grundterm und s eine Instanz von t, dann heißt t *Grundinstanz* von s.

BEISPIEL 2.2. $f(x)$ und $f(y)$ sind äquivalent, aber nicht $f(a)$ und $f(y)$. $f(a)$ ist eine strikte Instanz von $f(y)$. Ebenso ist $f(x, x)$ eine strikte Instanz von $f(x, y)$.

DEFINITION 2.12 (Unifikation). Zwei Terme t und s heißen *unifizierbar*, falls es eine Substitution σ mit $t\sigma = s\sigma$ gibt.

In diesem Fall heißt σ *Unifikator (von s und t)*.

Ein Unifikator σ heißt *allgemeinster Unifikator (von s und t)*, wenn für jeden anderen Unifikator τ von s und t eine Substitution μ mit $\sigma \circ \mu = \tau$ existiert.

2.3. Literale, Klauseln und Formeln

DEFINITION 2.13 (Literale). Ein *positives Literal* ist ein Atom. Ein *negatives Literal* ist ein Atom mit vorangestelltem Negationszeichen $\neg$. Ein *Literal* ist ein positives oder negatives Literal. Zu einem Atom A bezeichnet $\overline{A} = \neg A$ bzw. $\overline{\neg A} = A$ das *Komplement* von A bzw. $\neg A$.

Mit $\mathbf{L}$ bezeichnen wir die Menge aller Literale.

DEFINITION 2.14 (Komplementäre Literale). Zwei Literale L und L' heißen *komplementär*, wenn es einen allgemeinsten Unifikator σ mit $L\sigma = \overline{L'}\sigma$ gibt.

Falls σ nicht anderweitig spezifiziert ist, gehen wir bei der Schreibweise $L\sigma = \overline{L'}\sigma$ immer davon aus, daß σ ein allgemeinster Unifikator von L und $\overline{L'}$ ist.

Um gleiche Grundliterale ähnlich zur Lock-Resolution (Boyer, 1971) unterscheiden zu können, fügen wir jedem Literal einen beliebigen Index bei:

DEFINITION 2.15 (Index). Ein *indiziertes* Literal ist ein Paar $L : I$, mit einem Literal $L \in \mathbf{L}$ und einem beliebigem Index $I \in \mathbf{I}$.

DEFINITION 2.16 (Indizierte Klausel). Eine Folge $[L_1 : I_1, \ldots, L_n : I_n]$ von indizierten Literalen heißt *indizierte Klausel*.
$\mathbf{C}$sei die Menge aller (indizierter) Klauseln.

BEISPIEL 2.3. Die beiden indizierten Klauseln $[P(x) : 1, P(a) : 2]$, $[P(a) : 2, P(x) : 1]$ und $[P(x) : 2, P(a) : 1]$ sind verschieden.

Wir gehen im folgenden immer davon aus, daß Klauseln variablendisjunkt sind. An den meisten Stellen wird die übliche Darstellung einer Klausel als Menge von Literalen benutzt. Nur in einigen Ausnahmefällen kommt obige Definition zum Einsatz. Ist dies der Fall, weisen wir explizit darauf hin.

DEFINITION 2.17 (Grundklauseln). Eine Klausel K heißt *Grundklausel*, wenn für jedes $\sigma \in \mathbf{S}$ gilt: $K = K\sigma$.

Grundklauseln sind also alle Klauseln aus $\mathbf{C}$, welche keine Variablen enthalten.

Formel der Prädikatenlogik werden auf übliche Weise mit den Operatoren $\vee$ (Disjunktion), $\wedge$ (Konjunktion), $\neg$ (Negation), $\Rightarrow$ (Implikation) und $\Leftrightarrow$ (Äquivalenz) sowie den Quantoren $\exists$ und $\forall$ zu einer gegebener festen Signatur aus Prädikaten-, Funktions- und Variablensymbolen konstruiert.

Interpretationen werden mit einer Menge von Grundliteralen identifiziert.

KAPITEL 3

Tableaus

> „Grüß Gott“ sagte der Kommandant auf einmal verlegen, wie mir schien. „Füllen sie bitte auf. Super. Und reinigen Sie auch die Scheibe.“ Dann wandte er sich zu mir. „Gehen wir hinein!“
>
> Friedrich Dürrenmatt —
>
> Das Versprechen

Nach einigen Begriffsbestimmungen, legen wir in diesem Kapitel noch einmal die in (Hähnle & Klingenbeck, 1996) vorgestellten Kalküle für A-geordnete Tableaus dar (Abschnitt 3.2). Wir modifizieren diese, so daß noch weitere Suchraumeinschränkungen erzielt werden können (Abschnitt 3.3 und 3.4). In Abschnitt 3.5 diskutieren wir einige Methoden, mit denen Theorien in A-geordneten Tableaus behandelt werden können. Das Kapitel endet mit einem Vergleich mit verwandten Kalkülen, aus dem sich einige noch offene Fragen ergeben (Abschnitt 3.6).

3.1. Vorbemerkungen

Um alle behandelten Kalküle möglichst uniform darstellen zu können, fassen wir ein Tableau immer als Menge von Ästen auf, welche wiederum Mengen logischer Formeln sind (hier: Klauseln). Die einzelnen Tableauregeln können so übersichtlich angegeben werden und es ist noch Raum für weitere Ergänzungen (z.B. Constraints). Bei Beispielen von Tableaus bevorzugen wir die Baumdarstellung. Das heißt, ein Tableau ist ein geordneter Baum, bei dem die einzelnen Knoten mit einer Formel (hier: ein Literal) markiert werden. Wir stellen Abschlüsse in diesem Fall mit Pfeilen von einem Knoten zum anderen dar. Bei Klauseltableaus markieren wir die Wurzel des Baumes mit einem Literal oder mit true.

Beispiel 3.1. Abbildung 3.1 zeigt ein (offenes) Klauseltableau. Die zugehörige Mengenschreibweise ist

$$\{\{A, C\}, \{B, D\}, \{B, E\}\} \ .$$

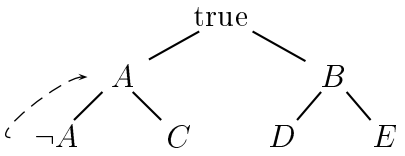


ABBILDUNG 3.1. Beispiel eines Klauseltableaus

DEFINITION 3.1. Ein *Grundklauseltableau* für eine Grundklauselmenge M ist ein Tableau, welches nach den in Tabelle 3.1 gegebenen Regeln konstruiert wurde.

$\dfrac{T \cup \{A\}}{T \cup \{A \cup \{L_1\}\} \cup \ldots \cup \{A \cup \{L_n\}\}}$	für eine Klausel $K = L_1 \vee \ldots \vee L_n$ aus M und einen nicht geschlossenem Ast A.
$\dfrac{T \cup \{A \cup \{L, \overline{L}\}\}}{T - \{A \cup \{L, \overline{L}\}\}}$	für ein Literal L.

TABELLE 3.1. Konstruktionsregeln für Grundklauseltableaus

Dieser Kalkül besitzt noch keine Einschränkung. Ein Ast kann beliebig oft mit einer Klausel aus M erweitert werden. Diese Regel wird *Extension* genannt. Die zweite Regel (*Reduktion*) besagt, daß ein Ast *abgeschlossen* ist, wenn er ein komplementäres Paar enthält. Praktischerweise wird ein Ast, sobald er geschlossen ist, aus dem Tableau entfernt. Ist es möglich, aus dem *leeren Tableau* $\{\{\}\}$, welches einen leeren Ast enthält, mit obigen Regeln die leere Menge abzuleiten, so ist das Tableau zu M *geschlossen*.

Ein Klauseltableau für prädikatenlogische Klauselmengen kann mit jeweils einer der folgenden verschiedenen Änderungen realisiert werden:

- Extensionsschritte werden nur mit Grundinstanzen von Klauseln aus M durchgeführt.
- Es wird Unifikation benutzt, um die Äste abzuschließen.

Um mit der ersten Methode einen vollständigen Kalkül zu erhalten, muß sichergestellt sein, daß eine Beweisprozedur alle möglichen Grundinstanzen von Klauseln aus M aufzählt. Der Vorteil einer solchen Implementierung ist dessen Beweiskonfluenz: ein Reduktions- oder Extensionsschritt braucht nie rückgängig gemacht zu werden.

Da das Herbrand-Universum in diesem Fall recht ziellos aufgezählt wird, benutzt man üblicherweise Unifikation, um Abschlüsse zu finden. Der Unifikator wird meistens sofort auf das ganze Tableau angewendet. Existiert eine Methode universelle Variablen zu erkennen (z.B. Variablen, die in genau einem Literal einer Klausel vorkommen), so brauchen diese bei einer Instantiierung nicht berücksichtigt werden. Um alle möglichen Abschlüsse fair auszuwählen, ist die Astlänge bei fast allen Implementierungen beschränkt. Mit Hilfe von

Backtracking werden dabei alle Abschlußsubstitutionen für das Tableau aufgezählt. Dieses Verfahren ist allerdings nicht mehr beweiskonfluent.

Man kann Klauseltableaus einschränken, wenn Extensionsschritte mit Tautologien oder mit Klauseln verboten werden, die von Literalen des betrachteten Asts subsumiert werden. Dies entspricht in etwa dem Löschen von Tautologien und der Subsumtion von Klauseln im Resolutionskalkül. Sind keine universellen Variablen im Tableau vorhanden, dann handelt es sich bei der Subsumtion um einen Test auf Gleichheit. Obige Einschränkung wird dann üblicherweise als *Regularität* (Letz *et al.*, 1992) bezeichnet.

Die Extensionsschritte können noch weiter eingeschränkt werden, wenn verlangt wird, daß mindestens einer der neu enstandenen Äste abgeschlossen wird (mit Ausnahme des initialen Schrittes). Das heißt, die entsprechende Klausel K und der zu erweiternde Ast A müssen folgende Bedingung erfüllen:

DEFINITION 3.2. Ein Klausel K besitzt eine *schwache Konnektion* in eine Menge von Literalen A, wenn es eine Substitution σ, ein $L \in K$ und ein $L' \in A$ mit $L\sigma = \overline{L'}\sigma$ gibt.

Für eine schwache Konnektion schreiben wir auch $\langle L, L', \sigma \rangle$, wobei σ ein allgemeinster Unifikator von L und $\overline{L'}$ ist.

Tabelle 3.2 zeigt den Kalkül für Klauseltableaus mit Regularität und der schwachen Konnektionsbedingung für prädikatenlogische Klauselmengen M. Dabei gehen wir hier und auch später davon aus, daß bei einer Extension eines Astes mit einer Klausel $K \in M$, die Variablen von K mit neuen, noch nicht vorhandenen Variablen umbenannt werden.

$\dfrac{T \cup \{A\}}{(T \cup \{A \cup \{L_2\}\} \cup \ldots \cup \{A \cup \{L_n\}\})\sigma}$	falls $K = L_1 \vee \ldots \vee L_n \in M$ eine schwache Konnektion $\langle L_1, L, \sigma \rangle$ in den nicht geschlossenem Ast A besitzt und von keinem Literal aus A subsumiert wird.
$\dfrac{T \cup \{A \cup \{L, \overline{L}\}\}}{(T - \{A \cup \{L, \overline{L}\}\})\sigma}$	falls es einen allgemeinsten Unifikator σ mit $L\sigma = \overline{L'}\sigma$ gibt.

TABELLE 3.2. Klauseltableaus mit Regularität und schwacher Konnektion

Wird ein Ast nicht als Menge, sondern als endliche Folge von Literalen betrachtet, so kann man diesen Kalkül noch weiter eingeschränken. Dabei wird bei einer schwachen Konnektion $\langle L, L', \sigma \rangle$ verlangt, daß L' letztes Element des Asts ist (L' ist also Markierung eines Blatts eines nicht geschlossenen Asts des Tableaus). Diese Form der Einschränkung ist als *Konnektionsbedingung* bekannt. Zusammen mit Regularität erhält man *Konnektionstableaus* (Letz, 1993).

Bei beiden Kalkülen geht allerdings die Beweiskonfluenz verloren:

BEISPIEL 3.2. Die Grundklauselmenge $\{P, \neg Q, Q\}$ ist offenbar unerfüllbar. Wird die Konstruktion des Tableaus mit der Klausel P begonnen, so

existiert keine schwache Konnektion von Q und $\neg Q$ im einzigen Ast $\{P\}$. Die Konnektionsbedingung ist dann natürlich erst recht nicht erfüllt.

Die Beweiskonfluenz läßt sich im Fall der schwachen Konnektion erhalten, wenn zugelassen wird, das Tableau mit einer Klausel K zu erweitern, welche eine Konnektion zu einer anderen Klausel besitzt. Solch ein Extensionsschritt sei mit *Neustart* bezeichnet. Allerdings geht dabei die Einschränkung der Konnektionsbedinung fast völlig verloren, da es sehr wahrscheinlich ist, daß eine beliebige Klausel eine schwache Konnektion zu einer anderen hat.

Es ist deswegen sinnvoll, die Menge der Neustartklausel möglichst klein zu halten. Kann die betrachtete (unerfüllbare) Klauselmenge S in zwei disjunkte Teile S' und S'' zerlegt werden, so daß beide Mengen erfüllbar sind, dann reicht es, die Klauseln aus einer dieser Mengen für einen Neustart zu benutzen.

Da es im allgemeinen unentscheidbar ist, solch eine Partitionierung vorzunehmen, zeigen wir im nächsten Abschnitt, wie Ordnungseinschränkungen benutzt werden können, um die Anzahl erlaubter Konnektion einzuschränken.

3.2. A-geordnete Tableaus

3.2.1. A-Ordnungen. A-Ordnungen sind strikte, partielle Ordnungen auf der Menge der Atome mit einer zusätzlichen Eigenschaft (Def. 3.3,3), welche für die Vollständigkeit des Kalküls im prädikatenlogischen Fall notwendig ist.

Definition 3.3. Eine *A-Ordnung* ist eine binäre Relation $\prec_A$ auf der Menge aller Atome, welche folgende Bedingungen erfüllt:

Für alle Atome P, Q und R gilt:

1. $P \not\prec_A P$,
2. wenn $P \prec_A Q$ und $Q \prec_A R$ gilt, dann gilt auch $P \prec_A R$ und
3. für alle Substitutionen σ und Atome mit $P \prec_A Q$ gilt $P\sigma \prec_A Q\sigma$.

Die letzte Bedingung wird *Substitutivität* genannt.

Ein Literal L einer Klausel K ist *$\prec_A$-maximal* in K, wenn es kein Literal in K gibt, welches bezüglich $\prec_A$ größer als L ist.

Da die Ordnung nicht total zu sein braucht, kann es unvergleichbare Atome geben, so daß in einer Klausel mehrere maximale Literale vorkommen können. Für die triviale A-Ordnung $\prec_\epsilon$, welche überhaupt keine Atome vergleicht, sind zum Beispiel alle Literale einer Klausel maximal.

Mit Hilfe von A-Ordnungen kann die Anzahl gültiger Konnektionen zwischen zwei Klauseln eingeschränkt werden.

Definition 3.4. Eine Klausel K hat eine *$\prec_A$-maximale Konnektion* zu einer Klausel K', wenn es eine Konnektion $\langle L, L', \sigma\rangle$ zwischen K und K' gibt, so daß L $\prec_A$-maximal in K und L' $\prec_A$-maximal in K' ist.

K hat eine *$\prec_A$-maximale Konnektion* in eine Menge von Literalen A, wenn es eine schwache Konnektion $\langle L, L', \sigma\rangle$ gibt und L $\prec_A$-maximal in K ist.[1]

[1]Genauer müßten wir hier von schwacher maximaler Konnektion sprechen. Um nicht endlos lange Wortfolgen zu produzieren, sei uns dieser Lapsus erlaubt.

BEISPIEL 3.3. $\prec_A$ sei eine A-Ordnung mit $P \prec_A Q$. Dann haben $\neg Q$ und $P \vee Q$ eine $\prec_A$-maximale Konnektion; aber nicht $\neg P$ und $P \vee Q$.

Wenn klar ist, welche A-Ordnung gemeint ist, sprechen wir statt von $\prec_A$-maximal auch nur von maximal. Gibt es keinen Bezug zu einer speziellen A-Ordnung, so sind A-Ordnungen im Allgemeinen gemeint.

Bevor wir die Regeln für A-geordnete Tableaus angeben, stellen wir noch zwei nicht triviale A-Ordnungen vor. Weitere A-Ordnungen lernen wir noch in späteren Kapiteln und Abschnitten kennen.

DEFINITION 3.5. Die *Termtiefenordnung* ist eine binäre Relation $\prec_\tau$ auf der Menge der Atome. Für Atome P und Q gilt $P \prec_\tau Q$ genau dann, wenn

1. $\tau(P) \prec \tau(Q)$ gilt und
2. für alle $x \in \mathbf{V}$ ist $\tau(x, Q) \prec \tau(x, R)$.

$\prec_\tau$ ist offenbar nicht total, da $P(a) \not\prec_\tau Q(b)$ gilt. Diese A-Ordnung spielt zwar für theoretische Überlegungen bei $\prec_\tau$-geordneter Resolution eine Rolle, ist für den praktischen Einsatz im Tableaukalkül aber weniger geeignet,da $\prec_\tau$ keinerlei Freiheitsgrade besitzt, die ausgenutzt werden können, um $\prec_\tau$ an ein gegebenes Problem anzupassen.

DEFINITION 3.6. Es sei $<$ eine partielle Ordnung auf $\mathbf{F}$.

Für die Ordnung $\prec_L$ gelte $f(t_1, \ldots, t_n) \prec_L f'(t'_1, \ldots, t'_{n'})$ ($n = n' = 0$ möglich) genau dann, wenn

1. $f < f'$ oder
2. $f = f', n = n'$ und $t_i \prec_L t'_i$ für alle $1 \leq t \leq n$ ist.

Falls $<$ auf die Menge $\mathbf{P} \cup \mathbf{F}$ ausgedehnt wird, so kann $\prec_L$ auf offensichtliche Weise auch auf die Menge aller Atome erweitert werden.

Indem die partielle Grundordnung $<$ modifiziert wird, kann die Ordnung $\prec_A$ an ein gegebenes Problem angepaßt werden. Man kann dies ausnutzen, um die Anzahl maximaler Atome in einer Klausel zu minimieren.

3.2.2. A-geordnete Klauseltableaus. A-geordnete Klauseltableaus, so wie sie in (Hähnle & Klingenbeck, 1996) vorgestellt werden, entsprechen Klauseltableaus mit schwacher Konnektion und Regularität sowie Neustart. A-Ordnungen werden benutzt, um die Anzahl schwacher Konnektionen einzuschränken, so daß weniger Neustarts möglich sind.

Tabelle 3.3 zeigt die Regeln für den prädikatenlogischen Kalkül. Wir erweitern dabei die Regeln um eine zusätzliche Komponente, mit deren Hilfe wir garantieren, daß jede Grundsubstitution des Tableaus noch A-geordnet ist. In einem (nicht aufgeführten) initialen Schritt sind Tableau und die zusätzliche Komponente leer.

SATZ 3.1. (Hähnle & Klingenbeck, 1996) *M sei eine unerfüllbare Klauselmenge und $\prec_A$ eine A-Ordnung. Dann gibt es ein geschlossenes $\prec_A$-geordnetes Klauseltableau zu M.*

BEWEIS. Diese Ausage ist eine Folgerung aus Satz 3.4. Siehe auch die Bemerkung 3.1. □

$$\frac{T \cup \{A\}, C}{(T \cup \{A \cup \{L_2\}\} \cup \ldots \cup \{A \cup \{L_n\}\})\sigma, C \cup \langle L_1, K\rangle\sigma}$$

falls $K = L_1 \vee \ldots \vee L_n \in M$ eine maximale Konnektion $\langle L_1, L, \sigma\rangle$ in dem nicht geschlossenen Ast A oder in M besitzt und von keinem Literal aus A subsumiert wird.

$$\frac{T \cup \{A \cup \{L, \overline{L}\}\}, C}{(T - \{A \cup \{L, \overline{L}\}\})\sigma, C\sigma}$$

falls es einen allgemeinsten Unifikator σ mit $L\sigma = \overline{L'}\sigma$ gibt und für alle $\langle L'', K\rangle \in C\sigma$ gilt: L'' ist maximal in K.

TABELLE 3.3. A-geordnete Klauseltableaus

Klauseltableaus mit schwacher Konnektion und Regularität sind auch ohne Neustart noch vollständig, wenn im initialen Schritt eine relevante Klausel ausgewählt wird. Dies gilt für A-geordnete Tableaus nicht mehr:

BEISPIEL 3.4. Betrachte die unerfüllbare Klauselmenge

$$M = \{A \vee \underline{B}, A \vee \underline{\neg B}, \neg A \vee \underline{C}, \neg A \vee \underline{\neg C}\}$$

mit der A-Ordnung $A < B < C$ (maximale Literale sind unterstrichen).

Die Literale der ersten beiden Klauseln, besitzen keine maximale Konnektion zu den letzten beiden Klauseln. Falls kein Neustart zugelassen wird, ist ein A-geordnetes Tableau für M nach zwei Expansionsschritten erschöpft.

Ohne Ordnungseinschränkung kann ein geschlossenes Tableau konstruiert werden, *ohne* daß ein Neustart notwendig ist, wenn mit einer beliebigen Klausel von M angefangen wird (jede Klausel von M ist relevant).

3.2.3. A-geordnete Negations-Normal-Form Tableaus. In diesem Abschnitt werden wir den in (Hähnle & Klingenbeck, 1996) vorgestellten Kalkül A-geordneter Tableaus für Formeln in Negations-Normal-Form (NNF) rekapitulieren. Dieser erweist sich analog zu A-geordneten Klauseltableaus als Spezialfall eines allgemeineren Konzepts.

Expansionen von Ästen in A-geordnete Tableaus für NNF werden ähnlich zu A-geordneten Klauseltableaus, mit Hilfe maximaler Konnektionen eingeschränkt. Dazu ist es allerdings notwendig zu beschreiben, wann zwei Formeln in NNF eine (maximale) Konnektion besitzen. Die folgenden Definitionen aus (Hähnle & Klingenbeck, 1996) klären diesen Sachverhalt.

DEFINITION 3.7. Für eine NNF-Formel gilt:

1. Ein Literal ist eine NNF-Formel.
2. Sind $\phi_1, \ldots, \phi_n (n \geq 2)$ NNF-Formeln, dann ist $\phi_1 \wedge \ldots \wedge \phi_n$ eine NNF-Formel. Jedes Paar ϕ_i, $\phi_j (i \neq j)$ heißt *konjunkt* und die Formel heißt *konjunktiv*.
3. Sind $\phi_1, \ldots, \phi_n (n \geq 2)$ NNF-Formeln, dann ist $\phi_1 \vee \ldots \vee \phi_n$ eine NNF-Formel. Jedes Paar ϕ_i, $\phi_j (i \neq j)$ heißt *disjunkt* und die Formel heißt *disjunktiv*.

4. Ist ϕ eine NNF-Formel und $x \in \mathbf{V}$ kommt in ϕ nicht gebunden vor, dann ist $(\forall x)\phi$ eine NNF-Formel.

Man erhält eine NNF einer Formel ϕ, indem diese zuerst in Pränexform gebracht und die Existenzquantoren durch Skolemisierung eliminiert werden. Anschließend werden die Negationsoperatoren so weit wie möglich nach „innen“ verschoben. Es sind aber auch andere Transformationen denkbar, bei denen die Allquantoren nicht völlig nach außen geschoben werden müssen.

DEFINITION 3.8. Zwei Teilformeln A, B einer NNF-Formel ϕ heißen *k-verbunden (d-verbunden)*, wenn es Teilformeln F, G in ϕ gibt, so daß A eine Teilformel von F, B eine Teilformel von G ist und F, G sind konjunkt (disjunkt) in ϕ.

Ein *k-Pfad (d-Pfad)* durch ϕ ist eine maximale Menge paarweise k-verbundener (d-verbundener) Literale in ϕ.

Mengen von Formeln werden als konjunkt angenommen.

DEFINITION 3.9. Φ sei eine Menge von Formeln in NNF, τ und τ' seien Variablenumbenennungen aller gebundenen Variablen von Φ. Eine *Konnektion* in Φ ist ein Paar (L, L') von k-verbundenen Literalen, die in Φ vorkommen, so daß $\{L\tau, \overline{L'}\tau'\}$ unifizierbar ist.

Ein Unifikator von $\{L\tau, \overline{L}'\tau'\}$ heißt *Konnektions-Unifikator* von (L, L').

ϕ sei eine Formel. *ϕ enthält eine Konnektion in* Φ, wenn es eine Konnektion (L, L') in $\Phi \cup \{\phi\}$ gibt, so daß L in ϕ vorkommt und L' in einer Formel von Φ.

DEFINITION 3.10. Φ sei eine NNF-Formel oder eine Menge von Formeln in NNF und $\prec_A$ eine A-Ordnung auf der Menge der Literale von Φ.

Ein Literal L ist *maximal* in Φ, wenn L maximal in einem d-Pfad durch Φ vorkommt.

Eine *maximale Konnektion* in Φ ist eine Konnektion (L, L') in Φ, so daß L und L' maximal in Φ sind.

Diese Definition einer maximalen Konnektion stimmt im Fall von Klauseln mit Definition 3.4 überein. Dies rechtfertigt es, für beide Sachverhalte die gleichen Begriffe zu verwenden.

Die Regeln des Kalküls sind in Tabelle 3.4 aufgeführt.

Es handelt sich hier um einen vollständigen Kalkül für die Prädikatenlogik erster Stufe, bei dem keine Unifikation benutzt wird, um Abschlüsse zu suchen, sondern immer sofort Grundterme für alle universell quantifizierten Variablen einer Formeln eingesetzt werden. Dadurch vereinfacht sich der in (Hähnle & Klingenbeck, 1996) gegebene Vollständigkeitsbeweis.

SATZ 3.2. *Ist Φ eine unerfüllbare Formelmenge in NNF und $\prec_A$ eine A-Ordnung, dann gibt es ein geschlossenes $\prec_A$-geordnetes Tableau für Φ.*

BEWEIS. Diese Ausage ist eine direkte Folgerung aus Satz 3.5. □

$\dfrac{}{\{\Phi\}}$	für eine Formelmenge Φ in NNF.
$\dfrac{T \cup \{A \cup \{L, \overline{L}\}\}}{T - \{A \cup \{L, \overline{L}\}\}}$	für ein Literal L.
$\dfrac{T \cup \{A \cup \{\phi\}\}}{T \cup \{A \cup \{\phi_1, \ldots, \phi_n\}\}}$	falls $\phi = \phi_1 \wedge \ldots \wedge \phi_n$ und ϕ eine maximale Konnektion in A besitzt.
$\dfrac{T \cup \{A \cup \{\phi\}\}}{T \cup \{A \cup \{\phi_1\}\} \cup \ldots \cup \{A \cup \{\phi_n\}\}}$	falls $\phi = \phi_1 \vee \ldots \vee \phi_n$ und ϕ eine maximale Konnektion in A besitzt.
$\dfrac{T \cup \{A \cup \{\forall x \phi(x)\}\}}{T \cup \{A \cup \{\forall x \phi(x), \phi(t)\}\}}$	wobei t ein beliebiger Grundterm ist, $\phi(t) \notin A$ und $\phi(t)$ eine maximale Konnektion in A besitzt.

TABELLE 3.4. A-geordnete Tableaus für Formeln in NNF

Von der zusätzlichen Regularitätseinschränkung der Klauseltableaus wird hier kein Gebrauch gemacht. Es sei aber darauf hingewiesen, daß die in (Hähnle & Klingenbeck, 1996) benutzte Regularität für A-geordnete Tableaus in NNF auch für die hier vorgestellten Kalküle ausgenutzt werden kann.

3.3. Tableaus mit Auswahlfunktion

Wie eingangs angekündigt, können die voranstehenden A-geordneten Kalküle mit einer geringer Änderung noch weiter eingeschränkt werden. Dabei wird die A-Ordnung durch eine Auswahlfunktion ersetzt, welche aus jeder Grundklausel genau ein Literal auswählt.

Bei einigen anderen Modifikationen ist noch nicht geklärt, ob der Kalkül vollständig ist (siehe Abschnitt 3.6).

DEFINITION 3.11. Eine *Auswahlfunktion* f für eine Menge von Grundklauseln M ist eine totale Funktion auf M, die aus jeder Klausel von M genau ein Literal auswählt.

Zum Beispiel definieren totale A-Ordnungen im Grundfall eine Auswahlfunktion, falls eine Klausel als Menge von Literalen aufgefaßt wird. Der Begriff einer *Konnektion mit Auswahlfunktion* f wird analog zur Definition 3.4 einer maximalen Konnektion gebildet.

3.3.1. Klauseltableaus mit Auswahlfunktion. Der Kalkül für Klauseltableaus mit Auswahlfunktion stellt sich genau so dar, wie A-geordnete Klauseltableaus (siehe Tabelle 3.3), mit der einzigen Änderung, daß anstatt maximaler Konnektionen jetzt Konnektionen mit Auswahlfunktion betrachtet werden.

Zusätzlich wollen wir in diesem Abschnitt Klauseln nicht mehr als Mengen von Literalen, sondern als Folge indizierter Literale auffassen (Def. 2.16). Man mache sich diesen Umstand bei den folgenden Beweisen klar.

Wie üblich, wird zuerst die Grundvollständigkeit des Kalküls gezeigt. Diese wird benutzt, um ein geschlossenes Grundtableau T zu einer unerfüllbaren prädikatenlogischen Klauselmenge M zu konstruieren, welches zu einem geschlossenen Tableau zu M gehoben werden kann. Dazu benötigen wir allerdings noch folgenden Begriff:

DEFINITION 3.12. Ein Klauseltableau T zu einer Grundklauselmenge M heißt *erschöpft*, wenn sich auf T keine Tableauregeln mehr anwenden lassen.

SATZ 3.3. *Ist T ein erschöpftes Klauseltableau mit Auswahlfunktion f zu einer endlichen Grundklauselmenge M und $A \in T$ ein offener Ast. Dann besitzt M ein Modell.*

BEWEIS. A ist endlich, da M endlich ist und identische Klauseln nicht mehrfach benutzt werden dürfen, um einen Ast zu expandieren (Regularität). Da A nicht geschlossen ist, also keine komplementären Literale enthält, bilden die Literale von A eine partielle Interpretation I.

M' sei die Menge aller Klauseln aus M, die nicht von I erfüllt werden. Dann gelten für jede Klausel K aus M' folgende Bedingungen:

1. Es gibt in A kein zu $f(K)$ komplementäres Literal, da sonst K Konnektion mit Auswahlfunktion in A besitzt und die Klausel K den Ast expandieren könnte.
2. Es gibt keine von K verschiedene Klausel $K' \in M'$, welche Konnektion mit Auswahlfunktion zu K besitzt. Ansonsten wären K und K' für einen Neustart zulässig.

J sei die Menge aller Literale, die von f in allen Klauseln von M' ausgewählt werden. J ist eine partielle Interpretation: Gemäß Bedingung 2 ist J wohldefiniert und kann mit I wegen Punkt 1 zu einer Interpretation aller Klauseln aus M erweitert werden. □

Satz 3.3 stellt sicher, daß es zu jeder unerfüllbaren endlichen Klauselmenge ein geschlossenes Klauseltableau mit Auswahlfunktion gibt. Für die Vollständigkeit des Kalküls im prädikatenlogischen Fall, muß die Auswahlfunktion noch folgende Eigenschaft besitzen:

DEFINITION 3.13. Eine *hebbare* Auswahlfunktion f für eine Klauselmenge M ist eine Auswahlfunktion für M, so daß für jede Substitution σ und Klausel K aus M gilt:
Falls $f(K\sigma) = L\sigma$ gilt, dann gilt auch $f(K) = L$ für $L \in K$.[2]

Diese Eigenschaft enspricht der Substitutivität von A-Ordnungen.

[2]Man beachte, daß aus einer prädikatenlogischen Klausel auch mehr als ein Literal ausgewählt werden kann. Wir behalten in diesem Fall trotzdem die nachlässige Schreibweise $f(K) = L$ bei.

SATZ 3.4. *f sei eine hebbare Auswahlfunktion zu einer unerfüllbaren Klauselmenge M. Dann gibt es ein geschlossenes Klauseltableau mit Auswahlfunktion f für M.*

BEWEIS. Nach dem Satz von Herbrand gibt es eine endliche unerfüllbare Grundklauselmenge M' von Grundinstanzen von M. Zu dieser Menge existiert ein Grundklauseltableau mit Auswahlfunktion f (Satz 3.3), welches zu einem geschlossenen Tableau mit Auswahlfunktion f zu M gehoben werden kann. □

Die starke Einschränkung, die eine Auswahlfunktion definiert, kann wieder teilweise aufgegeben werden, wenn mehr als nur ein Literal aus einer Klausel ausgewählt wird. Der entsprechende Kalkül bleibt mit dieser Änderung selbstverständlich weiterhin vollständig.

DEFINITION 3.14. Eine *hebbare Auswahl* ist eine totale Funktion $g : \mathbf{C} \to \mathbf{L}$ mit $g(K) \subseteq K$ und aus $K\sigma \in g(K\sigma)$ folgt $K \in g(K)$.

BEISPIEL 3.5.
1. Ist $\prec_A$ eine A-Ordnung, so definiert sie eine hebbare Auswahl auf $\mathbf{C}$, indem alle $\prec_A$-maximalen Literale aus einer Klausel ausgewählt werden.
2. Die Funktion, die aus einer Klausel alle positiven Literale auswählt oder, falls solche nicht vorhanden sind, alle Literale auswählt ist eine hebbare Auswahl.
3. Eine hebbare *Auswahlfunktion* ist eine hebbare Auswahl.

Da A-geordnete Tableaus, Tableaus mit Auswahlfunktion oder Tableau mit Auswahl letztlich ein und das selbe Schema zugrunde liegt, unterscheiden wir im folgenden nicht immer zwischen diesen Kalkülen. Nur wenn es explizit auf die besonderen Eigenschaften eines dieser Begriffe ankommt, nehmen wir genauer darauf Bezug.

BEMERKUNG 3.1. Da A-Ordnungen eine Auswahl auf Klauseln definieren, sind insbesondere A-geordnete Tableaus vollständig. Offenbar können statt A-Ordnungen auch *L-Ordnungen* benutzt werden. Dies sind strikte, hebbare Ordnungen auf der Menge der Literale (nicht nur Atome).

Bei eingehender Betrachtung des Vollständigkeitsbeweises für prädikatenlogische Klauseltableaus mit Auswahlfunktion und der zugehörigen Beweisprozedur in Tabelle 3.3 fällt auf, daß unter Umständen etwas von der Ordnungseinschränkung aufgegeben wird. So sind die beiden Atome $P(x)$ und $Q(y)$ bzgl. der Termtiefenordnung maximal in $K = P(x) \vee Q(y)$. K kann deswegen einen offenen Ast A eines Tableaus expandieren, wenn K z.B. eine maximale Konnektion $\langle P(x), \neg P(a), \{x \leftarrow y\}\rangle$ in A besitzt.

Da sich die Maximalität von $P(x)$ in K aufgrund einer späteren Substitution ändern kann, ist ein zugehöriges Grundtableau evtl. nicht mehr A-geordnet. Für diesen Fall bieten sich Ordnungsconstraints an, die — solange sie erfüllbar sind — sicherstellen, daß es eine Grundsubstitution des Tableaus gibt, so daß dieses noch A-geordnet ist. Im obigen Fall würde bei Extension mit K ein

α	α_1	α_2
$\phi \wedge \psi$	ϕ	ψ
$\neg(\phi \vee \psi)$	$\neg\phi$	$\neg\psi$
$\neg(\phi \Rightarrow \psi)$	ϕ	$\neg\psi$
$\neg\neg\phi$	ϕ	ϕ

β	β_1	β_2
$\phi \vee \psi$	ϕ	ψ
$\neg(\phi \wedge \psi)$	$\neg\phi$	$\neg\psi$
$\phi \Rightarrow \psi$	$\neg\phi$	ψ

δ	$\delta(t)$
$\neg(\forall x)(\phi(x))$	$\neg\phi(t)$
$(\exists x)(\phi(x))$	$\phi(t)$

wobei $t = f_{[\delta]}(x_1, \ldots, x_n)$ und $f_{[\delta]}$ (Skolem-)Funktion ist, die nur von δ abhängt und $x_1, \ldots, x_n$ sind alle freien Variablen, die in δ vorkommen.

γ	$\gamma(y)$
$(\forall x)(\phi(x))$	$\neg\phi(y)$
$\neg(\exists x)(\phi(x))$	$\phi(y)$

wobei y eine neue freie Variable ist.

TABELLE 3.5. Formeltypen

Constraint $P(x) \not<_\tau P(y)$ erzeugt werden. Ob dieses Verfahren sinnvoll ist und was für Probleme dabei auftreten, besprechen wir detaillierter in Kapitel 4.

Im folgenden wollen wir uns weiterhin mit dem Fall beschäftigen, daß nichts über die gegebene Beweisprozedur hinausgehende unternommen wird, um ein geordnetes Grundtableau sicherzustellen.

3.4. Tableaus mit Auswahl für beliebige Formeln

In Abschnitt 3.2.3 haben wir gesehen, wie A-geordnete Klauseltableaus für Formeln in NNF verallgemeinert werden können (ohne den Beweis aus (Hähnle & Klingenbeck, 1996) zu wiederholen). Dies läßt sich analog auch für Tableaus mit Auswahlfunktion durchführen. Wir betrachten im folgenden den noch allgemeineren Fall, bei dem prädikatenlogische Formeln mit den gebräuchlichen zweiwertigen Quantoren ($\vee, \wedge, \neg, \Rightarrow$) und Operatoren ($\forall, \exists$) aufgebaut sind.

Formeln werden wie üblich mit den Typen α, β, γ und δ klassifiziert (Fitting, 1996) (siehe Tabelle 3.5). Die verwendete δ-Regel ist von der Fittingschen verschieden: Anstatt immer neue Skolemfunktionen einzuführen, reicht es aus, Skolemfunktionen zu benutzten, die allein von δ abhängen (Beckert *et al.*, 1993).

Diese δ-Regel — δ^{++}-Regel genannt — ist für die Definition der disjunktiven Pfade einer Formel ϕ wichtig: Im Fall von NNF-Formeln unterscheiden sich disjunktive Pfade von ϕ nur bezüglich Variablenumbenennungen, da Existenzquantoren durch Skolemisierung entfernt wurden. Wird statt der δ^{++}-Regel im Tableau eine andere δ-Regel benutzt, bei der nicht sichergestellt ist, daß für zwei (bis auf Variablenumbenennung) identische Formeln auch die selben Skolemfunktionen benutzt werden, so unterscheiden sich die entsprechenden disjunktiven Pfade. In diesem Fall ist es nicht möglich Konnektionen in einem Vorverarbeitungsschritt zu berechnen. Deswegen ist die δ^{++}-Regel notwendig für eine effiziente Implementierung.

DEFINITION 3.15. Die Menge D_ϕ aller *disjunktiven Pfade* einer prädikatenlogischen Formel ϕ wird rekursiv wie folgt berechnet:

- $D_\phi := \{\{\phi\}\}$, falls ϕ ein Literal ist.
- $D_\phi := D_{\alpha_1} \cup D_{\alpha_2}$, falls ϕ vom Typ α ist.
- $D_\phi := \{B_1 \cup B_2 | B_i \in D_{\beta_i}\}$, falls ϕ vom Typ β ist.
- $D_\phi := D_{\delta(t)}$, falls ϕ vom Typ δ ist.
- $D_\phi := D_{\gamma(x)}$, falls ϕ vom Typ γ ist.

Nach obiger Diskussion ist folgende Bemerkung leicht einzusehen:

BEMERKUNG 3.2. Sind ϕ und ψ bis auf Variablenumbennung identisch, so sind dies auch D_ϕ und D_ψ.

Aufgrund dieser Aussage, sprechen wir von *der* Menge disjunktiver Pfade einer Formel. Ein Pfad aus D_ϕ wird wie eine Klausel behandelt, weswegen sich der Begriff Konnektion mit Auswahlfunktion für zwei Formeln analog zu Klauseltableaus mit Auswahlfunktion definieren läßt.

DEFINITION 3.16. Eine prädikatenlogische Formel ϕ besitzt eine *Konnektion mit Auswahlfunktion* f zu einer Formel ψ, wenn es einen disjunktiven Pfad $d_\phi \in D_\phi$ und $d_\psi \in D_\psi$ gibt, so daß d_ϕ eine Konnektion mit Auswahlfunktion f zu d_ψ besitzt.

Es ist klar, daß bei NNF-Formeln dieser Begriff einer Konnektion mit der von Def. 3.9 identisch ist.

Wenn $\phi = \psi$ gilt, dann entspricht dies einem Neustart bei Klauseltableaus. Es können auch wieder die Begriffe *A-geordnete Konnektion* und *Konnektion mit Auswahl* gebildet werden.

Tabelle 3.6 zeigt die Tableauregeln für Tableaus mit Auswahl. Diese sind eine direkte Verallgemeinerung von A-geordneten Tableaus für Formeln in NNF (Hähnle & Klingenbeck, 1996).

3.4.1. Hintikkamenge für Tableaus mit Auswahlfunktion. Für den Vollständigkeitsbeweis benutzen wir eine Standardbeweistechnik, die auf Hintikkamengen basiert.

DEFINITION 3.17. H und M seien Mengen prädikatenlogischer Formeln und f eine Auswahlfunktion auf Formeln von M. H heißt *Hintikkamenge mit Auswahlfunktion f für M*, wenn für jede Formel $\phi \in H$ mit Konnektion mit Auswahlfunktion f in H gilt:

1. Ist Φ vom Typ α, dann gilt $\{\alpha_1, \alpha_2\} \subset H$.
2. Ist Φ vom Typ β, dann ist $\beta_1 \in H$ oder $\beta_2 \in H$.
3. Ist Φ vom Typ δ, dann ist $\delta(t) \in H$ für einen variablenfreien Term t.
4. Ist Φ vom Typ γ, dann ist $\gamma(t) \in H$ für jeden variablenfreien Term t.
5. H enthält keine komplementären Literale.
6. Nicht anderes ist in H.

LEMMA 3.1 (Hintikkas Lemma).
Jede Hintikkamenge H zu einer Auswahlfunktion f besitzt ein Modell.

Init

$$\frac{}{\{\Phi\}}$$ zu einer gegebenen Formelmenge Φ.

Abschluß

$$\frac{T \cup \{A \cup \{L, L'\}\}}{\{T - (A \cup \{L, L'\})\}\sigma}$$ falls $L\sigma = \overline{L'}\sigma$ für einen allgemeinsten Unifikator σ ist.

α-Expansion

$$\frac{T \cup \{A \cup \{\alpha\}\}}{\{T - \{(A \cup \{\alpha_1, \alpha_2\}) - \{\alpha\}\}}$$ wobei α eine Konnektion mit Auswahlfunktion in A besitzt.

β-Expansion

$$\frac{T \cup \{A \cup \{\beta\}\}}{\{T \cup \{(A \cup \{\beta_1\}) - \{\beta\}, (A \cup \{\beta_2\}) - \{\beta\}\}}$$ wobei β eine Konnektion mit Auswahlfunktion in A besitzt.

δ-Expansion

$$\frac{T \cup \{A \cup \{\delta\}\}}{\{T \cup \{(A \cup \{\delta(t)\}) - \{\delta\}\}}$$ wobei δ eine Konnektion mit Auswahlfunktion in A besitzt.

γ-Expansion

$$\frac{T \cup \{A \cup \{\gamma\}\}}{\{T \cup \{A \cup \{\gamma, \gamma(y)\}\}}$$ wobei γ eine Konnektion mit Auswahlfunktion in A besitzt.

TABELLE 3.6. Tableau mit Auswahlfunktion

Für den Beweis von Hintikkas Lemma benötigen wir eine zu Proposition 1 in (Hähnle & Klingenbeck, 1996) analoge Aussage. Da wir nur eine Richtung der Aussage brauchen, verzichten wir auf den Beweis der anderen.

LEMMA 3.2. *ϕ sei eine prädikatenlogische Formel und I eine Interpretation.*

Erfüllt I jede Grundinstanz aller disjunktiven Pfade aus D_ϕ, so erfüllt I auch jede Grundinstanz von ϕ (Die Umkehrung gilt auch).

BEWEIS. Wir benutzen strukturelle Induktion über den Formelaufbau von ϕ.

Induktionsanfang:
Ist ϕ ein Literal, so gilt $D_\phi = \{\{\phi\}\}$. Offenbar erfüllt I jede Grundinstanz von ϕ.

Induktionsschritt:

- Ist ϕ vom Typ α, so gilt $D_\phi = D_{\alpha_1} \cup D_{\alpha_2}$. I erfülle jede Grundinstanz von Pfaden von D_ϕ. Dann erfüllt I auch jede Grundinstanz von Pfaden von D_{α_1} und D_{α_2}. Nach IV erfüllt I also jede Grundinstanz von α_1 und α_2 und damit auch jede Grundinstanz von ϕ.
- ϕ sei vom Typ β. I erfülle jede Grundinstanz σ von Pfaden in D_ϕ. Das heißt, I erfüllt alle Pfade in $D_{\beta_1}\sigma$ oder $D_{\beta_2}\sigma$. Nach IV erfüllt I dann $\beta_1\sigma$ oder $\beta_2\sigma$ und damit auch $\phi\sigma$.
- ϕ sei vom Typ δ. Dann ist $D_\phi = D_\delta(t)$ für einen Term t, welcher nur von ϕ abhängt. Erfüllt I jede Instanz eines disjunktiven Pfad von δ, dann jede Instanz eines Pfades von $D_{\delta(t)}$. Nach IV erfüllt I jede Instanz von $\delta(t)$ und damit auch jede Instanz von ϕ.

- Ist ϕ vom Typ γ. Dann gilt $D_\phi = D_{\gamma(x)}$ für eine neue freie Variable x. Erfüllt I jede Grundinstanz eines Pfades von D_ϕ, dann auch jeden Grundinstanz eines Pfades von $D_{\gamma(x)}$. Nach IV erfüllt I jede Grundinstanz von $\gamma(x)$, d.h. I erfüllt für jeden Term t mit Variablen, die nur in ϕ auftreten, jede Grundsubstitution von $\gamma(t)$ und deswegen auch jede Grundsubstitution von ϕ.

□

Nun zum eigentlichen Beweis von Hintikkas Lemma.

Beweis. Wir geben eine Interpretation an und zeigen mit vollständiger Induktion über die Formelstruktur, daß diese Interpretation alle Formeln von H erfüllt.

Die Literale von H definieren eine partielle Interpretation I mit Hilfe von $I = \{L \in H : L \text{ ist Literal}\}$. Nach Teil 5 der Definition einer Hintikkamenge mit Auswahlfunktion enthält H keine komplementären Literale, d.h. I ist wohldefiniert.

Betrachte die Formelmenge $U =$

$$\begin{array}{ll} \{\phi \mid & \phi \text{ ist vom Typ } \alpha \text{ und } \alpha_1 \notin H \text{ oder } \alpha_2 \notin H, \\ & \phi \text{ ist vom Typ } \beta \text{ und } \beta_1 \notin H \text{ und } \beta_2 \notin H\} \cup \\ \{\delta(t) \mid & \phi \text{ ist vom Typ } \delta \text{ und } \delta(t) \notin H \text{ für jeden variablenfreien Term } t\} \cup \\ \{\gamma(t) \mid & \phi \text{ ist vom Typ } \gamma \text{ und } \gamma(t) \notin H \text{ für einen variablenfreien Term } t\} \ . \end{array}$$

D_U^g sei die Menge aller Grundinstanzen von $\bigcup_{\phi \in U} D_\phi$ und J sei die Menge aller Literale, die aus disjunktiven Pfaden von D_U^g ausgewählt werden. Dann ist J eine partielle Interpretation:

Falls nicht, so existiert ein komplementäres Paar $L, L' \in J$, wobei L bzw. L' aus einer Grundinstanz eines disjunktiven Pfads von $\phi \in U$ bzw. $\psi \in U$ ausgewählt werden. Da die Auswahlfunktion f hebbar ist, werden auch die zu L und L' entsprechenden Literale der disjunktiven Pfade von ϕ und ψ ausgewählt. ϕ und ψ besitzen also eine Konnektion mit Auswahlfunktion im Widerspruch zur Definition von U.

Analog läßt sich zeigen, daß $I \cup J$ eine partielle Interpretation der Formeln von H ist.

Nach Lemma 3.2 wird jedes $\phi \in U$ von J erfüllt, da J nach Definition jede Grundinstanz aller disjunktiven Pfade von ϕ erfüllt.

Mit struktureller Induktion über den Formelaufbau von Formeln $\phi \in H$ zeigen wir, daß $I \cup J$ eine Interpretation von H ist.

Induktionsanfang

$\phi \in H$ ist ein Grundliteral und wird schon von I erfüllt.

Induktionsschritt

- ϕ sei vom Typ α. Gilt $\phi \in U$, so wird ϕ bereits von J erfüllt. Nach Induktionsvoraussetzung (IV) werden α_1 und α_2 von $I \cup J$ erfüllt und damit auch ϕ.

- Ist ϕ vom Typ β, so erfüllt $I \cup J$ ϕ analog zum obigen Fall.
- ϕ sei vom Typ δ und $\phi \notin U$ (weil sonst ϕ schon von J erfüllt ist), dann erfüllt nach IV $I \cup J$ $\delta(t)$ für einen Term t und damit auch ϕ.
- Ist ϕ vom Typ γ, so ist für jeden Term t $\gamma(t)$ entweder in U oder in H enthalten. Also wird $\gamma(t)$ von J oder nach IV von $I \cup J$ erfüllt.

□

Bevor wir auf die Vollständigkeit von Tableaus mit Auswahlfunktion für beliebige prädikatenlogische Formeln eingehen noch einige Anmerkungen, die den Vollständigkeitsbeweis vereinfachen.

Wir gehen davon aus, daß es eine faire Strategie gibt, die Tableauregeln anzuwenden. Ohne näher auf den Begriff der Fairneß einzugehen (einiges mehr findet sich dazu in (Fitting, 1996, Seiten 191–196)), seien hier die Folgen einer solchen Strategie erwähnt.

Zum einen garantiert die Fairness, daß es in einem nicht geschlossenen erschöpften Tableau einen offenen (möglicherweise unendlichen) Ast A gibt, der durch faire Auswahl der Tableauregeln stetig erweitert wurde.

Desweiteren muß die Fairness gewährleisten, daß alle möglichen Terme des Herbrand-Universums in A aufgezählt werden, so daß für eine γ-Formel aus A auch $\gamma(t)$ für jeden variablenfreien Term t in A ist.

Da die Variablen eines erschöpften Tableaus mit einem beliebigen Grundterm instantiiert werden können, beschränken wir uns auf einen Ast A, in dem keine freien Variablen vorkommen.

Satz 3.5. *Ist ϕ eine unerfüllbare prädikatenlogische Formel und f eine Auswahlfunktion auf ϕ. Dann gibt es — eine faire Auswahlstrategie der Tableauregeln vorausgesetzt — ein geschlossenes Tableau mit Auswahlfunktion f für ϕ.*

Beweis. Wir nehmen an, daß es kein geschlossenes Tableau mit Auswahl f für ϕ gibt. Dann existiert in einem erschöpften Tableau zu ϕ ein offener Ast A, der durch faire Auswahl der Tableauregeln stetig erweitert wurde. Die Menge aller Formeln von A bildet eine Hintikkamenge mit Auswahlfunktion (der Nachweis ist einfach). Nach Lemma 3.1 besitzt ϕ ein Modell, im Widerspruch zur der Unerfüllbarkeit von ϕ. □

3.5. Theoriebehandlung in geordnetem Tableau

Viele logische Probleme lassen sich einfacher beschreiben, wenn zur ihrer Formulierung Prädikate mit einer festgelegten Semantik benutzt werden können. Hierzu gehören insbesondere Gleichheit, Ordnungen und natürliche Zahlen mit Prädikaten zur Addition, Multiplikation etc. Zum Beispiel spricht man von der Gleichheitstheorie, wenn ein entsprechendes Prädikat die Semantik der Gleichheit besitzt (im folgenden sei dies das Prädikat $\approx$).

Die Semantik einer Theorie $\mathcal{T}$ läßt sich mit Hilfe einer (nicht notwendigerweise endlichen) universell abgeschlossenen, erfüllbaren Formelmenge definieren. Die Theorie $\mathcal{T}$ wird deswegen mit dieser Formelmenge identifiziert. Die

Einschränkung auf universell abgeschlossene Theorien ist für eine Theorie-Version des Satzes von Herbrand Voraussetzung. Sie stört allerdings in der Praxis nicht, da Existenzquantoren durch Skolemisierung eliminiert werden können, ohne die Erfüllbarkeit des Problems zu beeinträchtigen.

Die Begriffe Erfüllbarkeit, Interpretation etc. werden auf $\mathcal{T}$ eingeschränkt.

DEFINITION 3.18. Jede erfüllbare universell quantifizierte Formelmenge ist eine *Theorie* $\mathcal{T}$.

Eine Interpretation ist eine $\mathcal{T}$*-Interpretation*, wenn sie $\mathcal{T}$ erfüllt.

Eine Formel ϕ heißt $\mathcal{T}$*-erfüllbar*, wenn es eine $\mathcal{T}$-Interpretation gibt, die ϕ erfüllt. Andernfalls heißt ϕ $\mathcal{T}$*-unerfüllbar*.

Theorien können im Tableaukalkül mit verschiedenen Methoden behandelt werden:

- Zu einer gegeben Signatur können einer Formelmenge Φ zusätzliche Formeln Ψ beigefügt werden, so daß Φ $\mathcal{T}$-erfüllbar ist genau dann, wenn $\Phi \cup \Psi$ erfüllbar ist. Wir erörtern diese Methode in Abschnitt 3.5.1.
- Man kann den Tableaukalkül um neue Regeln erweitern, die die $\mathcal{T}$-Erfüllbarkeit sicherstellen. Darauf gehen wir kurz in Abschnitt 3.5.2 ein.
- Der vielversprechendste Ansatz ist das *Theorieschließen*. In diesem Fall wird die Theorie innerhalb eines Spezialbeweisers — dem Hintergrundbeweiser — behandelt (Abschnitt 3.5.3).

3.5.1. Hinzufügen zusätzlicher Axiome. Ein automatischer Beweiser kann am einfachsten um eine Theorie $\mathcal{T}$ ergänzt werden, wenn $\mathcal{T}$ dem zu beweisenden Problem hinzugefügt wird. Erfüllt eine Interpretation I eine Formelmenge $\Phi \cup \mathcal{T}$, so erfüllt I auch $\mathcal{T}$ und ist daher eine $\mathcal{T}$-Interpretation. Offenbar gilt auch die Umkehrung, so daß eine Formelmenge Φ genau dann $\mathcal{T}$-erfüllbar ist, wenn $\Phi \cup \mathcal{T}$ erfüllbar ist.

Zum Beispiel definieren die folgenden Formeln zu einer gegebenen Signatur die wichtige Gleichheitstheorie $\mathcal{E}$:

(1) $(\forall x)(x \approx x)$ (*Reflexivität*)

(2) $(\forall x)(\forall y)(x \approx y \Rightarrow y \approx x)$ (*Symmetrie*)

(3) $(\forall x)(\forall y)(\forall z)(x \approx y \wedge y \approx z \Rightarrow x \approx z)$ (*Transitivität*)

(4) für jedes Funktionssymbol $f \in \mathbf{F}$ mit $arity(f) = n > 0$:

$$(\forall x_1) \cdots (\forall x_n)(\forall y_1) \cdots (\forall y_n)((x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n) \Rightarrow f(x_1, \ldots, x_n) \approx f(y_1, \ldots, y_n))$$

(f*-Monotonie*)

(5) für jedes Prädikatensymbol $p \in \mathbf{P}, p \neq\approx$ mit $arity(p) = n > 0$:

$$(\forall x_1) \cdots (\forall x_n)(\forall y_1) \cdots (\forall y_n)((x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n) \Rightarrow (p(x_1, \ldots, x_n) \Rightarrow p(y_1, \ldots, y_n)))$$

(p*-Monotonie*)

Dieser einfache Ansatz ist allerdings mit einigen Nachteilen verbunden. Die Formelmenge wird je nach Signatur mit einer Vielzahl von neuen Formeln

aufgebläht, von denen in einem Beweis evtl. nur ein geringer Teil verwendet wird. Dazu kommt, daß bei transitiven und reflexiven Relationen wie Ordnungen und Gleichheit, einige der Formeln von $\mathcal{T}$ immer wieder angewendet werden können. Da der Suchraum sich hierdurch massiv erhöht, sind auch einfache Probleme mit Theorembeweisern, die diese Methode verwenden, nicht mehr in angemessener Zeit zu lösen.

Insbesondere für die Gleichheitstheorie $\mathcal{E}$ wurde deswegen schon frühzeitig versucht durch Modifikation der Probleme selbst, einige dieser Nachteile zu vermeiden. Ein Methode ist die in (Brand, 1975) vorgestellte *STE-Modifikation*, welche heute noch von vielen Theorembeweisern verwendet wird (z.B. SETHEO (Letz *et al.*, 1992) und KoMeT (Bibel *et al.*, 1994)).

Bei diesem Verfahren wird eine Klauselmenge M in eine neue Klauselmenge transformiert, so daß nur noch das Reflexivitätsaxiom $\forall x(x \approx x)$ nötig ist, um $\mathcal{E}$ vollständig zu behandeln.

Das Verfahren gliedert sich in drei Schritte, die wir hier am Beispiel der Klausel $P(f(a))$ erläutern. Die Reihenfolge, in der diese Transformationen ausgeführt werden, ist notwendig für eine vollständige Behandlung des Gleichheitsprädikats.

Im ersten Schritt werden alle nichtvariablen Argumente t eines jeden Prädikatensymbols (außer $\approx$) und Funktionssymbols durch eine Gleichung $t \approx x$ mit einer neuen Variablen x ersetzt. Dies ist die sogenannte *E-Modifikation*. Eine Klausel, die nicht mehr E-modifizierbar ist, ist in *E-Form*.

Beispiel 3.6. Zuerst wird $P(f(a))$ durch $\neg f(a) \approx x_1 \vee P(x_1)$ ersetzt. Danach gibt es nur noch a als nichtvariables Argument von $f(a)$ und damit ist $\neg a \approx x_2 \vee \neg f(x_2) \approx x_1 \vee P(x_1)$ eine E-Form von $P(f(a))$.

Nach dieser Modifikation kann auf die Axiome zur Monotonie von Prädikaten- und Funktionssymbolen verzichtet werden (Siehe Abbildung 3.2).

Das Symmetrieaxiom kann fortgelassen werden, wenn zu jeder Klausel $s \approx t \vee K$ einer Klauselmenge zusätzlich $t \approx s \vee K$ betrachtet wird. Dies ist die *S-Modifikation*. Eine Klauselmenge, die nicht mehr S-modifizierbar ist, ist in *S-Form*.

Beispiel 3.7. Da die E-Modifikation $\neg a \approx x_2 \vee \neg f(x_2) \approx x_1 \vee P(x_1)$ von $P(f(a))$ keine Gleichung enthält, ist sie auch S-Modifikation.

Die S-Modifikation von $a \approx b$ ist die Menge $\{a \approx b, b \approx a\}$.

Der Nachteil der S-Modifikation ist offensichtlich: Eine Klausel mit n Gleichungen wird in eine Menge von 2^n Klauseln transformiert.

Bei der *T-Modifikation* wird jede Klausel $s \approx t \vee K$ durch $(s \approx y \Rightarrow t \approx y) \vee K$ ersetzt, wobei y eine neue Variable ist und nirgendwo sonst in K auftreten darf. Eine Klausel ist in *T-Form*, wenn sie nicht mehr T-modifizierbar ist. Da die neu eingeführten Variablen nur zweimal in einer Klausel auftreten dürfen, ist gewährleistet, daß eine Klausel nicht beliebig oft T-modifizierbar ist.

Beispiel 3.8. Die Menge

$$\{\neg a \approx x \vee b \approx x, \neg b \approx y \vee a \approx y\}$$

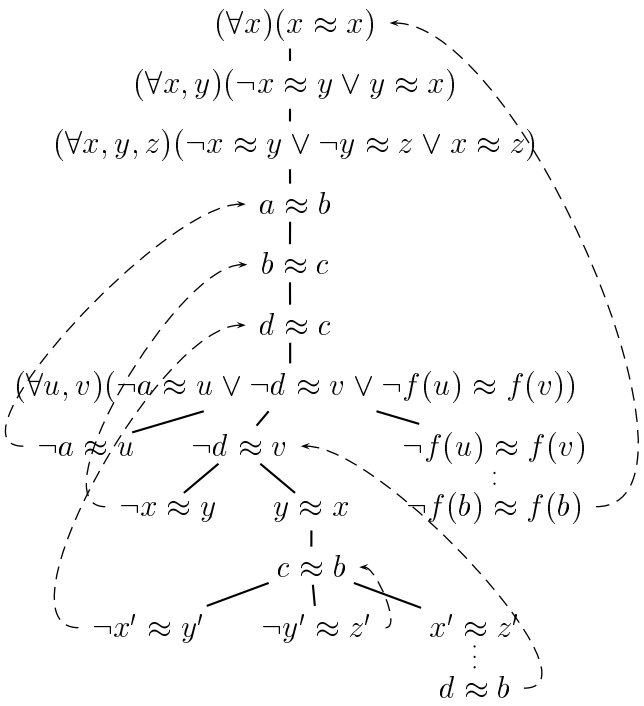


ABBILDUNG 3.2. Nach E-Modifikation von $\{a \approx b, b \approx c, d \approx c, \neg f(a) \approx f(d)\}$ können die Axiome zur Monotonie fortgelassen werden.

ist T-Modifikation der S-Form von $a \approx b$.

Wird eine Klauselmenge nach E- und S-Modifikation in T-Form gebracht, so kann auf das Symmetrie- und Transitivitätsaxiom verzichtet werden (Siehe Abbildung 3.3). Dies Beispiel zeigt aber auch deutlich zwei Nachteile dieses Verfahrens.

Einmal steigt der Suchraum durch die vielen neuen erzeugten Klauseln an, von denen die meisten gar nicht benötigt werden.

Viel problematischer ist allerdings die große Menge an neuen Abschlußmöglichkeiten. Durch die T-Modifikation gibt es bei Gleichungen immer einen Abschluß mit dem Reflexivaxiom. Diese Art des Abschlusses ist in vielen Fällen aber gerade unerwünscht und führt bei der Beweissuche nur in die Irre.

DEFINITION 3.19. Eine Klauselmenge M ist in *STE-Form* genau dann, wenn jede Klausel von M in E- und T-Form sowie M in S-Form ist.

M' heißt *STE-Modifikation* von M, wenn M' aus M durch obiges Verfahren in STE-Form transformiert wurde.

Da die STE-Form einer Klauselmenge M bis auf Variablenumbenennung eindeutig ist, spricht man auch von *der* STE-Form von M.

SATZ 3.6 (Brand, 1975). *M' sei die STE-Modifikation einer Klauselmenge M.*

M ist $\mathcal{E}$-erfüllbar genau dann, wenn $M' \cup \{x \approx x\}$ erfüllbar ist.

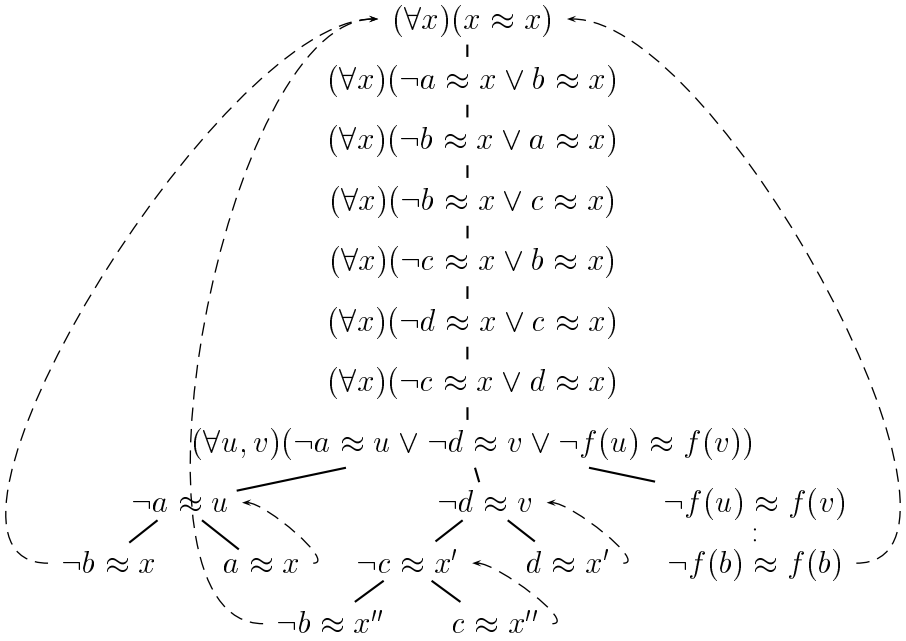


ABBILDUNG 3.3. Nach T-Modifikation der E- und S-Modifikation von $\{a \approx b, b \approx c, \neg c \approx a\}$ kann das Transitivitätsaxiom fortgelassen werden.

In der Praxis wird die STE-Modifikation nicht so durchgeführt wie hier beschrieben, da schon bei trivialen Problemen die resultierende Klauselmenge zu groß wird.

Oftmals können einige Modifikationsschritte unterlassen werden, ohne die $\mathcal{E}$-Erfüllbarkeit zu beeinträchtigen. Ist zum Beispiel sichergestellt, daß auf die E-Modifikation $(a \approx x_1 \Rightarrow P(x_1)) \vee (a \approx x_2 \Rightarrow Q(x_2))$ der Klausel $P(a) \vee Q(a)$ nur das Reflexivitätsaxiom angewendet werden kann, so kann man den E-Modifikationsschritt für diese Klausel unterlassen, da bei zwei Resolutionsinferenzen mit $v \approx v$ nur die ursprüngliche Klausel $P(a) \vee Q(a)$ entstünde.

Sowohl die STE-Modifikation als auch bloßes Hinzufügen der Formelmenge $\mathcal{T}$ zum Problem, haben im Fall geordneter Tableaus den immensen Nachteil, eine große Anzahl maximal verbundener Formeln zu erzeugen. Insbesondere das Reflexivitätsaxiom besitzt eine Vielzahl von (maximalen) Konnektionen.

Zusätzlich wird bei der STE-Modifikation die Struktur der Formelmenge derart zerstört und flachgewalzt, daß ein sinnvoller Einsatz vieler Ordnungen nicht möglich ist. Aufgrund der gleichförmigen Termtiefe der Literale, ist es unsinnig die Termtiefenordnung $\prec_\tau$ zu benutzen.

Werden Constraints verwendet, um trotzdem eine starke Einschränkung zu erzielen, so sind diese wahrscheinlich sehr oft unerfüllbar, da die Terme erst während des Beweisens stärker strukturiert werden. Dies ist ein in der Praxis recht unerwünschter Effekt, da der Beweiser dann öfter zurücksetzen muß, um sicherzustellen, daß das Tableau weiterhin geordnet ist.

3.5.2. Theoriebehandlung mit neuen Tableauregeln. Viele Theorien können in einen bestehenden Kalkül durch zusätzliche Regeln behandelt werden. In (Beckert, 1996) werden für die Gleichheitstheorie $\mathcal{E}$ einige Erweiterungen erörtert. Im folgenden betrachten wir kurz den dort behandelten Ansatz von Jeffrey (Jeffrey, 1967) für Grundformelmengen im Zusammenhang mit A-geordneten Tableaus.

Zu einer vollständigen Theoriebehandlung sind dazu die in Tabelle 3.7 angegeben zusätzlichen Tableauregeln nötig. Ein Tableauast A kann zusätzlich abgeschlossen werden, wenn eine Ungleichung $\neg t \approx t$ in A vorkommt.

$$\frac{t \approx s \quad \phi[t]}{\phi[s]} \qquad \frac{s \approx t \quad \phi[t]}{\phi[s]}$$

TABELLE 3.7. Jeffreys Tableauregeln für Gleichheit. Dabei ist $\phi[s]$ durch Substitution eines Auftretens von t in ϕ mit s aus ϕ entstanden.

Mit Hilfe dieser Regeln sind aus einer gegebenen Formelmenge alle $\mathcal{E}$-erfüllbaren Formeln ableitbar. Im Fall von A-geordneten Tableaus können die Regeln auf Formeln ϕ eingeschränkt werden, die eine maximale Konnektion in den betrachteten Ast besitzen (Siehe Tabelle 3.8).

$$\frac{t \approx s \quad \phi[t]}{\phi[s]} \qquad \frac{s \approx t \quad \phi[t]}{\phi[s]}$$

wobei $\phi[t]$ eine maximale Konnektion in den betrachteten Ast A besitzt.

TABELLE 3.8. Jeffreys Tableauregeln für A-geordnete Tableaus.

Jeffreys Regeln wurden in (Fitting, 1996) auch für die Prädikatenlogik erweitert. Die Vollständigkeit kann analog zum Beweis von Satz 3.5 mit einer erweiterten Hintikkamenge gezeigt werden, wobei die Gleichheit im Beweis ähnlich wie in (Fitting, 1996, Seiten 279–284) gehandhabt wird.

3.5.3. Theorieschließen. Das Konzept des Theorieschließens wurde für den Resolutionskalkül und Grundklauselmengen erstmals in (Stickel, 1985) vorgestellt. Der Ansatz wurde später auch für andere Kalküle übernommen und weiter ausgearbeitet. Bei allen diesen Ansätzen wird das Beweissystem in einen Vorder- und Hintergrundbeweiser aufgespalten, wobei letzterer ausschließlich den Theorieteil des Problems behandelt. Es wird zwischen partiellem und totalem Theorieschließen unterschieden. Während der Hintergrundbeweiser im ersten Fall neue Formeln ableitet, die dem Vordergrundbeweiser für weitere Inferenzen zur Verfügung stehen, werden im zweiten Fall nur Abschlußsubstitutionen berechnet. Gerade durch die strikte Trennung eines automatische Beweisers in einen Vorder- und Hintergrundbeweiser bei totalem Theorieschließen können Theorien effizient behandelt werden. Eine Übersicht

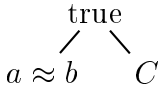
über Theorieschließen mit einigen Beispielen besonders zur Gleichheitstheorie ist in (Baumgartner *et al.*, 1992) zu finden. Eine Methode die Effizienz des Gesamtsystems von Tableau- und Hintergrundbeweiser zu verbessern, wurde in (Beckert & Pape, 1996) vorgestellt.

Beim Einbau von Theorien in Tableaukalkülen, die auf der Konnektionsmethode, wie z.B. Modellelimination (Loveland, 1968) beruhen, dürfen Klauseln zusätzlich zu den schon vorhandenen Kriterien einen Ast erweitern, wenn dadurch ein Abschluß bezüglich der betrachteten Theorie möglich ist. Meistens wird partielles Theorieschließen angewendet, da es oftmals recht aufwendig ist, einen Theorieunifikator zu berechnen.

Beispiel 3.9. Betrachte die Klauselmenge

$$\{a \approx b \vee C, P(a) \vee D, \neg P(b) \vee E\}$$

mit dem folgenden partiellen Tableau:



Mit den Literalen $P(a)$ und $P(b)$ kann der linke Ast, der bei der Erweiterung mit $P(a) \vee D$ und $P(b) \vee E$ entsteht, bzgl. der Gleichheitstheorie $\mathcal{E}$ abgeschlossen werden. Deswegen ist es möglich, bei der Konnektionsmethode diesen Ast entsprechend zu erweitern. Da keine neuen Formeln erzeugt werden, handelt es sich hier um eine totale Erweiterung des Astes. Abbildung 3.4 zeigt das entsprechende Tableau.

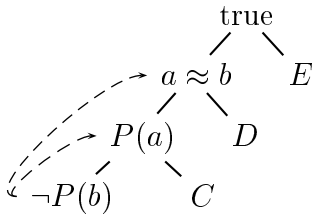


Abbildung 3.4. Totale Erweiterung

Wie man an diesem Beispiel sieht, ist die Konnektionsbedingung beim Theorieschließen etwas komplexer als zuvor, da mehr als eine Klausel betrachtet werden muß und die Berechnung der Konnektion meist zeitaufwendiger ist.

In (Petermann, 1992) wird ein allgemeines Konzept vorgestellt, um Theorien im Zusammenhang mit Hintergrundbeweisern in Tableaukalküle einzubauen. Dazu werden die Begriffe eines komplementären Paares und einer Konnektion verallgemeinert.

Definition 3.20. Eine Menge $\{L_1, \ldots, L_n\}$ von Literalen heißt $\mathcal{T}$-*komplementär* genau dann, wenn jede ihrer Instanzen $\mathcal{T}$-unerfüllbar ist.

$\mathcal{T}$-komplementäre Literale übernehmen wie „normale“ komplementäre Literale die Rolle elementarer Widersprüche.

Beispiel 3.10. Die Menge $\{a \approx b, p(a), \neg p(b)\}$ ist $\mathcal{E}$-komplementär und $\forall x \forall y(\neg x \approx y)$ ist nicht $\mathcal{E}$-komplementär aber $\mathcal{E}$-unerfüllbar.

Der Unterschied zwischen einer Konnektion (z.B. $p(x), \neg p(y)$) und einem komplementären Paar (z.B. $p(x), \neg p(x)$) besteht darin, daß erstere eine notwendige Bedingung zu letzterem ist. Konnektionen werden benötigt, um mit Hilfe von Unifikation Abschlußsubstitutionen zu berechnen. Dabei stellt ein komplementäres Paar auf einem Ast sicher, daß dieser auch geschlossen bleibt, wenn weitere Substitutionen auf das Tableau angewendet werden.

Definition 3.21. Eine Substitution σ heißt $\mathcal{T}$-*Unifikator* zu einer Menge M von Literalen, wenn $\sigma(M)$ $\mathcal{T}$-komplementär ist.

Eine Menge M von Literalen heißt $\mathcal{T}$-*Konnektion*, wenn ein $\mathcal{T}$-Unifikator für M existiert.

Der Hintergrundbeweiser wird nun eingesetzt, um $\mathcal{T}$-*Unifikationsprobleme* zu lösen. Das heißt, zu einer gegebenen Menge von Literalen muß ein $\mathcal{T}$-Unifikator gefunden werden. Dies kann entweder für jeden Ast des Tableaus einzeln geschehen oder für alle Äste auf einmal. Im letzteren Fall spricht man von einem *simultanen* $\mathcal{T}$-*Unifikationsproblem*.

Als Eingabe zu einem $\mathcal{T}$-Unifikationsproblem bekommt der Hintergrundbeweiser eine Teilmenge der Formeln eines Astes. Diese Menge wird *Schlüssel* genannt. Wird kein simultanes Problem behandelt, sondern jeder Ast einzeln betrachtet, so muß sichergestellt sein, daß der Hintergrundbeweiser genügend Lösungen berechnet, weil nicht jede gefundene Lösung zu einem Beweis führen muß.

Da es mehrere Konnektionen auf einem Ast geben kann, muß der Hintergrundbeweiser ausreichend viele verschiedene Konnektionen (also Abschlußsubstitutionen) berechnen können, damit der Kalkül vollständig bleibt. Es brauchen dazu allerdings nicht notwendigerweise alle möglichen Konnektionen berechnet werden. Zum Beispiel ist es ausreichend bei einem partiellen Pfad $\{p(x), \neg p(x), q(x), \neg q(x)\}$ durch eine Klauselmenge, nur eine der beiden vorhandenen Konnektionen zu betrachten, da deren Unifikator identisch ist. Die folgende Definition aus (Petermann, 1992) beschreibt diesen Sachverhalt.

Definition 3.22. $\mathcal{U}$ sei eine berechenbare Menge von $\mathcal{T}$-Konnektionen einer Menge M' aller Instanzen einer Klauselmenge M.

$\mathcal{U}$ heißt $\mathcal{T}$-*vollständig zu* M, wenn zu jedem $\mathcal{T}$-komplementären Pfad p durch M' ein $u \in \mathcal{U}$ und eine Substitution σ mit $\sigma(u) \subseteq p$ existiert.

M' ist bezüglich Substitutionen abgeschlossen. Diese Eigenschaft ist notwendig, da während eines Tableaubeweises beliebige Substitutionen auftreten können, die einen Ast verändern. $\mathcal{U}$ muß berechenbar sein, da ansonsten die $\mathcal{T}$-Komplementarität einer Menge von Literalen nicht erkannt werden kann.

Falls die Theorie $\mathcal{T}$ entscheidbar ist, gilt die folgende Aussage:

BEISPIEL 3.11. M sei eine Klauselmenge und $\mathcal{U}_M$ sei Menge aller $\mathcal{T}$-Konnektionen von Pfaden durch alle Instanzen von M. Dann ist $\mathcal{U}_M$ $\mathcal{T}$-vollständig zu M.

In der Praxis ist eine solche $\mathcal{T}$-vollständige Menge nicht besonders hilfreich, da sie zu viele redundante Konnektionen enthält. Deswegen ist man bestrebt kleinere $\mathcal{T}$-vollständige Mengen zu einer Klauselmenge M zu bekommen. Um die Vollständigkeit zu garantieren, muß zu jeder solchen $\mathcal{T}$-vollständigen Menge ein entsprechendes Herbrand-Theorem bewiesen werden. Wir wollen auf dieses Problem nicht näher eingehen.

Wir sind nun in der Lage maximale $\mathcal{T}$-Konnektionen zu definieren. Sie sind eine direkte Verallgemeinerung von maximalen Konnektionen. Anstatt A-Ordnungen zu betrachten, können hier auch wieder Auswahlfunktionen oder Auswahlen benutzt werden. Die Ergebnisse sind mühelos übertragbar.

DEFINITION 3.23. M sei eine Klauselmenge. Eine Klausel $K \in M$ hat eine maximale $\mathcal{T}$-Konnektion in eine Menge

$$\{K_1, \ldots, K_n\} \subseteq M \ ,$$

wenn es ein maximales Literal $L \in K$, maximale $L_i \in K_i$ und eine Substitution σ gibt, so daß $\{L, L_1, \ldots, L_n\}\sigma$ $\mathcal{T}$-komplementär ist.

In diesem Fall schreiben wir abkürzend $\langle L, \{L_1, \ldots, L_n\}, \sigma\rangle$.

Tabelle 3.9 führt die Regeln für A-geordnete Klauseltableaus mit Theorie-Konnektionen auf. Wir gehen dort und im folgenden davon aus, daß alle $\mathcal{T}$-Konnektionen $\langle L, N, \sigma\rangle$ minimal sind. Das heißt, für keine echte Teilmenge $N' \subset L \cup N$ ist $N'\sigma$ $\mathcal{T}$-komplementär. Diese Einschränkung ist sinnvoll, da sich sonst eine $\mathcal{T}$-Konnektion beliebig mit Literalen erweitern läßt.

$$\frac{T \cup \{A\}, C}{(T \cup \{A \cup \{L_2\}\} \cup \ldots \cup \{A \cup \{L_n\}\})\sigma, C \cup \langle L_1, K\rangle\sigma}$$

falls $K = L_1 \vee \ldots \vee L_n \in M$ eine maximale $\mathcal{T}$-Konnektion $\langle L_1, N, \sigma\rangle$ in $A \cup M$ besitzt und von keinem Literal aus A subsumiert wird.

$$\frac{T \cup \{A \cup K\}, C}{(T - \{A \cup K\})\sigma, C\sigma}$$

falls es einen $\mathcal{T}$-Unifikator σ gibt, so daß $K\sigma$ $\mathcal{T}$-komplementär ist und für jedes $\langle L, K'\rangle$ gilt: L ist maximal in K'.

TABELLE 3.9. A-geordnete Klauseltableaus, Theorieversion

Wie zeigen die Vollständigkeit des Kalküls über dessen Grundvollständigkeit. Für den allgemeinen Fall prädikatenlogischer Klauseltableaus benötigen wir folgende Version des Satzes von Herbrand.

SATZ 3.7 (Satz von Herbrand, Theorieversion). *Zu jeder $\mathcal{T}$-unerfüllbaren Klauselmenge M gibt es eine $\mathcal{T}$-unerfüllbare Menge von Grundinstanzen von M.*

BEWEIS. Wenn M $\mathcal{T}$-unerfüllbar ist, so ist $M \cup \mathcal{T}$ unerfüllbar. Nach dem üblichen Herbrand-Theorem gibt es eine unerfüllbare Menge $M' \cup \mathcal{T}'$ von Grundinstanzen von $M \cup \mathcal{T}$. Da $\mathcal{T}$-universell abgeschlossen ist, gilt offenbar $\mathcal{T} \models \mathcal{T}'$, und $M' \cup \mathcal{T}'$ ist aus $M' \cup \mathcal{T}$ ableitbar. Wäre $M' \cup \mathcal{T}$ $\mathcal{T}$-erfüllbar, dann auch $M' \cup \mathcal{T}'$. Also ist M' $\mathcal{T}$-unerfüllbar. □

LEMMA 3.3. *M sei eine Grundklauselmenge, $\mathcal{T}$ eine Theorie und A ein offener Ast eines erschöpften A-geordneten $\mathcal{T}$-Grundklauseltableaus. Dann besitzt M ein $\mathcal{T}$-Modell.*

BEWEIS. Die Elemente des offenen Astes bilden eine partielle $\mathcal{T}$-Interpretation I von M. M' sei die Menge aller Klauseln aus M, die nicht von I erfüllt werden und J sei die Menge aller Literale, die maximal in Klauseln von M' vorkommen.

J entspricht ebenfalls einer partiellen $\mathcal{T}$-Interpretation von M, da sonst Klauseln aus M' eine maximale $\mathcal{T}$-Konnektion besitzen. Dann könnte aber eine dieser Neustartklauseln den offenen Ast erweitern.

Die Vereinigung $I \cup J$ ist eine $\mathcal{T}$-Interpretation von M, da sonst einen Teilmenge von Literalen I und Klauseln von M' eine maximale $\mathcal{T}$-Konnektion hat. □

SATZ 3.8. *M sei eine $\mathcal{T}$-unerfüllbare Klauselmenge, $\mathcal{T}$ eine Theorie, $\prec_A$ eine A-Ordnung und $\mathcal{U}$ eine berechenbare $\mathcal{T}$-vollständige Menge von $\mathcal{T}$-Konnektionen zu M. Dann gibt es ein geschlossenes $\prec_A$-geordnetes Klauseltableau zu M.*

BEWEIS. Zu M gibt es nach Satz 3.7 eine endliche $\mathcal{T}$-unerfüllbare Grundklauselmenge M', zu der ein geschlossenes Grundklauseltableau T existiert (Lemma 3.3). Aufgrund der Substitutivität von $\prec_A$, können maximale $\mathcal{T}$-Konnektionen des Grundklauseltableaus zu maximalen $\mathcal{T}$-Konnektionen von M gehoben werden. Hat eine Grundinstanz einer Klausel $K \in M$ einen Ast von T erweitert, so kann K das zugehörige Tableau zu M erweitern.

$\mathcal{U}$ garantiert, daß ausreichend $\mathcal{T}$-Konnektionen zwischen Klauseln aus M berechnet werden können. □

3.5.4. Gleichheitskonnektion. Selbst bei entscheidbaren Theorien ist es viel zu aufwendig, Konnektionen zu berechnen. Deswegen gibt man sich in der Praxis mit einfacheren Kriterien zur $\mathcal{T}$-Komplementarität zufrieden. Wir wollen im folgenden ein solches Kriterium skizzieren.

Wird die Gleichheitstheorie mit einem totalen Hintergrundbeweiser behandelt, so sind *E-Unifikationsprobleme* zu lösen. Sind alle beteiligten Variablen universell, dann wird von *universeller* E-Unifikation (Siekmann, 1989) gesprochen; sind alle Variablen (wie meistens im Tableaukalkül) frei, so handelt es sich um *starre* E-Unifikation (Gallier *et al.*, 1992); werden freie und universelle

Variablen in den zu behandelnden Problemen vermischt, so spricht man von *gemischter* E-Unifikation (Beckert, 1994).

DEFINITION 3.24. Ein *gemischtes* E-Unifikationsproblem

$$\langle E, s, t\rangle$$

besteht aus einer endlichen Menge E von Gleichungen der Form

$$(\forall x_1)\cdots(\forall x_n)(l \approx r)$$

und Termen s und t. Eine Substitution σ ist eine Lösung zu diesem Problem genau dann, wenn

$$E\sigma \models (s\sigma \approx t\sigma) \text{ gilt,}$$

wobei die freien Variablen in $E\sigma$ starr sind, d.h. als Konstanten betrachtet werden.

Die Berechnung eines E-Unifikators zu einer gegebenen Menge M von Literalen (einem Schlüssel), kann auf folgende E-Unifikationsprobleme zurückgeführt werden:

DEFINITION 3.25. M sei eine Menge von Literalen. Dann besteht eine vollständige Menge von E-Unifikationsproblemen zu M aus

$$\langle E, \langle s_1, \ldots s_n\rangle, \langle t_1, \ldots t_n\rangle\rangle$$

für alle potentiell komplementären Paare

$$P(s_1, \ldots, s_n), \neg P(t_1, \ldots, t_n) \in M \text{ mit } P \neq \approx$$

und aus

$$\langle E, s, t\rangle$$

für jede Ungleichung $\neg s \approx t \in M$. E ist die Menge aller Gleichungen von M.

BEMERKUNG 3.3. p sei eine minimale Menge $\mathcal{E}$-komplementärer Grundliterale, d.h. jede echte Teilmenge von p ist nicht mehr $\mathcal{E}$-komplementär. Dann gibt es genau ein lösbares E-Unfikationsproblem $\langle E, s, t\rangle$ zu p.

Gibt es eine Substitution σ, so daß $s\sigma = t\sigma$ ist, dann muß E die leere Menge sein, da σ schon Lösung des Problems ist. In diesem Fall besteht p entweder aus einer Ungleichung $\neg s \approx t$ oder aus einem komplementären Paar $\langle L, L', \sigma\rangle$.

Ansonsten ist E nicht leer und eine Gleichung von E läßt sich auf t oder s anwenden.

Mit Hilfe dieser Eigenschaft läßt sich das folgende notwendige Kriterium zur $\mathcal{E}$-Komplementarität definieren:

DEFINITION 3.26. Eine Gleichung $s \approx t$ besitzt eine *schwache Gleichheits-Konnektion* zu einem Literal L, wenn es eine Position p in L mit $L_{|p} = u$ und einen allgemeinsten Unifikator σ gibt, so daß $u\sigma = s$ oder $u\sigma = t$ gilt.

Eine Ungleichung $s \not\approx t$ hat eine *schwache Gleichheits-Konnektion* zu sich selbst, wenn es einen allgemeinsten Unifikator σ mit $s\sigma = t\sigma$ gibt.

Diese Art von Gleichheitskonnektion laßt sich in polynomieller Zeit berechnen. Im Gegensatz zur NP-vollständigen starren E-Unifikation (Gallier *et al.*, 1990) oder dem gar unentscheidbaren universellen Fall stellt dies sicher einen erheblichen Fortschritt dar. Nachteilig ist die geringe Einschränkung, die mit ihr erzielt wird, da eine Gleichung mit einer starren Variablen auf jeden anderen Term angewendet werden kann bzw. eine Gleichung immer auf eine starre Variable. Zumindest bei universellen Variablen kann dies untersagt werden.

Mit Sorten kann die Anzahl schwacher Gleichheitskonnektionen noch weiter reduziert werden.

3.6. Vergleich und offene Fragen

3.6.1. Weitere Einschränkungen. In (Baumgartner & Furbach, 1994) wird ein Kalkül vorgestellt, der einige Ähnlichkeit zu Klauseltableaus mit Auswahlfunktion aufweist. Es handelt sich dabei um eine Einschränkung von Konnektionstableaus und andererseits um eine zusätzliche Liberalisierung. Die in (Baumgartner & Furbach, 1994) gegebene Auswahlfunktion wählt aus einer Grundklausel genau ein *positives* Literal aus, sofern ein solches vorhanden ist. Extensionsschritte werden nur noch zugelassen, wenn eine Konnektion zu dem ausgewählten Literal besteht. Da diese Form der Einschränkung nicht mehr vollständig ist (sie ist es auch mit schwacher Konnektionsbedingung nicht mehr), wird auch hier ein Neustart zugelassen. Als Neustartklauseln sind alle rein negativen Klauseln zugelassen, da eine von diesen sicherlich relevant für einen Beweis ist (ansonsten gibt es ein Modell für die Klauselmenge).

Dieser Kalkül ist — im Gegensatz zu Klauseltableaus mit Auswahlfunktion — mit der Regularitätseinschränkung in ihrer allgemeinsten Form unvollständig, da Neustartklauseln mehrmals zugelassen werden müssen. Deswegen wird dort Regularität nur zwischen positiven Literalen auf einem Ast und zwischen allen Literalen betrachtet, die von zwei Neustartklauseln eingefaßt sind. Dies wird in (Baumgartner & Furbach, 1994) *blockweise Regularität* genannt.

Zusätzlich werden noch Reduktionsschritte von negativen Blättern untersagt, so daß auch in diesem Fall ein Neustart notwendig ist.

Der Vorteil dieses Kalküls liegt in der geringen Anzahl von Konnektionen (abhängig von einer geeigneten Auswahlfunktion) und von Reduktionsschritten, die eine effiziente PTTP-Implementierung (Stickel, 1988) zulassen. Auf Probleme, die sich aus der Zunahme der Beweislänge im Zusammenhang mit einer iterativen Tiefensuchstrategie ergeben, wird leider nicht eingegangen.

Im Fall von Klauseltableaus mit Auswahlfunktion ist nicht bekannt, ob diese mit Konnektionsbedingung noch vollständig sind. Tabelle 3.10 zeigt eine mögliche Modifikation. Im Gegensatz zur bisherigen Notation, schreiben wir einen Ast als Folge von Literalen. Bei dem letzten Literal in der Sequenz handelt es sich um das Blatt des Astes. Der Operator $\cup$ bewirkt in diesem Zusammenhang, daß ein neues Literal am Ende des Astes angefügt wird.

Wir haben keine zusätzliche Regularitätseinschränkung angegeben, weil nicht klar ist, wie diese auszusehen hat, damit der Kalkül noch vollständig

$$\frac{T \cup \{A\}, C}{(T \cup \{A \cup \{L_2\}\} \cup \ldots \cup \{A \cup \{L_n\}\})\sigma, C \cup \langle L_1, K\rangle\sigma}$$

falls $K = L_1 \vee \ldots \vee L_n \in M$ eine maximale Konnektion $\langle L_1, L, \sigma\rangle$ in dem nicht geschlossenen Ast $A = \langle L'_1, \ldots, L\rangle\sigma$ oder in M besitzt.

$$\frac{T \cup \{A \cup \{L, \overline{L}\}\}, C}{(T - \{A \cup \{L, \overline{L}\}\})\sigma, C\sigma}$$

falls es einen allgemeinsten Unifikator σ mit $L\sigma = \overline{L'}\sigma$ gibt und für alle $\langle L'', K\rangle \in C\sigma$ gilt: L'' ist maximal in K.

TABELLE 3.10. Konnektionstableaus mit Auswahlfunktion

ist. Wie folgendes Beispiel zeigt, gibt es Situationen, in denen eine Klausel zweimal zu einem Beweis benutzt werden muß:

BEISPIEL 3.12. Betrachte die Klauselmenge

$$M = \{\underline{\neg C}, \underline{C} \vee B, \underline{\neg B} \vee A, \underline{\neg B} \vee \neg A\}$$

mit A-Ordnung $A < B < C$ (maximale Literale sind unterstrichen). Abbildung 3.5 zeigt ein Tableau zu M bei dem es notwendig ist, $A \vee B$ zweimal auf einem Ast zuzulassen, falls die Konnektionsbedingung benutzt wird, um Extensionen einzuschränken. Beachte, daß sich diese Situation nicht vermeiden läßt.

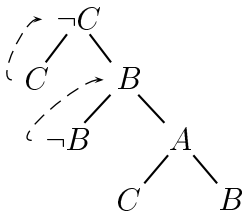


ABBILDUNG 3.5. Keine Regularität bei Konnektionsbedingung

3.6.2. Nichthebbare Ordnungen. In (de Nivelle, 1996) wird für geordnete Resolution gezeigt, daß man mit folgener Abschwächung der Substitutivitätseigenschaft einer A-Ordnung noch einen vollständigen geordneten Resolutionskalkül erhält.

DEFINITION 3.27. Ein *nicht hebbare* Ordnung $\sqsubset$ ist eine binäre Relation auf der Menge der Atome, für die gilt.

- $\sqsubset$ ist *stabil (bzgl. Variablenumbenennungen)*, d.h. für jede Variablenumbenennung θ und Atome A und B mit $A \sqsubset B$ gilt auch $A\theta \sqsubset B\theta$.[3]
- Für jede strikte Instanz $A\sigma$ eines Atoms A gilt $A\sigma \sqsubset A$.

[3]In der Literatur ist mit Stabilität häufig die Substitutivität einer Ordnung gemeint. Dies ist hier nicht der Fall.

Zwar erfüllt jede A-Ordnung die erste Bedingung, aber niemals die zweite, da A-Ordnungen strikte Ordnungen sind.

DEFINITION 3.28. Für zwei Atome A und B gilt $A \sqsubset_\# B$ genau dann, wenn die Anzahl der Funktionssymbole von A größer ist als die von B.

$\sqsubset_\#$ ist eine nicht hebbare Ordnung und keine A-Ordnung, da $P(f(x)) \not\sqsubset_\# P(y)$ aber $P(y)\sigma \sqsubset_\# P(f(x))\sigma$ mit $\sigma = \{y \leftarrow a, x \leftarrow f(f(a))\}$ gilt ($\sqsubset_\#$ ist also nicht substitutiv).

Es stellt sich die Frage, ob nichthebbare Ordnungen auch für Klauseltableaus mit Auswahlfunktion zu nutzen sind. Wir zeigen, daß es ausreichend ist zu verlangen, daß eine Auswahlfunktion stabil bezüglich Variablenumbenennungen ist.

DEFINITION 3.29. Eine Auswahlfunktion f auf Klauseln heißt *stabil (bzgl. Variablenumbenennungen)*, wenn zu jeder Variablenumbenennungen θ und jeder Klausel K mit $f(K) = L$ auch $f(K\theta) = L\theta$ gilt.

SATZ 3.9. *M sei eine unerfüllbare Klauselmenge und f eine stabile Auswahlfunktion, dann gibt es ein geschlossenes Klauseltableau mit Auswahlfunktion f für M.*

BEWEIS. Klauseln seien hier Folgen von Literalen.

Nach dem Satz von Herbrand existiert eine endliche unerfüllbare Grundklauselmenge M' von Grundinstanzen aus M. Wir bilden eine indizierte Klauselmenge $M' : M$ wie folgt:

Zu jedem $K' = L'_1 \vee \ldots \vee L'_n \in M'$, welches Grundinstanz von $K = L_1 \vee \ldots \vee L_n \in M$ ist, sei die indizierte Klausel $K' : K = L'_1 : L_1 \vee \ldots \vee L'_n : L_n$ in $M' : M$.

Wir definieren eine Auswahlfunktion f' auf $M' : M$. Es sei $f'(K' : K) = L' : L$ genau dann, falls $f(K) = L$ gilt. f' ist wohldefiniert, da f stabil bezüglich Variablenumbenennungen ist. f' ist hebbar, da $M' : M$ eine Menge indizierter *Grund*klauseln ist.

Nach Satz 3.5 gibt es ein geschlossenes Tableau T' mit Auswahlfunktion f' zu $M' : M$. Dieses läßt sich in ein geschlossens Tableau T zu M transformieren, indem eine Knotenmarkierung $L' : L$ von T' durch den Index L ersetzt wird und der allgemeinste Unifikator zweier Indizes auf T angewendet wird, falls die zugehörigen Grundliterale einen Ast in T' abschließen.

Wir müssen noch zeigen, daß das Tableau T mit den Tableauregeln von Tabelle 3.3 konstruiert werden kann. Es sind zwei Fälle zu unterscheiden:

1. Ist ein Ast von T' aufgrund einer maximalen Konnektion $\langle L'_1 : L_1, L'_2 : L_2, \mathrm{id}\rangle$ einer Klausel $K' : K$ erweitert worden, so hat K auch eine maximale Konnektion $\langle L_1, L_2, \sigma\rangle$ in diesen Ast, da L_2 aufgrund der Stabilität von f aus K ausgewählt wird.
2. Ist ein Tableauast mit einer Neustartklausel $K_1 : K'_1$ erweitert worden, so gibt es eine Klausel $K'_2 : K_2 \in M' : M$ mit maximaler Konnektion zu $K'_1 : K_1$. Offenbar hat dann auch K_1 eine maximale Konnektion zu K_2 (Stabilität).

□

Dieses Ergebnis läßt sich leider nicht auf Tableaus mit Auswahlfunktion für beliebige prädikatenlogische Formeln übertragen. Formeln dieser Tableaus enthalten freie Variablen, die wie Konstanten zu behandeln sind. Die Stabilitiät bezüglich Variablenumbenennungen ist deshalb nicht mehr ausreichend für einen vollständigen Kalkül.

Im Fall von A-Ordnungen, die ebenfalls stabil bezüglich Variablenumbenennungen sind, bedeutet die Aussage von Satz 3.9, daß A-geordnete Klauseltableaus auch ohne die Verwendung von Ordnungsconstraints vollständig sind.

KAPITEL 4

Geordnete Tableaus mit Constraints

> NEWTON: Enschuldigen Sie, doch weil wir gerade von Ordnung gesprochen haben: Hier dürfen nur die Patienten rauchen und nicht die Besucher. Sonst wäre gleich der ganze Salon verpestet.
>
> Friedrich Dürrenmatt —
>
> Die Physiker

Im folgenden betrachten wir ausschließlich A-geordnete Tableaus für beliebige prädikatenlogische Formeln. Wie wir in Abschnitt 3.3.1 schon bemerkten, können in diesem Fall Ordnungsconstraints benutzt werden, um so wenig wie möglich von der Einschränkung des Kalküls zu verlieren. Das folgende Beispiel soll illustrieren, daß dies sinnvoll sein kann.

Betrachte die Klauselmenge

$$\begin{aligned} M \quad = \{ \quad & \underline{Q(b,c)} \vee \underline{Q(x,x)}, \underline{\neg Q(y,y)} \vee P(y,a), \\ & \underline{\neg Q(z,z)} \vee P(z,d), \underline{\neg P(a,a)} \vee \ , \ , \underline{\neg P(d,d)}, \\ & \underline{\neg Q(b,c)} \vee \underline{R(b)}, \underline{\neg R(b)} \} \end{aligned}$$

zusammen mit $\prec_L$, wobei $P < Q < R$ und $a < b < c < d$ sei (maximale Literale sind unterstrichen). Die erste Klausel von M hat eine maximale Konnektion mit dem zweiten Literal zu den beiden folgenden Klauseln und darf deswegen das leere Tableau erweitern (Neustart). Der linke Ast läßt sich dann problemlos mit den letzten beiden Klauseln von M schließen. Abbildung 4.1 zeigt das Tableau mit noch offenem rechten Ast.

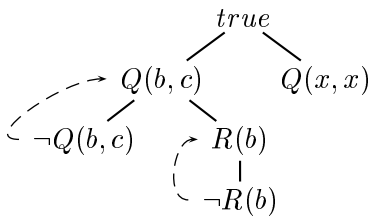


ABBILDUNG 4.1. Partielles Tableau für M (siehe Text)

Der rechte Ast kann mit der zweiten oder dritten Klausel von M erweitert werden. Wenn die Klausel $\neg Q(z,z) \vee P(z,d)$ für einen Extensionsschritt

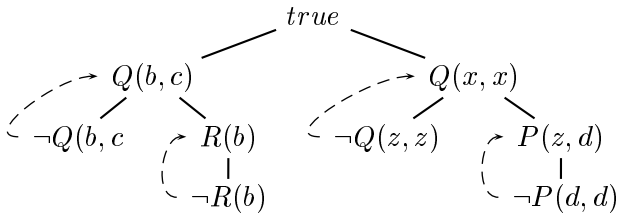


ABBILDUNG 4.2. Gute Auswahl (siehe Text)

gewählt wird, so kann man das Tableau nach einem weiteren Schritt abschließen (Abbildung 4.2).

Die Wahl der Klausel $K = \neg Q(y, y) \vee P(y, a)$ verursacht allerdings einen größeren Aufwand. Nach einem weiteren Extensionsschritt bleibt ein offener Ast mit , übrig (Abbildung 4.3). Da , eine beliebige Formel sein kann, ist das Tableau evtl. nur noch mit exponentiellem Zeitaufwand zu schließen. Eine Extension des Tableaus mit K sollte deswegen — wenn möglich — verhindert werden.

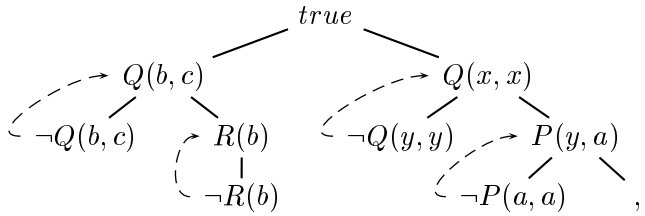


ABBILDUNG 4.3. Schlechte Auswahl (siehe Text)

Dies kann man mit $\prec_L$-Ordnungsconstraints erreichen. Werden diese benutzt, so ist nach dem initialen Schritt bekannt, daß das Tableau nur dann A-geordnet ist, wenn $x \not\prec_L b \vee x \not\prec_L c$ gilt. Wenn daraufhin K und $\neg P(a, a) \vee$, auf den Ast geholt werden, ist die Ordnungsbedingung verletzt, da $a \prec_L b$ und $a \prec_L c$ gilt. Der letzte Extensionsschritt kann daher vermieden werden.

Natürlich können wir ein Beispiel auch so konstruieren, daß Ordnungsconstraints zu zusätzlichem (exponentiellen) Aufwand des Suchraums und der Beweisläange führen. Ob der Einsatz solcher Constraints sinnvoll ist oder nicht hängt von zweierlei ab: Von der verwendeten A-Ordnung $\prec_A$ und den Problemen, welche mit $\prec_A$-geordneten Tableaus behandelt werden sollen. Näheres hierzu findet sich in Kapitel 5. Nachdem wir im nächsten Abschnitt einige Begriffe im Zusammenhang mit Constraints klären, wenden wir uns in Abschnitt 4.2 A-geordneten Tableaus mit Constraints zu. Die Erzeugung von Constraints und deren Form stehen in den Abschnitten 4.3 und 4.4 im Vordergrund. In Abschnitt 4.5 erörtern wir einige Methoden, um die Erfüllbarkeit von Ordnungsconstraints zu testen. Wir beschließen dieses Kapitel mit einigen Anmerkungen zu Heuristiken, die in Rahmen A-geordneter Tableaus von Interesse sind.

4.1. Vorbemerkungen

In diesem Abschnitt führen wir einige Begriffe und Schreibweisen ein, welche die Beschreibung später auftretender Constraints erleichtern.

DEFINITION 4.1. $\prec$ sei eine Ordnungsrelation. Ein *Constraint* c ist ein prädikatenlogischer Ausdruck, der $\prec$ und $\doteq$ als einzige Prädikatensymbole enthält.

Ein *elementarer Constraint* ist ein Literal von der Form

$$x \doteq y, \neg(x \doteq y), x \prec y, \neg(x \prec y) \ .$$

Disjunktiv verknüpfte elementare Constraints werden auch als Klauselmengen geschrieben, wie z.B. $\{x \doteq y, \neg(a \doteq z), f(a) \prec x\}$.

DEFINITION 4.2. $s \doteq t$ ist erfüllbar, genau dann wenn s und t unifizierbar sind.

$\doteq$ wird als *syntaktische Gleichheit* bezeichnet.

Substitutionen $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ identifizieren wir mit dem syntaktischen Gleichheitsconstraint $(x_1 \doteq t_1 \wedge \ldots \wedge x_n \doteq t_n)$. Deswegen unterscheiden wir später nicht mehr genau zwischen einer Substitution und einem *erfüllbaren* syntaktischen Gleichheitsconstraint obiger Form.

Die Relation $\prec$ ist mit der dem Kalkül zugrunde liegenden A-Ordnung $\prec_A$ identisch. Die Betrachtungen über die Erfüllbarkeit von $\prec$-geordneten Constraints gelten aber auch, wenn $\prec$ keine A-Ordnung ist.

BEISPIEL 4.1. Der Constraint $(f(a) \prec g(x) \wedge y \doteq c) \vee z \prec b$ ist erfüllbar, falls $f(a) \prec g(a), b \prec c$ und $a \prec b$ gilt.

$x \doteq a \wedge y \doteq f(u)$ ist ein zur Substitution $\{f(x) \leftarrow f(a), y \leftarrow f(u)\}$ zugehöriger syntaktischer Gleichheitsconstraint. Ebenso $f(x) \doteq f(a) \wedge y \doteq f(u)$.

Weil $x \doteq a \wedge x \doteq b$ nicht erfüllbar ist, existiert dazu auch keine Substitution.

Da bei der formalen Definition geordneter Tableaus mit Constraints Ordnungsbedingungen uniform auftreten, verwenden wir folgende Abkürzungen.

DEFINITION 4.3. M sei eine Menge von Literalen und A sei ein Atom.

Zu einem gegeben Literal L sei $atom(L)$ das Atom von L, also: $atom(A) = A$ und $atom(\neg A) = A$.

$$\max_{\prec}(A, M) := \bigwedge_{B \in M} A \not\prec atom(B) \ .$$

Für eine Formel ϕ sei

$$\max_{\prec}(A, \phi) := \bigvee_{M \in D_\phi} \max_{\prec}(A, M)$$

und

$$\max_{\prec}(M, \phi) := \bigvee_{A \in M} \max_{\prec}(A, \phi) \ .$$

$\max_{\prec}(A, M)$ ist genau dann erfüllbar, wenn A maximal bezüglich $\prec$ in M ist. $\max_{\prec}(A, \phi)$ ist genau dann erfüllbar, wenn A maximal in einem disjunktiven Pfad von ϕ vorkommt und $\max_{\prec}(M, \phi)$ ist genau dann erfüllbar, wenn ein Literal aus M maximal in einem disjunktiven Pfad von ϕ vorkommt.

BEMERKUNG 4.1. Die Constraints aus Def. 4.3 sind Hornformeln.

BEWEIS. $\max_{\prec}(A, M)$ ist von der Form $A \not\prec B_1 \wedge \ldots \wedge A \not\prec B_n$.

Da alle weiteren Constraints nur mit Konjunktionen und Disjunktionen aus Constraints der Form $\max_{\prec}(A, M)$ aufgebaut werden, ist deren konjunktive Normalform eine Menge von Hornformeln. □

4.2. Geordnete Tableaus mit Constraints

4.2.1. Vollständige Ordnungseinschränkung. Betrachten wir noch einmal die Tableauregeln für geordnete Tableaus (Tabelle 3.6, Seite 19). Wird ein Ast A eines Tableaus T mit einer Formel ϕ aus einer Formelmenge Φ erweitert, so darf dies nur geschehen, wenn ϕ eine maximale Konnektion in A besitzt. Soll sichergestellt werden, daß während des späteren Beweisverlaufs diese Bedingung erhalten bleibt, so müssen zu den betrachteten Konnektionen Ordnungsconstraints erzeugt werden. Diese hängen von ϕ, dem betrachteten Ast A und der Formelmenge Φ ab. Wir wollen $c(\phi, A, \Phi)$ für diesen Constraint schreiben, ohne näher auf dessen Form einzugehen.

Der globale Constraint c_T eines Tableaus, welcher sicherstellt, daß das Tableau noch A-geordnet ist, wird bei jeder maximalen Konnektion mit $c(\phi, A, \Phi)$ konjunktiv verknüpft. Ist c_T unerfüllbar, so ist das Tableau nicht mehr A-geordnet.

Wenn die Formeln nicht in Klauselnormalform sind, dann treten bei dieser Vorgehensweise allerdings einige Probleme auf.

Der Constraint c_T enthält sehr viele redundante Informationen. Betrachte zum Beispiel Abbildung 4.4: Der Ast A sei durch eine α-Anwendung auf ψ aus A' hervorgegangen und F sei die Menge aller Formel, die sowohl in A als auch A' vorkommen. Wenn ϕ zur weiteren Expansion ausgewählt wird, dann muß der Ordnungsconstraint $c(\phi, A, \Phi)$ erzeugt werden. Dieser enthält als Teilconstraint $c(\phi, F, \Phi)$. Aber da die disjunktiven Pfade von ϕ Teilmenge der disjunktiven Pfade von ψ sind, ist $c(\phi, F, \Phi)$ schon im Constraint $c(\psi, A', \Phi)$ berücksichtigt worden. Deswegen ist es ausreichend, statt $c(\phi, A, \Phi)$ nur $c(\phi, A \Leftrightarrow F, \Phi)$ zu berechnen. Wird dies nicht berücksichtigt, dann sollte der Erfüllbarkeitstest die Redundanzen erkennen und entfernen.

Wie die Erfahrung mit der Berechnung von Konnektionen zeigt, wäre es in der Praxis zu aufwendig für alle möglichen Formeln ϕ, die während des Tableaubeweisens auftreten können, alle Ordnungsconstraints vorzuberechnen. Deswegen wollen wir uns nur mit einer weniger einschränkenden Version A-geordneter Tableaus mit Constraints befassen.

4.2.2. Abgeschwächte Ordnungseinschränkung. In Tabelle 4.2 sind die Tableauregeln für einen abgeschwächten A-geordneten Tableaukalkül mit

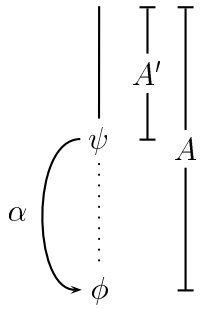


ABBILDUNG 4.4. Redundante Berechnung von Constraints (siehe Text)

γ	$\gamma\{x \leftarrow y\}$
$(\forall x)(\phi(x))$	$\neg\phi(y)$
$\neg(\exists x)(\phi(x))$	$\phi(y)$

wobei y neue freie Variable ist.

TABELLE 4.1. Abweichende Darstellung von Formeltypen

Constraints aufgeführt. Im Gegensatz zur bisherigen Schreibweise von Tableaus, ist $\cup$ die disjunkte mengentheoretische Vereinigung. Das heißt, es gilt $T \cup \{A\} \neq T$.

Es sind zwei weitere Komponenten explizit aufgeführt:

- Eine Menge Φ von Formeln. Anstatt wie bisher alle Formeln zum Tableau hinzuzufügen, ist diese Operation explizit als Regel formuliert (Holen).
- Ein Constraint c. Solange dieser erfüllbar ist, ist sichergestellt, daß jede Formel aufgrund einer maximalen Konnektion zum Tableau geholt wurde.

Es werden wieder α, β, δ und γ-Formeln unterschieden (siehe Tabelle 3.5, Seite 17). Da wir später die Substitution σ zu einer γ-Regel benötigen, ist $\sigma = \{x \leftarrow y\}$ explizit angegeben (Tabelle 4.1).

Ordnungsconstraints werden nur dann erzeugt, wenn eine neue Formel aufgrund einer maximalen Konnektion zum Tableau geholt wird. Weitere Ordnungseinschränkungen gibt es nicht. Dieser Kalkül hat den Vorteil, erheblich weniger Constraints zu erzeugen als eine Variante, welche garantiert, daß jede Grundinstanz des Tableaus A-geordnet ist. Letzteres ist allerdings auch bei unserem Verfahren der Fall, wenn Φ in Klauselnormalform ist und auch bei γ-Expansion Constraints erzeugt werden.

4.3. Erzeugen von Constraints

Wird eine Formel $\phi \in \Phi$ auf einen Tableauast A geholt, so verlangen die Regeln in Tabelle 4.2, daß dann auch der zu ϕ gehörige Constraint vorhanden ist. Wir wollen diese Art der Constrainterzeugung *statisch* nennen.

Init

$$\frac{\Phi}{\emptyset, \Phi, \text{true}}$$ zu einer gegebenen Formelmenge Φ.

Abschluß

$$\frac{T \cup \{A \cup \{L, L'\}\}, \Phi, c}{\{T - (A \cup \{L, L'\})\}\sigma, \Phi, c \wedge \sigma}$$ wobei c erfüllbar ist und $L\sigma = \overline{L'}\sigma$ für einen allgemeinsten Unifikator σ.

Holen 1

$$\frac{T \cup \{A \cup \{L\}\}, \Phi \cup \{\phi\}, c}{\{T \cup \{A \cup \{L, \phi\}\}, \Phi, c'}$$ wobei c erfüllbar ist, L' maximal in D_ϕ mit $L\sigma = \overline{L'}\sigma$ für einen allgemeinsten Unifikator σ und $c' = c \wedge \max_{\prec_A}(L', \phi) \wedge \sigma$.

Holen 2

$$\frac{T \cup \{A\}, \Phi \cup \{\phi, \psi\}, c}{\{T \cup \{A \cup \{\phi\}\}, \Phi \cup \{\psi\}, c'}$$ wobei c erfüllbar ist, L' maximal in D_ϕ, L maximal in D_ψ mit $L\sigma = \overline{L'}\sigma$ für einen allgemeinsten Unifikator σ und $c' = c \wedge \max_{\prec_A}(L', \phi) \wedge \max_{\prec_A}(L, \psi) \wedge \sigma$.

α-Expansion

$$\frac{T \cup \{A \cup \{\alpha\}\}, \Phi, c}{\{T \cup \{A \cup \{\alpha_1, \alpha_2\}\}, \Phi, c}$$ wobei c erfüllbar ist.

β-Expansion

$$\frac{T \cup \{A \cup \{\beta\}\}, \Phi, c}{\{T \cup \{A \cup \{\beta_1\}, A \cup \{\beta_2\}\}, \Phi, c}$$ wobei c erfüllbar ist.

δ-Expansion

$$\frac{T \cup \{A \cup \{\delta\}\}, \Phi, c}{\{T \cup \{A \cup \{\delta(t)\}, \Phi, c}$$ wobei c erfüllbar ist.

γ-Expansion

$$\frac{T \cup \{A \cup \{\gamma\}\}, \Phi, c}{\{T \cup \{A \cup \{\gamma, \gamma\{x \leftarrow y\}\}, \Phi, c}$$ wobei c erfüllbar ist.

TABELLE 4.2. Geordnete Tableaus mit Constraints.

Es ist aber auch denkbar, erst während der Expansion von ϕ den Ordnungsconstraint schrittweise zu berechnen. Diese Möglichkeit sei *dynamische* Constrainterzeugung genannt.

Beide Ansätze besitzen Vor- und Nachteile, die wir in den beiden nächsten Abschnitten erörtern. Tabelle 4.3 zeigt eine knappe Gegenüberstellung der Argumente.

4.3.1. Statische Erzeugung. Um eine Konnektion zwischen zwei Formeln ϕ und ψ im voraus berechnen zu können, werden die disjunktiven Pfade D_ϕ und D_ψ benötigt. Mit diesen kann für jedes Literal $L \in D_\phi$ der Constraint $\max_\prec(L, \phi)$ erzeugt werden. Deswegen sollten die Constraints idealerweise zusammen mit den Konnektionen berechnet werden. Aufgrund der Substitutivität der A-Ordnung ist sichergestellt, daß während das Tableaubeweisens keine neuen maximalen Konnektionen entstehen können.

Diese Vorgehensweise hat allerdings einen gravierenden Nachteil. Variablen, die in der Vorberechnung in D_ϕ auftreten, sind nicht identisch mit denen,

statisch	dynamisch
+ Nur einmaliges Erzeugen pro Formel	− Mehrfache Erzeugung bei γ-Anwendung
+ In der Vorberechnung kann ein besserer (und langsamerer) Test für die Erfüllbarkeit von Constraints benutzt werden	− Erzeugen kostet Aufwand, auch falls der Constraint immer erfüllbar ist
+ Erzeugung bei Link-Berechnung einfach	− Für effiziente Berechnung besondere Datenstrukturen notwendig
− Constraints kommen evtl. zu früh auf den Ast	+ Constraints brauchen erst dann erzeugt werden, wenn es sinnvoll ist
− Erzeugen für nicht benötigte Formeln	+ Constraints nur für gebrauchte Formeln
− Speicher- und Zeitaufwand bei Vorberechnung evtl. zu groß	+ Zeit- und Speicheraufwand hängt von verwendeten Formeln ab
− Problem der Variablenidentifizierung	+ Variablen von Constraint und Formel identisch

TABELLE 4.3. Vergleich statischer und dynamischer Constrainterzeugung

welche später bei der Expansion von ϕ während des Tableaubeweisens auftreten. Die zugehörigen Variablen müssen aber identifiziert werden, da sonst der vorberechnete Constraint variablendisjunkt zum Tableau ist.

Neue Variablen werden nur bei einer γ-Expansion eingeführt. Da γ durch eindeutiges Anwenden der Tableauregeln auf eine Formel $\phi \in \Phi$ erzeugt wurde, kann diese Ableitungsfolge benutzt werden, um neu eingeführte Variablen eindeutig zu identifizieren.

BEISPIEL 4.2. Die gebundene Variable x in $\phi = A \wedge (B \vee (\forall x C))$ kann durch die Folge $\alpha_1, \beta_2, \gamma$ eindeutig in ϕ identifiziert werden.

Wenn zu jeder Variable z aus dem Constraint $c(\phi, A, \Phi)$ diese Folge bekannt ist, muß der Unifikator $\{z \leftarrow y\}$ auf c_T angewendet werden, wenn y während des Tableaubeweisens mit genau der gleichen Folge aus ϕ erzeugt wurde. Danach stimmen die freien Variablen überein.

Hier wirkt sich die verwendete δ^{+^+}-Regel günstig aus: Skolemkonstanten und -funktionen unterscheiden sich im vorberechneten Constraint nicht von denen der später expandierten Formel ϕ. Das ist ein Vorteil von statischen Constraints, denn diese Symbole können den Constraint schon von Beginn an so stark einschränken, daß er unerfüllbar ist.

BEISPIEL 4.3. Betrachte die beiden Formeln

$$\phi_1 = \exists x(P(x) \vee Q(f(a))) \text{ und } \phi_2 = \forall y \neg P(y) \ .$$

Wird die δ^{+^+}-Regel verwendet, so ist

$$D_{\phi_1} = \{P(sko) \vee Q(f(a))\} \ .$$

δ	$\delta(t)$
$\neg(\forall x)(\phi(x))$	$\neg\phi(t)$
$(\exists x)(\phi(x))$	$\phi(t)$

wobei $t = f(x_1, \ldots, x_n)$ ist mit einer (Skolem-)Funktion f, die neu auf dem betrachten Ast A ist und $x_1, \ldots, x_n$ sind alle freien Variablen von A.

TABELLE 4.4. Alte Fittingsche δ-Regel

Es existieren keine $\prec_\tau$ geordneten Konnektionen zwischen den beiden Formeln. Wird allerdings die in (Fitting, 1990) gegeben δ-Regel verwendet (siehe Tabelle 4.4), so hängt es von der Anzahl Variablen auf dem betrachteten Tableauast A ab, ob ϕ_1 $\prec_\tau$-maximale Konnektion zu ϕ_2 besitzt oder nicht. Enthält A keine Variablen, so wird x durch eine Konstante ersetzt. Es existiert wie bei der δ^{++}-Regel keine maximale Konnektion zwischen ϕ_1 und ϕ_2. Gibt es hingengen Variablen in A, so wird x durch eine Skolemfunktion ersetzt. Durch die größere Termtiefe wird das Atom maximal in ϕ_1 und besitzt deswegen eine maximale Konnektion zu ϕ_2.

Ein Erfüllbarkeitstest für Ordnungsconstraints kann recht aufwendig sein. In diesem Fall wird man bei einer Implementierung mit einfacheren, weniger restriktiven Tests auskommen müssen. Aber zumindestens in der Vorverarbeitung ist es sinnvoll, bessere Algorithmen zu benutzen, wenn zu erwarten ist, daß viel mehr Constraints als immer erfüllbar oder unerfüllbar erkannt werden können. Diese verursachen während des Tableaubeweisens keinen zusätzlichen Aufwand.

Wird nur eine geringe Anzahl von Formeln aus Φ tatsächlich für einen Tableaubeweis benutzt, so wurden die zugehörigen Constraints umsonst vorberechnet. Es kann auch passieren, daß eine Formel $\phi \in \Phi$ zu einem Tableauast A geholt aber nicht weiter expandiert wird. In diesem Fall ändert sich die Erfüllbarkeit des Constraints c zu ϕ evtl. nicht mehr. Dies ist zum Beispiel der Fall, wenn alle Variablen in c nur noch gebunden vorkommen. Leider ist es in der Praxis nicht möglich, c vom globalen Constraint c_T zu entfernen, da ein Erfüllbarkeitstest c_T üblicherweise so stark verändert (etwa durch eine Normalformtransformation), daß auf c nicht mehr explizit zugegriffen werden kann.

4.3.2. Dynamische Erzeugung. Eine Art dynamisch Constraints zu erzeugen brauchen wir nicht näher betrachten: wird eine Formel ϕ zum Tableau geholt, so kann derselbe Algorithmus, der für eine statische Berechnung benutzt wird, auch für ϕ verwendet werden. Dies unterscheidet sich aber dann nicht mehr von statischen Constraints. Vorteilhaft ist, daß nur Constraints von solchen Formeln Aufwand verursachen, welche auch wirklich zum Tableau geholt werden.

Eine andere Möglichkeit besteht darin, während der Expansion einer Formel ϕ den zugehörigen Ordnungsconstraint $c(\phi, A, \Phi)$ schrittweise zu berechnen. Wenn ϕ vollständig expandiert ist, dann sind auch alle disjunktiven Pfade durch ϕ bekannt. Erst dann ist es möglich, $c(\phi, A, \Phi)$ vollständig zu berechnen.

Im schlechtesten Fall muß der Ast A auf den ϕ geholt wurde geschlossen sein, um festzustellen, ob ϕ überhaupt eine maximale Konnektion in A besitzt.

Um dies zu vermeiden, sollte die Information, ob ein ϕ aus Φ eine maximale Konnektion zu einem bestimmten Literal hat oder nicht, vorberechnet werden. Weil dabei D_ϕ berechnet werden muß, ist dies allerdings fast genauso zeitaufwendig, wie die Vorberechnung der Constraints $\max_\prec(L, \phi)$.

Diese Nachteile zeigen deutlich, daß es nicht sinnvoll ist, Ordnungsconstraints zu maximalen Konnektionen erst während des Tableaubeweisens zu generieren.

Trotzdem seien noch zwei Vorteile gegenüber statischen Constraints erwähnt:

- Das Problem der Variablenidentifizierung gibt es bei dynamischen Constraints offensichtlich nicht, da die disjunktive Pfade von ϕ nur einmal berechnet werden.
- Es werden Constraints nur für Formeln erzeugt, die auch wirklich auf einen Tableauast geholt werden. Weiterhin werden für nicht expandierte Teile einer Formel keine Constraints erzeugt.

4.4. Darstellung der Constraints

Wird eine Formel $\phi \in \Phi$ aufgrund einer maximalen Konnektion auf einen Ast A geholt, so besitzt ϕ wahrscheinlich mehrere verschiedene maximale Konnektionen in A und auch in Φ (bei einem Neustart). Da diese Konnektionen es rechtfertigen, ϕ auf A zu holen, müssen für eine vollständige Implementierung alle Alternativen berücksichtigt werden. Dazu kommen die beiden folgenden Methoden in Betracht.

4.4.1. Backtracking über Auswahl von Konnektionen. Am einfachsten ist es, die indeterministische Auswahl der maximalen Konnektionen von Formeln ϕ in A oder in Φ mit Hilfe von Backtracking zu implementieren. Bei diesem Verfahren entsteht allerdings eine große Anzahl von Choicepoints. Diese machen sich beim Tableaubeweisen nachteilig bemerkbar, da sie den Suchraum stark erhöhen, wie folgendes Beispiel zeigt:

Beispiel 4.4. In Abbildung 4.5 gibt es n verschiedene Konnektionen. Bei einer Grundordnung $a_1 < \ldots < a_n$ sind die Konnektionen alle $\prec_L$-maximal. Wird die Konnektion $\langle P(x), P(a_n), \{x \leftarrow a_n\}\rangle$ erst zu allerletzt ausgewählt, so muß der Beweiser bis dahin $n \Leftrightarrow 1$ mal zurücksetzen.

Der Constraint c_T zu einem Tableau T hat bei dieser Methode immer die folgende Form:

$$\overbrace{(\bigwedge_i \sigma_i)}^{e} \wedge \overbrace{(\bigwedge_j \max_\prec(L_j, \phi_j)}^{o_1} \wedge \overbrace{(\bigwedge_k \max_\prec(L_k, \phi_k) \wedge \max(L'_k, \psi_k))}^{o_2} .$$

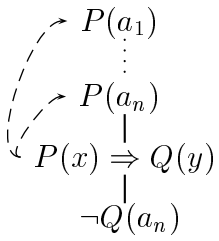


ABBILDUNG 4.5. Erhöhung der Choicepoints bei Backtracking über Konnektionsauswahl

e ist ein rein syntaktischer Gleichheitsconstraint, der aus den Konnektionsunifikatoren gebildet wurde. o_1 enthält alle Constraints zu maximalen Konnektionen in einem Tableauast und o_2 enthält alle Constraints zu maximalen Konnektionen zu Formeln $\psi_k \in \Phi$.

Da c_T in einen reinen syntaktischen Gleichheitsconstraint e und einen reinen Ordnungsconstraint $o_1 \wedge o_2$ zerfällt, können beide getrennt auf Erfüllbarkeit getestet werden. Falls eine neue Ordnung $\prec'$ statt $\prec$ benutzt wird, so braucht der Erfüllbarkeitstest nur für die $\prec'$-Ordnungsconstraints geändert zu werden.

Der Erfüllbarkeitstest für e kann entfallen, wenn man die Konnektionsunifikatoren σ_i sofort auf das Tableau anwendet.

4.4.2. Eliminieren der Choicepoints. Wenn ein $\phi \in \Phi$ auf einen Tableauast A geholt wird, können alle dabei auftretenden maximalen Konnektionen zu einem einzigen Constraint $c(\phi, A, \Phi)$ zusammengefaßt werden. Es ist dann nicht mehr notwendig, über die Konnektionsauwahl zurückzusetzten. Bei diesem Verfahren entstehen keine zusätzlichen Choicepoints.

$c(\phi, A, \Phi)$ ist dabei von folgender Form:

$$(\overbrace{\bigvee_{\psi_i \in \Phi} (\sigma_i \wedge \max_{\prec}(\phi, \psi_i))}^{o_1}) \vee (\overbrace{\bigvee_{L_j \in A} (\sigma_j \wedge \max_{\prec}(L_j, \phi))}^{o_2}) \ .$$

Der Constraint o_1 beschreibt alle maximalen Konnektionen von ϕ zu anderen Formeln in Φ und o_2 beschreibt alle maximalen Konnektionen von ϕ in den Ast A. Der globale Constraint c_T eines Tableaus ist die Konjunktion von allen $c(\phi, A, \Phi)$.

Da in $c(\phi, A, \Phi)$ syntaktische Gleichheitsconstraints und Ordnungsconstraints verschränkt sind, können diese nicht mehr getrennt behandelt werden. Wenn aber etwas von der Ordnungseinschränkung aufgegeben wird, kann man c_T trotzdem in zwei unabhängige Teile aufspalten:

Offenbar ist $c(\phi, A, \Phi)$ genau dann erfüllbar, wenn

$$\sigma_{i'} \wedge \max_{\prec}(\phi, \psi_{i'}) \text{ oder } \sigma_{j'} \wedge \max_{\prec}(L_{j'}, \phi))$$

für ein i' oder j' erfüllbar ist. Dann ist auch der Constraint

$$(\bigvee_i \sigma_i \vee \bigvee_j \sigma_j) \wedge (\bigvee_i \max_{\prec}(\phi, \psi_i) \vee \bigvee_j \max_{\prec}(L_j, \phi_j))$$

erfüllbar. Mit Hilfe dieser Transformation zerfällt der globale Constraint c_T wieder in einen rein syntaktischen Gleichheitsconstraint und in einen reinen Ordnungsconstraint. Wir wollen diese Art der Erfüllbarkeit von c_T *schwache Erfüllbarkeit* nennen.

Die Umkehrung gilt natürlich *nicht*, da $c(\phi, A, \Phi)$ unerfüllbar sein kann, obwohl der zugehörige schwach erfüllbare Constraint erfüllbar ist.

Beispiel 4.5. Bei einer Grundordnung $a < b < c$ ist der Constraint

$$(x \doteq a \wedge \neg f(x) \prec_L f(b)) \vee (x \doteq b \wedge \neg f(x) \prec_L f(c))$$

offenbar unerfüllbar, aber schwach erfüllbar, da

$$(x \doteq a \vee x \doteq b) \wedge (\neg f(x) \prec_L f(b) \vee \neg f(x) \prec_L f(c))$$

erfüllbar ist.

Für die Vollständigkeit A-geordneter Tableaus mit Constraints ist dies zum Glück nicht von Belang.

Es stellt sich allerdings die Frage, wieviel von der Einschränkung des Kalküls durch die Verwendung der schwachen Erfüllbarkeit verloren geht. Offensichtlich hängt dies von der möglichen Anzahl verschiedener maximaler Konnektionen ab, die es rechtfertigen eine bestimmte Formel auf den aktuellen Ast zu holen. Je größer diese Anzahl ist, desto mehr Disjunktionen enthält der schwach erfüllbare Constraint und desto kleiner ist die Einschränkung.

4.5. Erfüllbarkeitstests

Wir erörtern hier einige Methoden, um die Erfüllbarkeit von Constraints zu testen. Dabei interessieren uns vor allen Dingen Ordnungsconstraints, da syntaktische Gleichheitsconstraints leicht zu behandeln sind (Abschnitt 4.5.1)

Einen effizienten, uniformen Erfüllbarkeitstest für Ordnungsconstraints zu beliebigen Ordnungen gibt es leider nicht. Dazu sind die Eigenschaften verschiedener Ordnungen zu unterschiedlich. Ein Test, der nur die Irreflexivität, Transitivität und Substitutivität ausnutzt, würde zu keinem befriedigenden Verfahren führen.

Bei manchen Erfüllbarkeitstest wird die Signatur zu einem gegebenen Problem erweitert, um so einige kritische Fälle zu umgehen.

Beispiel 4.6. Gibt es in der gegebenen Signatur kein Symbol b mit $a < b$ und $b < c$, dann ist der Constraint $a \prec_L x \wedge x \prec_L c$ unerfüllbar.

Es ist meist recht schwierig, solche Situationen ausfindig zu machen. Bei einer Signaturerweiterung nimmt man einfach an, daß solch ein Term immer existiert. Dies hat natürlich zur Folge, daß Constraints oftmals nicht als unerfüllbar erkannt werden, wenn sie es sind.

4.5.1. Syntaktische Gleichheitsconstraints. Bei A-geordneten Tableaus werden syntaktische Gleichheitsconstraints immer aufgrund einer Substitution erzeugt. Wir beschränken uns deswegen im folgenden auf syntaktische Gleichheitsconstraints, in denen nur positive Literale auftreten.

Die Erfüllbarkeit von e kann mit Unifikationsalgorithmen entschieden werden (vergleiche Def. 4.2).

Verfahren, die auf dem Algorithmus von (Martelli & Montanari, 1982) basieren, haben theoretisch einen fast linearen Zeitaufwand und scheinen deswegen für eine Erfüllbarkeitstest gut geeignet zu sein. In der Praxis lohnt sich deren Einsatz aber nur, falls die dabei benutzten Datenstrukturen einfach wiederverwendet werden können, wenn der Constraint erweitert oder eine Substitution darauf angewendet wird. Weniger aufwendige Verfahren, die in der Praxis gute Ergebnisse liefern, bekommt man durch Modifikationen des Algorithmus von Robinson (Robinson, 1965b).[1]

BEMERKUNG 4.2. Ein Konjunkt $x_1 \doteq t_1 \wedge \ldots \wedge x_n \doteq t_n$ ist genau dann erfüllbar, wenn $(x_1, \ldots, x_n)$ mit $(t_1, \ldots, t_n)$ unifizierbar ist.

Ist der zu behandelnde syntaktische Gleichheitsconstraint e keine Konjunktion positiver, elementarer syntaktischer Gleichheitsconstraints, dann kann zu e eine erfüllbarkeitsäquivalente disjunktive Normalform $e' = \vee_i \wedge_j e_{ij}$ berechnet werden.[2] Diese ist genau dann erfüllbar, wenn ein Konjunkt $\wedge_j e_{ij}$ erfüllbar ist. Syntaktische Gleichheitsconstraints, wie sie bei A-geordenten Tableaus auftreten, lassen sich deswegen mit Unifikation behandeln. Es handelt sich dabei um ein Entscheidungsverfahren.

4.5.2. Spezielle Erfüllbarkeitstests. In vielen Fällen können die speziellen Eigenschaften einer Ordnung ausgenutzt werden, um einen Erfüllbarkeitstest zu entwickeln. Ein solches Verfahren für die lexikographische Pfadordnung wird in (Nieuwenhuis & Rubio, 1995) vorgestellt. Dieser Test geht von einer Signaturerweiterung aus.

4.5.3. Einsatz automatischer Beweiser. Für einige Ordnungen kann ein Ordnungsconstraint in ein erfüllbarkeitsäquivalentes Problem der Prädikatenlogik umgeformt werden. Dieses kann dann mit einem automatischen Beweiser behandelt werden.

Wir wollen die Transformation in ein prädikatenlogisches Problem anhand der lexikographischen Ordnung $\prec_L$ veranschaulichen. Die Methode funktioniert aber auch mit andere Ordnungen, wie z.B. der lexikographische Pfadordnung, $\prec_\tau$ und $\prec_L$. Wir unterscheiden im folgenden nicht zwischen den Symbolen der Ordnungsconstraints und denen des prädikatenlogischen Problems.

Die Transformation τ_L von der Menge der $\prec_L$-Ordnungsconstraints in die Menge prädikatenlogischer Formeln ist recht einfach:

DEFINITION 4.4. c sei ein $\prec_L$ Ordnungsconstraint.

$$\tau_L(c) := c \ .$$

[1] Welche von diesen Methoden in unserem Zusammenhang die sinnvolleren sind, werden wir hier nicht weiter erörtern, da eine gründliche Darstellung zu umfangreich wäre und ein Erfüllbarkeitstest in der Implementierung keine Verwendung fand.

[2] Allerdings kann die DNF zu e einen exponentiellen Speicheraufwand verursachen.

Obige Schreibweise ist gewollt suggestiv. Es handelt sich bei τ_L aber *nicht* um die Identität auf der Menge der $\prec_L$-Constraints! Definitions- und Wertebereich von τ_L sind verschieden.

Wir geben eine universell abgeschlossene prädikatenlogische Formelmenge $M_{\prec_L}$ an. Diese stellt sicher, daß $\prec_L$ im transformierten Problem genau wie die entsprechende A-Ordnung interpretiert wird.

DEFINITION 4.5. Zu gegebener Signatur und Ordnung $<$ auf Funktions- und Prädikatensymbolen enthält $M_{\prec_L}$ folgende Formeln:

1. $\{(\forall x)\neg(x \prec_L x), (\forall x,y,z)(x \prec_L y \wedge y \prec_L z \Rightarrow x \prec_L z)\} \subset M_{\prec_L}$.
2. Zu jedem 0-stelligen Funktions- und Prädikatensymbol s und t mit $s < t$ gelte
$$s \prec_L t \in M_{\prec_L} \ .$$
3. Zu jedem n-stelligen Funktions- und Prädikatensymbol f und g mit $f < g$ gelte

$$\{f < g \Rightarrow ((\forall x_1, \ldots, x_n, y_1 \ldots, y_n)(f(x_1, \ldots, x_n) \prec_L g(y_1, \ldots, y_n)))\} \subset M_{\prec_L}$$

und

$$\{(\forall x_1, \ldots, x_n, y_1, \ldots, y_n)(x_1 \prec_L y_1 \wedge \ldots \wedge x_n \prec_L y_n \Rightarrow f(x_1, \ldots, x_n) \prec_L f(y_1, \ldots, y_n))\} \subset M_{\prec_L} \ .$$

4. Nichts anderes ist in $M_{\prec_L}$.

Nach Definition von $M_{\prec_L}$ gilt folgende Aussage:

BEMERKUNG 4.3. $<$ sei eine Ordnung auf der Menge der Terme und Prädikatensymbole. Ein $\prec_L$-Ordnungsconstraint c ist genau dann erfüllbar, wenn $\{\tau_L(c)\} \cup M_{\prec_L}$ erfüllbar ist.

Die Analogie dieser Methode zu Verfahren der Theoriebehandlung (Abschnitt 3.5) liegt auf der Hand. Auch dort garantiert eine universell abgeschlossene Formelmenge die korrekte Semantik bestimmter Prädikate.

Ein Erfüllbarkeitstest von $\prec_L$-geordneten Constraints kann nach Bemerkung 4.3 auf einen Erfüllbarkeitstest von $\tau_L(c) \cup M_{\prec_L}$ zurückgeführt werden. Dazu können beliebige für die Prädikatenlogik erster Stufe vollständige Beweiser benutzt werden. Leider führt dieses Methode i.allg. zu keinem Entscheidungsverfahren, da für beliebige prädikatenlogische Formeln nicht entscheidbar ist, ob sie erfüllbar sind oder nicht. Das so gewonnene Verfahren muß also zeitlich beschränkt werden. Diese Zeitschranke wählt man in der Praxis so, daß möglichst viele unerfüllbare Constraints erkannt werden können.

Zu einer gegeben Ordnungsrelation wird nur eine bestimmte eingeschränkte Problemklasse erzeugt. Deswegen ist es evtl. möglich, mit bestimmten Beweiskalkülen einen effizienten Algorithmus oder gar ein Entscheidungsverfahren zu realisieren.

In unserem Fall fällt auf, daß $M_{\prec_L}$ nur aus Hornformeln besteht. Dies sind Klauseln, in denen höchstens ein positives Literal vorkommt. Da τ_L die Struktur von c nicht ändert, ist $\tau_L(c)$ nach Bemerkung 4.1 selbst auch in Hornform.

Deswegen können speziell für Probleme in Hornform geeignete Beweiskalküle für einen Erfüllbarkeitstest genutzt werden.

Im vorliegenen Fall bietet sich positive Hyperresulotion mit Einschränkung auf Einer-Klauseln an. Dieser Kalkül ist vollständig für Hornformeln. Da die Klauseln von $\tau_L(c)$ keine positiven Literale enthalten, ist die Anzahl der Resolutionsableitungen stark beschränkt. Zusätzlich wird durch die Einschränkung auf Einer-Resolventen garantiert, daß die Transitivität von $\prec_L$ zielgerichtet angewendet wird.

Basierend auf diesem Kalkül wird in (Baumgartner, 1996b) eine Methode vorgestellt, Horntheorien $\mathcal{T}$ automatisch in Hintergrundbeweiser zu transformieren. Mit einem Verfahren, das an eine Vervollständigung eines Gleichungssystems erinnert, wird $\mathcal{T}$ in eine erfüllbarkeitsäquivalente Menge $\mathcal{T}'$ transformiert. Danach kann zusätzlich verlangt werden, daß bei einem Resolutionsschritt mindestens eine der beteiligten Klauseln aus $\mathcal{T}'$ oder der Problemmenge stammt. Dies ist als Input-Einschränkung bekannt. Sie erlaubt eine effiziente PTTP-Implementierung (Stickel, 1988) des Beweiskalküls.

Leider funktioniert dies nur, wenn die Eingabe für den Hintergrundbeweiser eine Menge von Literalen ist. In unserem Fall sind dies leider Hornformeln. Das Verfahren ist aber wieder vollständig, wenn Resolutions- und Vervollständigungsschritte verschränkt werden (Baumgartner, 1996a). In diesem Fall braucht die Vervollständigung nicht mehr endlich zu sein.

4.5.4. Ganzzahlige Optimierung. In diesem Abschnitt zeigen wir, daß auch Methoden der ganzzahligen Optimierung (Schrijver, 1986) benutzt werden können, um Ordnungsconstraints verschiedener A-Ordnungen zu behandeln. Dieser Ansatz ist schon deswegen interessant, da in diesem Bereich aufgrund jahrzehntelanger Bemühungen viele effiziente Verfahren und Implementierungen existieren, um ganzzahlige Optimierungsprobleme zu lösen. Wir veranschaulichen dies an folgender A-Ordnung:

Definition 4.6. $\#$ sei eine Bewertung auf der Menge M aller Funktions- und Prädikatensymbole (vergleiche Def. 2.3).

Für ein Atom A sei $\#$ wie folgt auf der Menge der Atome und Terme fortgesetzt:

$$\#(A) = \sum_{f \in M} a_f \cdot \#(f), \text{ wobei } f \text{ genau } a_f \text{ mal in } A \text{ vorkommt.}$$

Für zwei Atome A und B gilt $A \prec_\# B$ genau dann, wenn

1. $\#(A) \leq \#(B)$ gilt,
2. $\mathbf{V}(A) \subseteq \mathbf{V}(B)$ und
3. in einen der beiden Bedingungen die Ungleichheit gilt.

Wenn Bedinung 2 fortgelassen wird, so bekommt man eine *stabile A-Ordnung* (vergl. stabile Auswahlfunktion Def. 3.29).

Im Grundfall vergleicht $\prec_\#$ die gewichtete Summe zweier Atome. Bedingung 2 garantiert die Substitutivität von $\prec_\#$ und mit Bedingung 3 wird $\prec_\#$

zu einer strikten Ordnung. $\prec_\#$ ist keine totale Ordnung auf der Menge aller Grundatomen, da $P(a,b)$ und $P(b,a)$ unvergleichbar sind.

Existiert eine totale Grundordnung $<$ auf der Menge aller Funktions- und Prädikatensymbole, dann kann $\prec_\#$ zu einer totalen Ordnung erweitert werden. Dazu muß Bedingung 3 in Definition 4.6 durch folgende induktive Bedingung ersetzt werden:

3. Für $A = f(t_1, \ldots, t_n)$ und $B = g(s_1, \ldots, s_m)$ gilt:
   (a) $f < g$ oder
   (b) $f = g$ und $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}, t_i \prec_\# s_i$ für ein $1 \leq i \leq \min(n,m)$.

Die so erhaltene Ordnung ist als erweiterte Knuth-Bendix-Ordnung bekannt (Peterson, 1983).

Ein elementarer Ordnungsconstraint $A \prec_\# B$ ist erfüllbar, wenn eine Grundsubstitution σ mit $\#(A\sigma) < \#(B\sigma)$ existiert. Dies ist genau dann der Fall, wenn

$$\sum_{f \in M} a_f \cdot \#(f_i) + \sum_{x \in \mathbf{V}(A)} a_x \cdot \#(x\sigma) < \sum_{f \in M} b_f \cdot \#(f_i) + \sum_{x \in \mathbf{V}(B)} b_x \cdot \#(x\sigma)$$

gilt. a_f, b_f, a_x und b_x bezeichnen die Anzahl der Vorkommen von Symbolen und Variablen in A bzw. B.

Ersetzen wir $\#(x\sigma)$ durch eine neue Variable x', so hat obige Ungleichung folgende Form:

$$z < \sum_{x'} c_x \cdot x' \ ,$$

wobei $z = \sum_{f \in M} a_f \cdot \#(f_i) \Leftrightarrow \sum_{f \in M} b_f \cdot \#(f_i)$ ist und die Variablen auf der rechten Seite zusammengefaßt sind. x' ist eine Variable, die einen positiven ganzzahligen Wert annimmt. Bei der Ungleichung handelt es sich also um ein ganzzahliges Optimierungsproblem mit Nebenbedingung $x' > 0$. Uns interessiert in diesem Zusammenhang nur die Lösung dieses Problems.

Wir wollen diese Abbildung eines elementaren $\prec_\#$-Ordnungsconstraints in ein Problem der ganzzahligen Optimierung mit ι bezeichnen. Eine bei dieser Transformation eingeführte ganzzahlige Variable x' entspricht eineindeutig einer Variablen $x \in \mathbf{V}(A) \cup \mathbf{V}(B)$. Wir bezeichen diese Bijektion ebenfalls mit ι. Zwischen den beiden verschiedenen Constraints besteht folgende Beziehung:

SATZ 4.1. *$A \prec_\# B$ sei ein elementarer erfüllbarer Ordnungsconstraint. Dann ist auch $\iota(A \prec_\# B)$ mit Nebenbedingungen $\iota(x) > 0$ für alle $x \in \mathbf{V}(A) \cup \mathbf{V}(B)$ erfüllbar.*

BEWEIS. Ist $A \prec_\# B$ erfüllbar, so gibt es eine Grundsubstitution σ, so daß $\#(A\sigma) < \#(B\sigma)$ gilt. Aus obiger Transformation ist ersichtlich, daß dann $x' = \#(x\sigma)$ den Constraint $\iota(A \prec_\# B)$ erfüllt. Da die Bewertung $\#(t)$ eines Terms t immer positiv ist, ist auch die Nebenbedingung erfüllt. □

Die Umkehrung von Satz 4.1 gilt nicht, da zu einer ganzzahligen Lösung x' einer Variablen $\iota(x)$ nicht unbedingt ein Term t existieren muß, so daß $\#(t) = x'$ ist.

BEISPIEL 4.7. f, g seien Funktionssymbole und a eine Konstante mit der Bewertung

$$\begin{aligned} \#f &= 2 \ , \\ \#g &= 2 \text{ und} \\ \#a &= 2 \ . \end{aligned}$$

Dann ist der Constraint $c = f(x) \prec_{\#} g(a)$ unerfüllbar, da jede Grundinstanz von $f(x)$ mit mindestens 4 bewertet wird.

Das zugehörige Problem $\iota(c)$ führt auf die Ungleichung $2 + \iota(x) < 4$ mit Nebenbedingung $\iota(x) > 0$. Dieses hat nur die einzige ganzzahlige Lösung $\iota(x) = 1$. Allerdings gibt es hierzu keinen Grundterm t mit $\#t = 1$.

Erlauben wir, daß die Signatur beliebig erweitert werden kann, so können wir einfach eine neue Konstante c einführen, welche die gewünschte Bewertung erhält. Damit garantieren wir, daß zu jeder positiven ganzzahligen Lösung des transformierten Problems auch ein entsprechender Grundterm existiert.

SATZ 4.2. *Kann die Signatur um beliebige neue Terme mit einer beliebigen Bewertung erweitert werden, dann ist $A \prec_{\#} B$ erfüllbar genau dann, wenn $\iota(A \prec_{\#} B)$ mit Nebenbedingungen $\iota(x) > 0$ für alle $x \in \mathbf{V}$ erfüllbar ist.*

Auch wenn eine Signaturerweiterung zu einem weniger restriktivem Erfüllbarkeitstest führt, wird dieser in der Praxis meist zufriedenstellende Ergebnisse liefern, falls genügend Funktions-, Konstanten- und Prädikatensymbole mit unterschiedlicher Bewertung vorhanden sind. Es ist allerdings auch möglich zu jeder ganzzahligen Variablen $\iota(x)$ einen Constraint zu erzeugen, so daß auch die Umkehrung von Satz 4.1 gilt. Eine Signaturerweiterung ist deshalb nicht notwendig.

Ein notwendiges Kriterium zur Existenz eines Terms t mit einer gewünschten Bewertung $\#t = z$ ist, daß es überhaupt eine Multimengen von Konstanten- und Prädikatensymbolen gibt, dessen Summe der einzelnen Bewertungen gleich z ist. Es muß also folgender Constraint erfüllbar sein (Funktions- und Konstantensymbole sind getrennt aufgeführt):

$$z = \sum_{f \in \mathbf{F}-\mathbf{K}} x_f \cdot \#f + \sum_{c \in \mathbf{K}} x_c \cdot \#c \ ,$$

wobei x_f und x_c neue ganzzahlige Variablen mit $x_f, x_c \geq 0$ sind.

Wir wollen nun mit Hilfe eines weiteren Constraints beschreiben, daß zu einer bestimmten Lösung obigen Problems ein Term t mit $\#t = z$ konstruiert werden kann. Ausgehend von einer Funktion f mit Stelligkeit $arity(f) = n$, müssen die n offenen Argumente von f „aufgefüllt“ werden. Wird dazu eine Konstante benutzt, so erniedrigt sich die Anzahl der offenen Argumente um eins; bei einer Funktion mit Stelligkeit k wird diese Anzahl um $k \Leftrightarrow 1$ erhöht. Das heißt, um einen Term aus einer Lösung der x_f und x_c konstruieren zu können, muß die Anzahl offener Argumente (produziert von den Funktionssymbolen) gleich der Anzahl der Konstantensymbole sein:

$$1 + \sum_{f \in \mathbf{F}-\mathbf{K}} x_f \cdot arity(f) = \sum_{c \in \mathbf{K}} x_c \ .$$

Wir haben also folgendes Ergebnis:

SATZ 4.3. *$A \prec_{\#} B$ sei ein elementarer erfüllbarer Ordnungsconstraint und M die Menge aller Funktions- und Prädikatensymbole von A und B.*

Dann ist $A \prec_{\#} B$ erfüllbar genau dann, wenn $\iota(A \prec_{\#} B)$ mit Nebenbedingungen

$$\iota(x) = \sum_{f \in \mathbf{F}-\mathbf{K}} x_f \cdot \#f + \sum_{c \in \mathbf{K}} x_c \cdot \#c$$

und

$$1 + \sum_{f \in \mathbf{F}-\mathbf{K}} x_f \cdot arity(f) = \sum_{c \in \mathbf{K}} x_c$$

für alle $x \in \mathbf{V}(A) \cup \mathbf{V}(B)$ erfüllbar ist. Dabei sind x_f und x_c neue ganzzahlige Variablen mit $x_f, x_c \geq 0$ für alle $f \in \mathbf{F} \Leftrightarrow \mathbf{K}, c \in \mathbf{K}$.

Da bei diesem Verfahren zusätzliche Constraints mit oftmals vielen neuen ganzzahligen Variablen entstehen, wird man in der Praxis besser eine Signaturerweiterung vornehmen. Das dabei entstehende Problem der ganzzahligen Optimierung ist wesentlich kleiner und kann deswegen meist schneller auf Erfüllbarkeit getestet werden.

Man erhält einen Erfüllbarkeitstest für nicht elementare Ordnungsconstraints c, indem die Abbildung ι so auf der Menge aller Constraints erweitert wird, daß die logischen Operatoren erhalten bleiben. Das resultierende Problem transformiere man in eine Negations-Normalform. Verbleibende negative elementare ganzzahligen Optimierungsproblemen $\neg L < R$ ersetzte man durch $L \geq R$. Das so erhaltende Problem ist dann im Sinne von Satz 4.2 erfüllbarkeitsäquivalent zu c.

Die Erfüllbarkeit von Constraints, die auf Polynomordnungen (Steinbach & Zehnter, 1990) beruhen, kann mit einem ähnlichen Verfahren getestet werden. Dazu müssen Funktionen als lineare Polynome interpretiert werden. Andere Polynome sind nicht möglich, da sonst keine linearen Probleme entstehen.

4.6. Heuristiken

Zum Abschluß des Kapitels beschäftigen wir uns mit einigen Heuristiken, die im Zusammenhang mit geordneten Tableaus und Constraints von Interesse sind.

Wird eine Formel ϕ aufgrund eines Neustarts zu einem Tableauast A geholt, so sollte ϕ möglichst eine (nicht maximale) Konnektion in A besitzen. Mit einer prototypischen Implementierung ließ sich sehr schnell feststellen, daß der Beweis sonst von seiner Zielorientierung verliert. Dies machte sich insbesondere bei redundanten Formelmengen bemerkbar.

Eine übliche Heuristik bei der Formelauswahl bevorzugt Grundformeln. Diese haben die Eigenschaft einfachere Ordnungsconstraints zu liefern. Die Komplexität des globalen Ordnungsconstraints kann dadurch gering gehalten werden.

Wenn mehrere Formeln zur Expansion des Tableaus zur Verfügung stehen, dann kann die Auswahl einer dieser Formeln für die weitere Einschränkung des Suchraums entscheidend sein.

Betrachte die Ordnung $\prec_L$ mit $a_1 < \ldots < a_n$ und folgenden partiellen Ast:

$$\begin{aligned} &1\text{: } P(x_1) \Rightarrow P(a_1) \\ &\vdots \\ &\text{n: } P(x_n) \Rightarrow P(a_n) \ . \end{aligned}$$

Jede dieser Formeln steht für eine Expansion des Asts zur Verfügung. Wird die erste Formel ausgewählt, so kann jede der anderen Formeln zu einem Abschluß verwendet werden. Beginnt man allerdings mit der letzten Formel, so ist kein Abschluß mit einer anderen Formel gültig (falls Constraints verwendet werden). Durch eine geeignete Auswahl der zu expandierenden Formel, lassen sich viele der Abschlüsse vermeiden (hier alle). Die Vollständigkeit des Kalküls bleibt erhalten. Besonders in Hinsicht auf Abschlüsse, die nicht zu einem geschlossenen Tableau führen ist dieser Sachverhalt von Interesse, da sehr viel Backtracking vermieden werden kann. Im obigen Beispiel gibt es bei kontinuierlich schlechter Formelauswahl bis zu $n(n \Leftrightarrow 1)/2$ verschiedene Abschlüsse! Abbildung 4.6 zeigt das in diesem Fall resultierende Tableau. Bei „richtiger“ Formelauswahl entsteht das gleiche Tableau, mit dem einzigen Unterschied, daß die Indizes umgekehrt auftauchen. Gültige Abschlüsse gibt es dann nicht; also auch kein Backtracking.

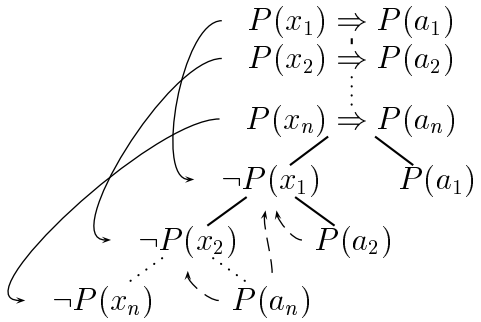


ABBILDUNG 4.6. Vermeiden von unsinnigen Abschlüssen bei geeigneter Formelauswahl

Formeln, die viele maximale Literale besitzen, haben meist auch mehr maximale Konnektionen in einen Ast oder zu anderen Formeln. Dies verursacht entweder eine große Anzahl von Choicepoints, falls Backtracking benutzt wird, um eine faire Auswahl der Konnektionen zu garantieren (Abschnitt 4.4.1). Oder es wird ein komplexer Constraint erzeugt, dessen Erfüllbarkeit alle möglichen maximalen Konnektionen garantiert, aber zeitaufwendiger zu testen ist (Abschnitt 4.4.2). Es könnte daher hilfreich sein Formeln ϕ zu bevorzugen, die wenige maximale Literale in den disjunktiven Pfaden D_ϕ besitzen. Als Vergleichsgröße kann z.B. der Index $\frac{M}{N}$ herangezogen werden, wobei M Anzahl maximaler Literale und N Anzahl aller Literal in ϕ ist. Formeln mit minimalem Index können bevorzugt geholt werden.

KAPITEL 5

Implementierung und Anwendungen

Während wir im vorangegangenen Kapitel Eigenschaften A-geordneter Tableaus mit Constraints besprochen haben, legen wir hier Erfahrungen und Ergebnisse mit einer konkreten Implementierung dar. In Abschnitt 5.1 beschäftigen wir uns mit einigen Aspekten der Implementierung. Danach zeigen wir, wie sich A-geordnete Tableaus sinnvoll für einige Probleme einsetzen lassen (Abschnitt 5.2). Im Mittelpunkt unseres Interesses stehen Probleme aus der Programmverifikation (Abschnitt 5.2.4).

5.1. Implementierung

5.1.1. Der Tableaubeweiser ${}_3T^AP$. ${}_3T^AP$ (Beckert *et al.*, 1996a; Beckert *et al.*, 1996b) ist ein an der Universität Karlsruhe entwickelter tableau-basierter Theorembeweiser für mehrwertige sortierte Logiken erster Stufe. Im wichtigen zweiwertigen Fall wird zusätzlich das Gleichheitsprädikat speziell behandelt. ${}_3T^AP$ ist fast ausschließlich in Sicstus-PROLOG geschrieben.

Der Kalkül von ${}_3T^AP$ basiert auf freie Variablen-Tableaus (Fitting, 1996). Es werden zwei verschiedene Typen von Variablen unterschieden: *starre* Variablen und *universelle*. Während erstere nur einmal für eine Instanziierung (bei einem Abschluß) zur Verfügung stehen, können letztere beliebig oft dafür benutzt werden. Universelle Variablen haben den Vorteil oft zu kleineren Tableaubeweisen zu führen.

${}_3T^AP$ benutzt eine Tiefensuchstrategie, um jeden Ast eines Tableaus einzeln zu schließen. Mit Hilfe von Unifikation werden alle möglichen Abschlußsubstitutionen eines Astes systematisch gesucht. Eine gefundene Abschlußsubstitution wird auf das ganze Tableau angewendet. Danach wird der nächste noch offene Ast behandelt. Die Tiefensuche ist durch die Anzahl möglicher γ-Anwendungen beschränkt. Kann ein Ast nicht mehr erweitert werden und ist die Tiefensuchschranke erreicht, so setzt der Beweiser zurück, um alternative Abschlüsse zu finden.

Bei eingeschalteter Gleichheitsbehandlung werden gemischte E-Unifikationsprobleme vom aktuellen Ast extrahiert. Diese werden mit einem vervollständigungs-basierten Verfahren (Beckert, 1993; Beckert, 1994) in einem Hintergrundbeweiser behandelt. Es wird also totales Theorieschließen benutzt, um einen Ast bezüglich der Gleichheitstheorie abzuschließen.

Um den Suchraum während des Beweisens einzuschränken, berechnet ${}_3T^AP$ zwei verschiedene Arten von Konnektionen. Es werden nur solche Formeln aus der Datenbasis zum betrachteten Ast A geholt, die

1. eine schwache Konnektion oder
2. (bei Gleichheit) eine schwache Gleichheitskonnektion

in A besitzen. Zusätzlich helfen Sorten, die Anzahl dieser Konnektionen gering zu halten.

5.1.2. Ordnungseinschränkungen. Wir haben die Ordnungseinschränkungen mit Hilfe zweier neuen Module als Teil von ${}_3T^AP$ implementiert. Eines, welches die verwendete A-Ordnung realisiert und eine zweites, welches Ordnungsconstraints bezüglich dieser A-Ordnung lösen kann. Falls zu einer Ordnung keine Ordnungsconstraints behandelt werden, kann letzteres Modul auch eine Attrappe sein. Möchte man eine andere A-Ordnung verwenden, so brauchen nur diese beiden Module ausgetauscht werden. Derzeit stehen die Ordnungen $\prec_L, \prec_\#$ und eine Lock-Einschränkung $\prec_{\text{lock}}$ zur Verfügung. $\prec_{\text{lock}}$ ordnet die Literale einer Formel mit Hilfe eines Index. Dieser Index ist eindeutig für jedes Vorkommen eines Literals in einer Formel. Diese drei Ordnungen haben den Vorteil, noch über eine Grundordnung variiert werden zu können.

Wir haben nur für $\prec_\#$-geordnete Constraints einen Erfüllbarkeitstest implementiert. $\prec_\#$-geordnete Constraints werden mit Hilfe von Satz 4.2 in Probleme der ganzzahligen Optimierung transformiert und mit dem Sicstus 3 Modul `clpq` weiterbehandelt. `clpq` ist eine PROLOG-Erweiterung mit der beliebige lineare rationale Constraints gelöst werden können. Die Verwendung dieses Moduls führt allerdings zu keinem sehr effizienten Erfüllbarkeitstest, da es keinerlei Möglichkeiten gibt, auf den von `clpq` vereinfachten Constraint zuzugreifen. Dadurch werden bei einem Erfüllbarkeitstest immer wieder die gleichen Teile des Constraints durchsucht, obwohl diese stark vereinfacht werden könnten. Das Modul `clpq` hat im Gegensatz zu vielen anderen Werkzeugen aber den wesentlichen Vorteil disjunktive Constraints behandeln zu können. Zusätzlich kann Backtracking einfach berücksichtigt werden, da `clpq` eine Erweiterung von PROLOG darstellt.

Anstatt wie bisher Formeln aufgrund einer schwachen (Gleichheits-)Konnektion zu einem Ast zu holen, werden diese Konnektionen auf maximale Literale eingeschränkt. Bei einer späteren Expansion einer Formel wird nicht mehr beachtet, ob diese maximale Konnektionen in den Ast oder zu einer anderen Formel in der Datenbasis besitzen. Es wird also der Kalkül von Tabelle 4.2 verwendet.

Um zu testen, ob eine maximale Konnektion besteht oder nicht, wird zu jedem Literal L einer Formel ϕ die Ordnungsinformation $\max(L, \phi)$ vorberechnet. Dafür wird eine von der verwendeten A-Ordnung noch unabhängige, speicherplatzsparende Darstellung benutzt, bei der für $\max(L, \phi)$ nur die relevanten Teile der disjunktiven Pfade von ϕ gespeichert werden. Pfade in denen L nie maximal auftritt können weggelassen werden. Falls dies für alle disjunktiven Pfade durch ϕ gilt, so ist $\max(L, \phi)$ immer unerfüllbar. Weiterhin brauchen auch die Teile eines Pfades nicht berücksichtigt werden, in denen L immer maximal auftritt. Ist L in einem disjunktiven Pfad durch ϕ immer maximal, so ist $\max(L, \phi)$ immer erfüllbar.

BEISPIEL 5.1. Gilt $A < L$ und $L < B$, so ist L maximal in $A \vee (B \wedge C)$ genau dann, wenn L maximal in $(B \wedge C)$ ist. Dies ist genau dann der Fall, wenn L maximal in C ist.

Wenn eine Formel auf das Tableau geholt wird, wird aus den vorberechneten Constraints eine Ordnungsbedingung erzeugt, die sowohl alle maximalen Konnektionen in den aktuellen Ast und auch zu Formeln, die sich noch in der Datenbasis befinden, umfaßt. Deswegen ist es nicht notwendig, über die Auswahl maximaler Konnektionen zurückzusetzen, um einen vollständigen Kalkül zu erhalten. Erst danach wird dieser Constraint in einen A-geordneten Constraint umgewandelt und speziell behandelt.

Zusätzlich zu den „normalen“ komplementären Paaren bzw. Konnektionen, existiert die schwache Gleichheitskonnektion, um gezielt Formeln mit Gleichungen oder potentiell komplementären Paaren auf das Tableau zu holen (Sulzmann, 1995). Diese haben wir auf maximale Konnektionen in den aktuellen Ast eingeschränkt. Dabei werden aber *keine* schwachen Gleichheitskonnektionen zwischen Formeln betrachtet, die nicht auf dem Ast sind. Da man im Extremfall drei Formeln betrachten müßte, um eine schwache Gleichheitskonnektion zu berechnen, würde dies einen unverhältnismäßig hohen Aufwand verursachen (z.B. bei den drei Formeln $p(a), \neg p(b)$ und $a \approx b$). Weil der Kalkül in diesem Fall nicht mehr vollständig ist, wird die Ordnungseinschränkung nur als Heuristik benutzt: Solange Formeln mit maximaler schwacher Gleichheitskonnektion in den aktuellen Ast existieren, werden diese bevorzugt. Ansonsten werden auch nicht maximal verbundene Formeln geholt. Der Zeitaufwand bei der Berechnung der schwachen Gleichheitskonnektionen ist auch der Grund, nicht alle möglichen maximalen Konnektionen in den aktuellen Ast zu berücksichtigen, sondern nur die erste gefundene. Der Beweiser setzt aber trotzdem nicht über die Auswahl dieser Konnektionen zurück, da dies wesentlich ungünstiger ist als ein unvollständiger Kalkül (falls Ordnungsconstraints behandelt werden).

Zusätzliche Probleme ergaben sich aus dem Vorhandensein universeller Variablen. Da Abschlüsse mit universellen Variablen nicht vermerkt werden und diese Variablen besonders im Hintergrundbeweiser für die Gleichheit beliebig oft umbenannt werden, wäre es nur mit unverhältnismäßigen Aufwand möglich, alle Instantiierungen aufzusammeln. Dies hat z.B. zur Folge, daß $p(x)$ in $\forall x(p(x) \vee q(f(a))$ bezüglich der $\prec_L$-Ordnung immer maximal bleibt, obwohl eine Instantiierung von x mit a die Ordnungsbedingung $x \not\prec_\tau f(a)$ verletzt. Da universelle Variablen aber meist nur in nichtverzweigenden Formeln vorkommen, ist dies nicht weiter schlimm. In diesen Formeln sind die Literale aus trivialem Grund immer maximal (es gibt in jedem disjunktiven Pfad nur ein Literal).

5.2. Anwendungen

In den folgenden Abschnitten zeigen wir einige Anwendungen geordneter Tableaus. Dabei kommen die Ordnungen $\prec_L$, $\prec_{\text{lock}}$ und $\prec_\#$ zum Einsatz.

Wir haben die Laufzeiten alle auf einer Sun SPARCstation 10 unter Sicstus-PROLOG Version 3 ermittelt. Die Einträge in den Spalten der Tabellen haben folgende Bedeutung:

Name:	Name eines Theorems des behandelten Problems.
Ordnung:	Die verwendete Ordnungsrelation.
TR:	Anzahl Tableauregeln.
EQ:	Anzahl Aufrufe des Hintergrundbeweisers.
EC:	Anzahl Abschlüsse mit Gleichheit.
CB:	Gesamtanzahl geschlossener Äste.
PR:	Anzahl abgeschnittener Äste.
BT:	Gibt an, wie oft der Beweiser zurücksetzen mußte.
MC:	Beschränkung der γ-Anwendungen.[1]

Bei den Ergebnissen geben wir nur die relevanten Teile dieser Informationen an. Da das Holen von Formeln bei geordnetem Tableau aufwendiger ist, als bei Tableaus ohne Ordnungseinschränkung, weisen wir darauf hin, daß die gemessenen Zeiten nur einen ungenauen Vergleich zulassen. Dagegen ist die Zahl der Tableauregeln (TR) und geschlossenen Äste (CB) aussagekräftiger, da diese in direkter Relation zur Größe des gefundenen Tableaus stehen.

5.2.1. Pigs and Balloonists. Das Problem Pigs and Balloonists (Carroll, 1986) (in ${}_3T^AP$-Syntax siehe Tabelle 5.1) hat folgende umgangssprachlichen Formulierung (aus der TPTP-Bibliothek (Sutcliffe *et al.*, 1994)):

- All, who neither dance on tight ropes nor eat penny-buns, are old.
- Pigs, that are liable to giddiness, are treated with respect.
- A wise balloonist takes an umbrella with him.
- No one ought to lunch in public who looks ridiculous and eats penny-buns.
- Young creatures, who go up in balloons, are liable to giddiness.
- Fat creatures, who look ridiculous, may lunch in public, provided that they do not dance on tight ropes.
- No wise creatures dance on tight ropes, if liable to giddiness.
- A pig looks ridiculous, carrying an umbrella.
- All, who do not dance on tight ropes, and who are treated with respect are fat.
- Show that no wise young pigs go up in balloons.

Wie sofort einzusehen ist, handelt sich es hier im Grunde um ein rein aussagenlogisches Problem, wobei die Formeln fast in Hornform sind. Von wesentlicher Bedeutung sind die drei atomaren Axiome `wise(piggy)`, `young(piggy)` und `pig(piggy)`. Diese können einen Ast sofort schließen und sollten deswegen bevorzugt geholt werden.

Mit $\prec_L$-geordneten Tableaus kann man dies wie folgt erreichen. Enthält ein Ast ein zu den atomaren Formeln komplementäres Literal, so sollte günstigerweise nur eine der atomaren Formeln zum Ast geholt werden. Alle anderen Konnektionen sollten nach Möglichkeit unberücksichtigt bleiben. Dies kann

[1] Dieser Wert gibt an, wie oft eine Formel für eine γ-Regel benutzt werden darf.

```
axiom boring_old_people;
  forall x ( -dances_on_tightropes(x) & -eats_pennybuns(x) =>  old(x)).
axiom giddy_pigs_reated_with_respect;
  forall x ( pig(x) & liable_to_giddiness(x) => treated_with_respect(x)).
axiom wise_ballonists_have_umbrellas;
  forall x ( wise(x) & balloonist(x) => has_umbrella(x)).
axiom dont_look_ridiculous_eating_buns_in_public;
  forall x ( looks_ridiculous(x) & eats_pennybuns(x) =>
     -eats_lunch_in_public(x)).
axiom young_balloonists_get_giddy;
  forall x ( balloonist(x) & young(x) => liable_to_giddiness(x)).
axiom fat_ridiculous_off_tightrope_eat_in_public;
  forall x ( fat(x) & looks_ridiculous(x) & -dances_on_tightropes(x) =>
      eats_lunch_in_public(x)).
axiom wise_giddy_dont_dance_on_tightrope;
  forall x ( liable_to_giddiness(x) & wise(x) => -dances_on_tightropes(x)).
axiom pigs_look_ridiculous_with_umbrellas;
  forall x ( pig(x) & has_umbrella(x) =>  looks_ridiculous(x)).
axiom non_dancers_who_are_respected_are_fat;
  forall x ( -dances_on_tightropes(x) & treated_with_respect(x) => fat(x)).

axiom either_young_or_old;
  forall x ( young(x) <=> -old(x)).

axiom piggy_is_wise; ( wise(piggy)).
axiom piggy_is_young; ( young(piggy)).
axiom piggy_is_a_pig; ( pig(piggy)).

theorem t; -balloonist(piggy).
```

TABELLE 5.1. Pigs and Balloonists

man erreichen, indem die Anzahl maximaler Konnektionen zu den Axiomen minimiert wird. Dazu stufen wir das Prädikatensymbol der Konklusion einer Formel immer größer ein als die Symbole der Prämisse.

Tabelle 5.2 zeigt die Ergebnisse mit ($\prec_L$) und ohne (keine) Ordnungseinschränkung. Besonders von Bedeutung sind die schattierten Spalten, da sie ein ungefähres Maß für die Größe des Tableaus darstellen. Die jeweiligen günstigeren Werte zu einem Theorem sind dunkler schattiert.

Wird keine Ordnungseinschränkung verwendet, so fällt die große Anzahl abgeschnittener Äste auf (PR). Dies deutet an, daß die Beweissuche wenig zielgerichtet vonstatten ging, da ein Teil der Formeln auf einem Ast für den Beweis nicht verwendet wurde. Mit Ordnungseinschränkung ist diese Redundanz verschwunden und das Tableau ist erheblich kleiner. Dieses Ergebnis kann auch mit einer Heuristik erzielt werden, die Formeln mit kleinstem Verzweigungsgrad bevorzugt.

Name	Ordnung	Zeit [s]	TR	CB	PR
t	$\prec_L$	2.75	62	35	0
	keine	4.16	323	112	55

TABELLE 5.2. Ergebnisse zu Pigs and Balloonists

5.2.2. Tante Agatha. Im folgenden bekannten Problem (PUZ001-1.p aus TPTP) geht es um einen Mordfall im Hause Dreadbury:

> Someone who lives in Dreadbury Mansion killed Aunt Agatha. Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein. A killer always hates his victim, and is never richer than his victim. Charles hates no one that Aunt Agatha hates. Agatha hates everyone except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone Aunt Agatha hates. No one hates everyone. Agatha is not the butler. Therefore: Agatha killed herself.

```
axiom agatha;  lives(agatha).
axiom butler;  lives(butler).
axiom charles; lives(charles).
axiom agatha_hates_agatha;  hates(agatha,agatha).
axiom agatha_hates_charles; hates(agatha,charles).

axiom poorer_killer;
  forall x,y  ( killed(x,y) => -richer(x,y)).
axiom different_hates;
  forall x  ( hates(agatha,x) => -hates(charles,x)).
axiom no_one_hates_everyone;
  forall x  ( - (hates(x,agatha) & hates(x,butler) & hates(x,charles)) ).
axiom killer_hates_victim;
  forall x,y ( killed(x,y) => hates(x,y)).
axiom same_hates;
  forall x ( hates(agatha,x) => hates(butler,x)).
axiom butler_hates_poor;
  forall x ( lives(x) & -richer(x,agatha) => hates(butler,x)).

theorem t; -(killed(butler,agatha) & killed(charles,agatha)).
```

TABELLE 5.3. Tante Agatha

Bei der Formulierung in Tabelle 5.3 soll bewiesen werden, daß weder der Butler noch Charles der Mörder ist. Damit das Tableau nicht unnötig verzweigt, sollten wie schon bei dem vorangehenden Problem `pigs and balloonists` die atomaren Formeln bevorzugt werden. Dazu benutzen wir wieder die $\prec_L$-Ordnung. Mit der Grundordnung `lives,hates < killed,richer` sorgen wir dafür, daß möglichst nur die atomaren Formeln aufgrund einer maximalen Konnektion geholt werden. Auch wenn in diesem Fall die Ordnung dies nicht immer garantieren kann, zeigen die Ergebnisse in Tabelle 5.4, daß diese Vorgehensweise erfolgreich ist. Das geordnete Tableau ist wieder kleiner (TR,CB) und der Beweiser muß weniger häufig zurücksetzen (BT).

Name	Ordnung	Zeit [s]	TR	CB	BT
t	$\prec_L$	1.11	24	9	3
	keine	1.57	52	12	14

TABELLE 5.4. Ergebnisse zu Tante Agatha

5.2.3. Ausnutzen hierarchischer Informationen. Bei einer großen Anzahl von Problemen läßt sich eine hierarchische Struktur der Axiomatisierung feststellen. Dies gilt besonderes für mathematische Probleme. Aufbauend auf einige Definitionen (z.B. Gruppen) werden neue Definitionen eingeführt (z.B. Ringe). Wir betrachten hier exemplarisch eine Axiomatisierung einer Art Mengenlehre. Abbildung 5.1 zeigt eine Hierarchie für Prädikate, wie sie in der Mengenlehre üblicherweise verwendet werden. Die symmetrische Differenz Δ kann mit Hilfe von Vereinigung ($\cup$), Durchschnitt ($\cap$) und Differenz ($\Leftrightarrow$) von Mengen definiert werden und letztere jeweils mit $\in$. Zusätzlich ist noch das Komplement A^c einer Menge A vorhanden. Eine Axiomatisierung von Mengen in ${}_3TAP$-Syntax ist in Tabelle 5.5 gegeben. Die Hierarchie der Mengenoperatoren spiegelt sich offenbar in den Axiomen wieder.

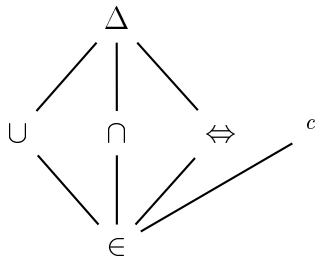


Abbildung 5.1. Hierarchie von Prädikaten einer Mengenlehre

Die Knoten einer Hierarchie mit den zugehörigen Formeln, bezeichnen wir im folgenden als eine *Spezifikation*. Zum Beispiel bilden die Axiome `def_of_union_1` und `def_of_union_2` die Spezifikation der Vereinigungsmenge. Diese ist eine *Teilspezifikation* der symmetrischen Differenz, da dort die Prädikate aus der Spezifikation der Vereinigung einer Menge benötigt werden.

Um ein Theorem wie `t_1` zu beweisen ist es sinnvoll das Tableau schrittweise mit Formeln zu erweitern, die zur nächsten tiefer gelegenen Hierarchie gehören. Vor allen Dingen ist es meist unnötig in der Hierarchie nach oben zu steigen. Ein Axiom aus der Spezifikation des symmetrischen Durchschnitts sollte beispielsweise nicht aufgrund einer Konnektion zu einem Literal mit dem Prädikat `member` geholt werden, da dies in keinster Weise zielorientiert ist (`member` kommt schließlich in fast jeder Formel vor). Mit Hilfe dieser Strategie werden dann hoffentlich nur Formeln zu einem Beweis benutzt, die auch wirklich benötigt werden.

Dieses Verfahren läßt sich teilweise mit einer Ordnungseinschränkung modellieren, in dem jedes Literal eines Axioms maximal ist, wenn es nicht zu einer darunter liegenden Hierarchieebene gehört, d.h. keine neu eingeführten Symbole enthält (Prädikaten-, Funktionssymbole und evtl. neue Sorten). Mit $\prec_A$ und der Grundordnung `member<diff,union,intersection<symdiff` kann dieses Ziel in unserem Beispiel nahezu perfekt erreicht werden.

Tabelle 5.6 zeigt die Ergebnisse für geordnete und nicht geordnete Tableaus. Die gefundenen $\prec_L$-geordneten Tableaus sind allesamt kleiner als die nicht geordneten Tableaus.

```
axiom def_of_union_1 ; forall x ( forall a,b,u (
   union(a,b,u) & member(x,u) => (member(x,a) v member(x,b)) )).
axiom def_of_union_2 ; forall x ( forall a,b,u (
   union(a,b,u) & member(x,u) <= (member(x,a) v member(x,b)) )).

axiom def_of_intersection_1 ; forall x ( forall a,b,i (
   (intersection(a,b,i) & member(x,i)) => (member(x,a) v member(x,b)) )).
axiom def_of_intersection_2 ; forall x ( forall a,b,i (
   (intersection(a,b,i) & member(x,i)) <= (member(x,a) v member(x,b)) )).

axiom def_of_diff_1 ;  forall x ( forall a,b,d (
    diff(a,b,d) & member(x,d) => member(x,b )  )).
axiom def_of_diff_1 ;  forall x ( forall a,b,d (
    diff(a,b,d) & member(x,d) <= member(x,b )  )).

axiom def_of_symdiff_1 ; forall x ( forall a,b,s,u,i,d(
    symdiff(a,b,s) & member(x,s) =>
    union(a,b,u) & intersection(a,b,i) & diff(u,i,d) & member(x,d) )).
axiom def_of_symdiff_1 ; forall x ( forall a,b,s,u,i,d(
    symdiff(a,b,s) & member(x,s) <=
    union(a,b,u) & intersection(a,b,i) & diff(u,i,d) & member(x,d) )).

axiom def_of_complement_1 ; forall x ( forall a,c (
    complement(a,c) & member(x,c) => -member(x,a) ) ).
axiom def_of_complement_1 ; forall x ( forall a,c (
    complement(a,c) & member(x,c) <= -member(x,a) ) ).

theorem t_1 ; forall a,u ( forall x (
 ( union(a,a,u) & member(x,u) <=> member(x,a) ) )).
theorem t_2 ; forall a,i ( forall x (
 ( intersection(a,a,i) & member(x,i) <=> member(x,a) ) )).
theorem t_3 ; forall a,b,u_1,u_2,u (
 ( union(a,b,u_1) & union(u_1,c,u) <=>
   union(b,c,u_2) & union(a,u_3,u) ) ).
theorem t_4 ; forall a,b,i_1,i_2,i (
 ( intersection(a,b,i_1) & intersection(i_1,c,i) <=>
   intersection(b,c,i_2) & intersection(a,i_2,i) ) ).
theorem t_5 ; forall a,b,s,d_1,d_2 (
 ( symdiff(a,b,s) <=> diff(a,b,d_1) & diff(b,a,d_2) & union(d_1,d_2,s) ) ).
theorem t_6 ; forall a,b,s,i,u (
 ( symdiff(a,b,s) & intersection(a,b,i) & union(s,i,u) <=> union(a,b,u) ) ).
```

TABELLE 5.5. Axiomatisierung einer Mengenlehre

Wir haben versucht diese positiven Ergebnisse auf die Problemklasse SET der TPTP-Bibliothek zu übertragen. Tabelle 5.7 zeigt einige Ergebnisse. SET umfaßt ca. 700 verschiedene Probleme, von denen wir nur einen kleinen Teil berücksichtigen konnten. Wir haben uns auf Probleme beschränkt, die keine Gleichheit enthalten. Auch bei SET sind die Tableaubeweise mit $\prec_L$-Ordnung meist kleiner. In einigen Fällen wird ohne Ordnungseinschränkung überhaupt kein Beweis gefunden. Dies zeigt deutlich, wie wichtig es sein kann zusätzliche Informationen auszunutzen, die nicht in der Syntax des Problems enthalten sind (hier die Hierarchieinformationen).

5.2.4. KIV-Probleme. Eine hierarchische Axiomatisierung, wie wir sie im vorangegangenen Kapitel erläuterten, findet sich auch bei Problemen aus

Name	Ordnung	Zeit [s]	TR	CB	PR	BT	MC
t_1	$\prec_L$	0.63	28	7	1	0	0
	keine	1.28	74	15	12	0	0
t_2	$\prec_L$	0.48	23	6	1	0	0
	keine	0.63	50	12	5	0	0
t_3	$\prec_L$	4.66	152	39	3	0	0
	keine	10.11	396	101	18	4	0
t_4	$\prec_L$	2.07	85	22	1	0	0
	keine	1.91	131	33	6	0	0
t_5	$\prec_L$	76.77	255	76	20	0	1
	keine	41.38	337	107	27	0	1
t_6	$\prec_L$	2.23	81	23	0	0	0
	keine	2.55	144	41	1	0	0

TABELLE 5.6. Ergebnisse zur Formulierung einer Mengenlehre

Name	Ordnung	Zeit [s]	TR	CB	PR	BT	MC
set001	$\prec_L$	20.98	337	40	1	123	0
	$\prec_L$	1.18	30	11	1	0	1
	keine	0.12	10	4	0	0	0
set002	$\prec_L$	27.82	546	206	37	2	0
	keine	> 500					0–2
set003	$\prec_L$	0.26	12	5	0	0	0
	keine	0.66	48	8	13	0	0
set004	$\prec_L$	6.12	12	5	0	0	0
	keine	0.72	48	8	13	0	0
set006	$\prec_L$	52.15	650	40	5	174	0
	keine	> 500					0–2
set008	$\prec_L$	102.29	1406	536	73	0	1
	keine	> 500					0–2
set009	$\prec_L$	0.27	22	10	4	0	0
	keine	0.17	22	10	4	0	0

TABELLE 5.7. Ergebnisse zur Problemklasse `SET` aus der TPTP-Bibliothek

der Programmverifikation wieder. Wir betrachten dazu Probleme des Karlsruhe Interactive Verifier (Reif, 1992) (KIV). Über die Hierarchieinformationen hinaus sind in den KIV-Problemen einige Implikationen und Äquivalenzen vom Benutzer mit einer Richtungsinformation versehen. Diese sogenannten Simplifier entsprechen semantisch der üblichen Implikation oder Äquivalenz, werden aber innerhalb des KIV-Systems als Ersetzungsregeln benutzt. Bei einem Tableaubeweiser bedeutet dies, daß bei einem Simplifier $A \Rrightarrow B$ jedes Vorkommen einer Instanz $A\sigma$ auf einem Ast durch $B\sigma$ ersetzt wird. Es ist also sinnvoll einen Simplifier nur dann auf einen Tableauast zu holen, wenn er eine Konnektion bezüglich A in den Ast besitzt. Simplifier sind in ${}_3T^AP$ noch nicht

auf diese Weise implementiert. Sie werden wie eine normale Implikation behandelt, mit dem einzigen Unterschied, daß die Extension, welche A enthält, bevorzugt betrachtet wird.

Mit einer geeigneten Ordnungseinschränkung kann erreicht werden, daß ein Simplifier nur dann auf einen Ast geholt wird, wenn er über die Prämisse eine Konnektion besitzt. Dazu müssen die Literale der Prämisse immer größer sein als die der Konklusion. Mit der $\prec_{\text{lock}}$-Ordnung kann dies immer garantiert werden. Dazu werden die zu einem Literal gehörigen Bezeichner in einer Vorverarbeitung geeignet geordnet. Leider läßt diese Ordnung keine Modifikation mehr zu, die es erlaubt, die Hierarchieeigenschaften der Probleme auszunutzen. Daß dies sinnvoll sein kann, hat sich aber im vorangegangenen Abschnitt gezeigt. Die $\prec_L$-Ordnung läßt sich zu diesem Zweck bei den KIV-Problemen leider nur schwer benutzen, da es meist nur sehr wenige Prädikatensymbole gibt, die sich aufgrund einer Grundordnung vergleichen lassen. Darüberhinaus ist das Gleichheitsprädikat meist das vorherrschende.

Bei eingehender Betrachtung der KIV-Probleme fällt auf, daß die Literale in einer Prämisse eines Simplifiers meist mehr Symbole enthalten, als die Literale der Konklusion. Wie zum Beispiel bei folgendem Axiom aus der Spezifikation natürlicher Zahlen:

$$\forall n, n_0 (jsuc(n) = jsuc(n0) <=>> n = n0)$$

Bei einem Simplifier wird ein Literal also meistens durch ein Literal mit weniger Symbolen ersetzt.

Diese Eigenschaft läßt sich mit der $\prec_{\#}$-Ordnung (siehe Def. 4.6) abbilden: Literale sind größer, wenn sie mehr und gewichtigere Symbole enthalten. Weiterhin kann mit Hilfe der Bewertung # der Symbole noch zusätzlich die Hierarchie der KIV-Probleme berücksichtigt werden. Ein Symbol einer niedrigeren Hierarchie bekommt einen kleineren Wert zugewiesen, als ein Symbol einer darüberliegenden. An unterster Stelle der Hierarchie befindet sich die Gleichheit, welche von allen oberen Spezifikationen benutzt wird. Die $\prec_{\#}$-Ordnung hat auch noch den Vorteil invariant bezüglich der Vertauschung von Funktionssymbolen zu sein. Besonders bei dem Gleichheitsprädikat ist dies hilfreich, da $\prec_{\#}$ die Literale $s \approx t$ und $t \approx s$ nicht unterschiedlich behandelt.

Wir haben eine Reihe verschiedener KIV-Probleme mit $\prec_{\text{lock}}$-, $\prec_{\#}$-geordnetem und nicht geordnetem Tableau getestet. Dabei haben wir jedes Symbol mit 1 bewertet. Hierarchiespezifische Eigenschaften werden also noch nicht ausgenutzt. Es werden auch keine Ordnungsconstraints erzeugt. Wir haben die $\prec_{\text{lock}}$-Einschränkung so benutzt, daß alle Literale der Prämisse eines Simplifiers maximal sind. Obwohl bei allen behandelten Beispiele das Gleichheitsprädikat vorkommt, konnten auch viele Beweise ohne Gleichheitsbehandlung gefunden werden. Wie geben zu jedem Problem trotzdem beide Ergebnisse an, da bei angeschalteter Gleichheitsbehandlung Gleichheitskonnektionen berechnet werden, welche den Suchraum anders orientieren können.

Das Problem `circ-list_memory` enthält eine größere Anzahl von Simplifiern. Die Ergebnisse $\prec_{\text{lock}}$-geordneter Tableaus weichen allerdings nicht sonderlich von Tableaus ohne Einschränkung ab (Tabelle 5.8). Wird Gleichheit benutzt (Tabelle 5.9), so verhält sich die Lock-Einschränkung sogar eher etwas schlechter (`th_4`). Bei $\prec_{\#}$-geordnetem Tableau sind die Tableaubeweise fast immer kleiner als bei $\prec_{\text{lock}}$- oder nicht geordnetem Tableau. In einigen Fällen kann ein Beweis nur mit $\prec_{\#}$ gefunden werden. Dies kann als Hinweis gelten, daß die $\prec_{\#}$-Ordnung über die vom Benutzer in den Simplifiern verborgenen Ordnungsinformationen hinausgeht.

Name	Ordnung	Zeit [s]	TR	CB	PR	BT	MC
th_1	$\prec_{\#}$	0.39	10	2	0	0	0
	Lock	0.16	10	2	0	0	0
	keine	0.16	10	2	0	0	0
th_2	$\prec_{\#}$	0.25	12	1	0	0	0
	Lock	0.14	13	3	0	0	0
	keine	0.14	13	3	0	0	0
th_3	$\prec_{\#}$	1.27	25	6	0	0	0
	Lock	0.25	22	7	0	0	0
	keine	0.25	22	7	0	0	0
th_4	$\prec_{\#}$	0.38	14	3	0	0	0
	Lock	0.51	14	3	0	0	0
	keine	0.51	42	8	0	0	0
th_5	$\prec_{\#}$	1.45	54	9	1	0	0
	Lock	> 500					0–2
	keine	> 500					0–2
th_6	$\prec_{\#}$	0.25	13	3	0	0	0
	Lock	0.90	20	5	0	0	0
	keine	0.64	48	10	0	0	0
th_7	$\prec_{\#}$	2.76	64	15	0	0	0
	Lock	2.54	61	16	4	0	0
	keine	0.70	61	16	4	0	0
th_8	$\prec_{\#}$	2.50	47	9	0	0	0
	Lock	> 500					0
	keine	> 500					0

TABELLE 5.8. Ergebnisse zu `circ-list_memory`

Das Problem `nonindtarget` (Tabelle 5.10) enthält nur ganz wenige Simplifier. Trotzdem wird bei Theorem `th_2` eine kleine Verbesserung gegenüber Tableaus ohne Ordnungseinschränkung erzielt. Die in den Simplifiern enthaltende Ordnungsinformation kann also sinnvoll verwendet werden. Nur die $\prec_{\#}$-Ordnung schneidet hier etwas schlecht ab. Bei eingeschalteter Gleichheit (Tabelle 5.11) wird kein einziger Beweis mit $\prec_{\text{lock}}$-Einschränkung gefunden. Da es aber nur zwei Theoreme gibt lassen sich hieraus keine allgemeinen Rückschlüsse ziehen.

Das Problem `objectmemory` enthält keinerlei Simplifier. Deswegen ist die Lock-Einschränkung nicht sinnvoll einsetzbar. Mit der Ordnung $\prec_{\#}$ werden

Name	Ordnung	Zeit [s]	TR	CB	EQ	EC	PR	BT	MC
th_1	$\prec_{\#}$	1.10	14	2	1	2	0	0	0
	Lock	1.23	14	2	1	2	0	0	0
	keine	1.08	14	2	1	2	0	0	0
th_2	$\prec_{\#}$	0.24	12	1	0	0	0	0	0
	Lock	0.95	17	3	1	3	0	0	0
	keine	0.92	17	3	1	3	0	0	0
th_3	$\prec_{\#}$	45.33	42	6	4	9	0	0	0
	Lock	1.05	34	7	3	9	0	0	0
	keine	1.10	34	7	3	9	0	0	0
th_4	$\prec_{\#}$	18.10	46	7	5	11	0	0	0
	Lock	34.68	89	13	9	21	0	0	0
	keine	1.52	42	6	1	6	0	0	0
th_5	$\prec_{\#}$	0.46	15	1	0	2	1	0	0
	Lock	> 500							0–2
	keine	> 500							0–2
th_6	$\prec_{\#}$	0.86	17	1	1	4	1	0	0
	Lock	> 500							0–2
	keine	2.66	41	6	1	6	0	0	0
th_7	$\prec_{\#}$	11.59	92	24	4	18	1	0	0
	Lock	3.51	95	20	8	27	0	0	0
	keine	3.48	95	20	8	27	0	0	0
th_8	$\prec_{\#}$	47.34	61	7	3	10	1	0	0
	Lock	> 500							0–2
	keine	> 500							0–2

TABELLE 5.9. Ergebnisse zu `circ-list_memory` mit Gleichheit

Name	Ordnung	Zeit [s]	TR	CB	PR	BT	MC
th_1	$\prec_{\#}$	1.77	57	7	9	0	0
	Lock	2.16	56	8	9	0	0
	keine	0.91	56	8	9	0	0
th_2	$\prec_{\#}$	53.36	501	115	14	8	0
	Lock	3.30	81	19	2	0	0
	keine	1.68	85	19	3	0	0

TABELLE 5.10. Ergebnisse zu `nonindtarget`

zum Teil gute Ergebnisse erzielt, obwohl keine vom Benutzer durch Simplifier vorgegebene Ordnungsinformation vorhanden ist (Tabelle 5.12). Auch bei angeschalteter Gleichheit verhält sich die $\prec_{\#}$-Ordnung positiv (Tabelle 5.13).

`hashtable` stellt eine Spezifikation einer Haschtabelle dar. Da ohne Gleichheit so gut wie keine Theoreme zu beweisen sind, zeigen wir nur die Ergebnisse mit eingeschalteter Gleichheit (Tabelle 5.14). $\prec_{\text{lock}}$-geordnete Tableaus haben hier gegenüber nicht geordnetem Tableau keinen großen Vorteil. Besser schneidet hier wieder die $\prec_{\#}$-Ordnung ab. In vielen Fällen werden kleinere Tableaus oder zusätzliche Beweise gefunden. Ein Wermutstropfen sind hier

Name	Ordnung	Zeit [s]	TR	EQ	EC	CB	PR	BT	MC
th_1	$\prec_{\#}$	4.25	35	7	3	5	0	0	0
	Lock	> 500							0–2
	keine	> 500							0–2
th_2	$\prec_{\#}$	> 500							0–2
	Lock	> 500							0–2
	keine	96.01	46	10	5	6	0	0	0

TABELLE 5.11. Ergebnisse zu `nonindtarget` mit Gleichheit

Name	Ordnung	Zeit [s]	TR	CB	PR	BT	MC
th_1	$\prec_{\#}$	0.19	9	2	0	0	0
	keine	0.16	12	3	0	0	0
th_2	$\prec_{\#}$	0.26	14	3	0	0	0
	keine	0.50	31	6	1	0	0
th_3	$\prec_{\#}$	0.16	13	2	0	0	0
	keine	0.24	21	4	0	0	0
th_4	$\prec_{\#}$	0.24	13	3	0	0	0
	keine	0.20	16	4	0	0	0
th_5	$\prec_{\#}$	> 500					0–2
	keine	2.69	148	30	8	0	0
th_6	$\prec_{\#}$	0.19	9	2	0	0	0
	keine	0.17	12	2	1	0	0

TABELLE 5.12. Ergebnisse zu `objectmemory`

Name	Ordnung	Zeit [s]	TR	CB	EQ	EC	PR	BT	MC
th_1	$\prec_{\#}$	1.00	19	2	2	3	0	0	0
	keine	1.28	32	3	3	5	0	0	0
th_2	$\prec_{\#}$	16.06	39	7	5	11	0	0	0
	keine	56.64	117	12	9	20	0	0	0
th_3	$\prec_{\#}$	0.46	20	2	1	2	0	0	0
	keine	1.48	44	4	3	6	0	0	0
th_4	$\prec_{\#}$	19.49	35	6	2	7	0	0	0
	keine	2.98	49	6	3	9	1	0	0
th_5	$\prec_{\#}$	6.13	56	13	5	9	0	0	0
	keine	> 500							0–2
th_6	$\prec_{\#}$	0.81	17	2	2	3	0	0	0
	keine	2.70	25	3	3	5	0	0	0

TABELLE 5.13. Ergebnisse zu `objectmemory` mit Gleichheit

aber die Theorem `th_25`, `th_26` und `th_27`, da diese nur mit Tableaus ohne Ordnungseinschränkungen oder mit $\prec_{\text{lock}}$-Ordnung bewiesen werden können.

Eine generelle Aussage über die Vor- und Nachteile der verwendeten Einschränkungen ist schwierig. Zumindestens $\prec_{\text{lock}}$-geordnete Tableaus bringen keinen entscheidenen Vorteil gegenüber uneingeschränkten Tableaus. Insbesondere wenn überhaupt keine Simplifier in den KIV-Problemen vorkommen,

Name	Ordnung	Zeit [s]	TR	CB	EQ	EC	PR	BT	MC
th_10	$\prec_{\#}$	92.50	32	11	6	6	0	0	0
	Lock	3.21	31	7	4	4	0	0	0
	keine	2.81	31	7	4	4	0	0	0
th_12	$\prec_{\#}$	17.70	5	3	2	2	0	0	0
	Lock	31.10	12	5	3	3	0	0	0
	keine	1.36	18	5	3	3	0	0	0
th_13	$\prec_{\#}$	0.20	9	1	1	1	0	0	0
	Lock	0.20	9	1	1	1	0	0	0
	keine	0.13	10	1	1	1	0	0	0
th_24	$\prec_{\#}$	18.49	24	4	3	6	0	0	0
	Lock	9.72	63	10	8	17	0	0	0
	keine	8.94	63	10	8	17	0	0	0
th_25	$\prec_{\#}$	> 500							0–2
	Lock	89.10	286	41	41	81	0	0	0
	keine	90.71	302	41	41	81	0	0	0
th_26	$\prec_{\#}$	> 500							0–2
	Lock	21.51	91	13	13	25	0	0	0
	keine	19.19	91	13	13	25	0	0	0
th_27	$\prec_{\#}$	> 500							0–2
	Lock	> 500							0–2
	keine	1.54	18	3	3	5	0	0	0
th_32	$\prec_{\#}$	23.17	12	3	1	3	0	0	0
	Lock	> 500							0–2
	keine	> 500							0–2
th_33	$\prec_{\#}$	23.39	12	3	1	3	0	0	0
	Lock	> 500							0–2
	keine	> 500							0–2
th_35	$\prec_{\#}$	9.76	16	5	0	4	0	0	0
	Lock	> 500							0–2
	keine	> 500							0–2
th_37	$\prec_{\#}$	10.81	6	2	2	3	0	0	0
	Lock	6.80	44	6	6	11	0	0	0
	keine	0.81	11	5	2	3	0	0	0

TABELLE 5.14. Ergebnisse zu `hashtable` mit Gleichheit

ist der Einsatz dieser Ordnung sinnlos. Hingegen scheinen $\prec_{\#}$-geordnete Tableaus in vielen Fällen den Suchraum günstig zu orientieren, auch wenn dies nicht generell der Fall ist.

Zusätzlich zu den obigen Versuchen haben wir noch $\prec_{\#}$-geordnete Tableaus mit Constraints eingesetzt und die Bewertung bezüglich zusätzlicher hierarchischer Informationen modifiziert. In beiden Fällen unterscheiden sich die Ergebnisse aber erstaunlicherweise in keinster Weise von den bisherigen. Ordnungsconstraints für $\prec_{\#}$ sind immer erfüllbar und bringen deswegen keinen Vorteil, selbst dann, wenn die Informationen über die Hierarchie der Spezifikationen mitbenutzt wird.

Daß die $\prec_{\#}$-Ordnungsconstraints in unseren Beispielen nie unerfüllbar werden, hat wahrscheinlich drei verschiedene Ursachen:

1. Die meisten Formeln der KIV-Problem besitzen meist nur ein $\prec_{\#}$-maximales Literal. Der zugehörige Constraint ist also immer erfüllbar.
2. Da alle maximalen Konnektionen einer Formel simultan betrachtet werden, kann es passieren, daß zu jedem maximalen Literal der Formel eine maximale Konnektion existiert. In diesem Fall ist der zugehörige Constraint immer erfüllbar.
3. Wird eine Formel ϕ zu einem Tableauast geholt und kann der zugehörige Teilbaum geschlossen werden, so kann ϕ maximale Konnektionen zu Formeln besitzen, die gar nicht mehr in diesen Teilbaum geholt werden (da dieser geschlossen ist). Auch hier ist der zu ϕ gehörige Constraint immer erfüllbar.

Etwas enttäuschend ist, daß die zusätzlichen hierarchischen Informationen bei $\prec_{\#}$-geordenten Tableaus keine weitere Verbesserung oder zumindestens Veränderung ergaben.

KAPITEL 6

Fazit und Ausblick

> "You mean that's it?" said Ford.
> "That's it."
> "Six by nine. Forty-two."
> "That's it. That's all there is."
>
> Douglas Adams —
> The Restaurant at the End of the Universe

6.1. Fazit

Wir haben in dieser Diplomarbeit gezeigt, daß sich A-geordnete Tableaus, wie sie in (Hähnle & Klingenbeck, 1996) vorgestellt werden, noch verallgemeinern lassen: Wir haben A-Ordnungen durch Auswahlfunktionen ersetzt, die bei Klauseltableaus nur noch stabil bezüglich Variablenumbenennungen zu sein brauchen. Weiterhin beschrieben wir, wie sich Theorien in A-geordnete Tableaus sinnvoll einbetten lassen.

Bei der Verwendung von Ordnungsconstraints stellten wir verschiedene Verfahren vor, um Erfüllbarkeitstests für Constraints zu bekommen. Welches dieser Verfahren sinnvoll ist, hängt sicherlich von den verwendeten Ordnungen ab. Bei einer effizienten Implementierung sind Verfahren, welche die speziellen Eigenschaften einer konkreten Ordnung ausnutzen zu bevorzugen. Aber auch eine automatische Erzeugung von Erfüllbarkeitstests auf Basis von automatischen Beweisern ist sicherlich interessant.

Der Einsatz von $\prec_{\#}$-geordneten Constraints bei Problemen aus der Programmverifikation hat sich als sinnlos herausgestellt, da die Ordnungsconstraints bei dieser Anwendung anscheinend immer erfüllbar sind. Ob dies auch bei anderen Ordnungen der Fall ist, konnten wir leider nicht untersuchen. Trotzdem zeigen unsere Ergebnisse, daß bei den KIV-Problemen $\prec_{\#}$-geordnete Tableaus erfolgreich eingesetzt werden können. Dies muß sich allerdings in der weiteren Praxis bewähren. Vor allen Dingen sollte noch genauer untersucht werden, ob es im Gegensatz zu unseren Erfahrungen doch noch möglich ist, die Hierarchieeigenschaften der KIV-Probleme sinnvoll auszunutzen, so wie das bei der Problemklasse `SET` aus der TPTP-Bibliothek der Fall ist. Eventuell ist es dazu nötig, nicht nur beim Holen von Formeln die Ordnungseinschränkung zu testen und mit Constraints sicherzustellen, sondern auch bei jeder Expansion einer Formel.

6.2. Ausblick

Bei A-geordneten Tableaus gibt es immer noch einige interessante offene Fragen. So ist für eine effiziente Implementierung mit Hilfe der PTTP-Technik die Konnektionsbedingung eine notwendige Voraussetzung. Wie wir an einem Beispiel gesehen haben führt die Regularitätseinschränkung in diesem Zusammenhang zu einem unvollständigem Kalkül (siehe Beispiel Abbildung 3.5, Seite 3.5). Um also an eine PTTP-Implementierung eines A-geordnetem Tableaus zu gelangen, müssen geeignete Modifikationen an dem Kalkül vorgenommen werden.

Da bei A-geordneten Klauseltableaus die Substitutivitätseigenschaft der Ordnung durch eine schwächere Bedingung ersetzt werden kann, steht hier ein größeres Spektrum von Ordnungen und Auswahlfunktionen zur Verfügung. Aus diesem Grund sollte man auch bei geordneten Tableaus für beliebige prädikatenlogische Formeln nach Abschwächungen der Substitutivitätseigenschaft suchen, so daß der Kalkül noch vollständig bleibt.

Bei geordneter Resolution spielen Entscheidbarkeitsfragen eine große Rolle. Auch geordnete Tableaus entscheiden einige Problemklassen. $\mathcal{P}$ sei beispielsweise die Menge aller Klauselmengen M, so daß in jeder Klausel aus M mindestens ein positives Literal vorkommt. Dann ist jedes $M \in \mathcal{P}$ erfüllbar. Zu einer partiellen Ordnung, die positive Literale größer als negative einordnet, existieren keinerlei maximale Konnektionen zwischen Klauseln von M. Deswegen gibt es auch keine Neustartklausel, die für einen initialen Schritt bei geordneten Tableaus nötig ist. Das Beweisverfahren terminiert also. $\mathcal{P}$ ist allerdings eine recht einfache Problemklasse. Ob geordnete Tableaus auch für komplexere Problemklassen ein Entscheidungsverfahren bilden, ist sicherlich nicht so einfach zu klären.

Symbolverzeichnis

$\mathbf{V}$	Menge von Variablen, Seite 3
$\mathbf{F}$	Menge von Funktionssymbolen, Seite 3
$arity$	Stelligkeit von Funktionen, Seite 3
$\mathbf{K}$	Menge von Konstantensymbolen, Seite 3
$\mathbf{T}$	Menge aller Terme, Seite 3
$\mathbf{V}(t)$	Menge aller Variablen eines Terms t, Seite 3
$\tau(t)$	Tiefe eines Terms t, Def. 2.2, Seite 4
$\# : \mathbf{F} \to \mathbb{N}$	Bewertung, Def. 2.3, Seite 4
$Pos(t)$	Positionen eines Terms t, Def. 2.4, Seite 4
$\|p\|$	Länge einer Position p, Seite 4
$t\|_p$	Teilterm von t an Position p, Def. 2.5, Seite 4
$\tau_{\max}(s,t)$	Maximale Tiefe von s in t, Def. 2.7, Seite 4
$\mathbf{P}$	Menge von Prädikatensymbolen, Seite 4
$L : I$	Literal mit Index I, Def. 2.15, Seite 6
$[L_1 : I_1, \ldots, L_n : I_n]$	Indizierte Klausel, Def. 2.16, Seite 6
$\mathbf{C}$	Menge aller Klauseln, Seite 6
$\langle L, L', \sigma \rangle$	$L\sigma = \overline{L'}\sigma$ bei einer Konnektion, Def. 3.2, Seite 9
$\prec_A$	A-Ordnung, Def. 3.3, Seite 10
$\prec_\epsilon$	Leere A-Ordnung, Seite 10
$\prec_\tau$	Termtiefenordnung, Def. 3.5, Seite 11
$\prec_L$	Lexikographische A-Ordnung, Def. 3.6, Seite 11
D_Φ	disjunktiven Pfade von Φ, Def. 3.15, Seite 18
$\mathcal{T}$	Theorie, Def. 3.18, Seite 22
$\mathcal{E}$	Gleichheitstheorie, Seite 22
$\langle E, s, t \rangle$	E-Unifikationsproblem, Def. 3.24, Seite 31
$\sqsubset$	Nicht hebbare Ordnung, Def. 3.27, Seite 33
$\sqsubset_\#$	Nicht hebbare Ordnung, Def. 3.28, Seite 34
$\doteq$	Syntaktische Gleichheit, Def. 4.2, Seite 39
$atom(L)$	Atom eines Literals, Def. 4.3, Seite 39
$\max_\prec(A, M)$	A maximal in Literalmenge M, Def. 4.3, Seite 39
$\max_\prec(A, \phi)$	A maximal in ϕ, Def. 4.3, Seite 39
$\max_\prec(M, \phi)$	Ex. Literal in M maximal in ϕ, Def. 4.3, Seite 39
τ_L	Transformation in PL1-Formeln, Def. 4.4, Seite 48
$M_{\prec_L}$	Def. 4.5, Seite 49
$\prec_\#$	Bewertungsordnung, Def. 4.6, Seite 50
$\prec_{\text{lock}}$	Lock-Einschränkung, Seite 56

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Baumgartner, P. 1996a (Mai). Persönliche Mitteilung.

Baumgartner, P. 1996b. Linear and Unit-Resulting Refutations for Horn Theories. *Journal of Automated Reasoning.* Noch unveröffentlicht.

Baumgartner, P. & Furbach, U. 1994. Model Elimination without Contrapositives. *Seiten 87–101 in:* Bundy, A. (Hrsg.), *'Automated Deduction – CADE-12'.* LNAI, vol. 814. Springer.

Baumgartner, Peter, Furbach, Ulrich & Petermann, Uwe. 1992. *A Unified Approach to Theory Reasoning.* Forschungsbericht 15/92. Universität Koblenz.

Beckert, Bernhard. 1993 (Juli). *Ein vervollständigungsbasiertes Verfahren zur Behandlung von Gleichheit im Tableaukalkül mit freien Variablen.* Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe.

Beckert, Bernhard. 1994. A Completion-Based Method for Mixed Universal and Rigid E-Unification. *Seiten 678–692 in:* Bundy, A. (Hrsg.), *Proceedings, 12th International Conference on Automated Deduction (CADE), Nancy, France.* LNCS 814. Springer.

Beckert, Bernhard. 1996. Equality and Other Theory Inferences. *In:* D'Agostino, M., Gabbay, D., Hähnle, R. & Posegga, J. (Hrsg.), *Handbook of Tableau Methods.* Kluwer, Dordrecht. Noch unveröffentlicht.

Beckert, Bernhard & Pape, Christian. 1996. Incremental Theory Reasoning Methods for Semantic Tableaux. *Seiten 93–109 in: Proceedings, 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods.* LNCS, vol. 1071. Springer.

Beckert, Bernhard, Hähnle, Reiner & Schmitt, Peter H. 1993. The Even More Liberalized δ-Rule in Free Variable Semantic Tableaux. *Seiten 108–119 in:* Gottlob, Georg, Leitsch, Alexander & Mundici, Daniele (Hrsg.), *3rd Kurt Gödel Colloquium (KGC).* LNCS 713. Brno, Czech Republic: Springer.

Beckert, Bernhard, Hähnle, Reiner, Geiss, Karla, Oel, Peter, Pape, Christian & Sulzmann, Martin. 1996a. *The Many-Valued Tableau-Based Theorem Prover ${}_3T^AP$, Version 4.0.* Interner Bericht 3/96. Universität Karlsruhe, Fakultät für Informatik.

Beckert, Bernhard, Hähnle, Reiner, Oel, Peter & Sulzmann, Martin. 1996b. The Tableau-based Theorem Prover ${}_3T^AP$, Version 4.0. *In:* McRobbie, Michael (Hrsg.), *Proceedings, 13th International Conference*

on Automated Deduction (CADE), New Brunswick, USA. LNCS. Springer. Noch unveröffentlicht.

Bibel, Wolfgang, Brüning, Stefan, Egly, Uwe & Rath, Thomas. 1994. KoMeT. *Seiten 783–787 in:* Bundy, A. (Hrsg.), *Proceedings, 12th International Conference on Automated Deduction (CADE), Nancy, France.* LNCS 814. Springer.

Boyer, Robert S. 1971. *Locking: A Restriction of Resolution.* Ph.D. thesis, The University of Texas at Austin.

Brand, D. 1975. Proving Theorems with the Modification Method. *SIAM Journal on Computing*, **4**(4), 412–430.

Carroll, L. 1986. *Lewis Carroll's Symbolic logic : part I, Elementary, part II, Advanced.* C. N. Potter, New York, NY.

de Nivelle, Hans. 1996. Resolution Games and Non-Liftable Resolution Orderings. *Seiten 1–20 in: Collegium Logicum. Annals of the Kurt-Gödel-Society*, vol. 2. Springer-Verlag, Wien New York.

Fermüller, C., Leitsch, A., Tammet, T. & Zamov, N. 1993. *Resolution Methods for the Decision Problem.* LNAI 679. Springer.

Fitting, Melvin C. 1990. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, New York.

Fitting, Melvin C. 1996. *First-Order Logic and Automated Theorem Proving.* Zweite Auflage. Springer-Verlag, New York.

Gallier, Jean H., Narendran, Paliath, Plaisted, David A. & Snyder, Wayne. 1990. Rigid E-Unification: NP-Completeness and Application to Equational Matings. *Information and Computation*, 129–195.

Gallier, Jean H., Narendran, Paliath, Raatz, Stan & Snyder, Wayne. 1992. Theorem Proving Using Equational Matings and Rigid E-Unification. *Journal of the ACM*, **39**(2), 377–429.

Hähnle, Reiner & Klingenbeck, Stefan. 1996. A-Ordered Tableaux. *Journal of Logic and Computation.* Noch unveröffentlicht.

Jeffrey, Richard C. 1967. *Formal Logic. Its Scope and Limits.* McGraw-Hill, New York.

Klingenbeck, Stefan & Hähnle, Reiner. 1994. Semantic Tableaux with Ordering Restrictions. *Seiten 708–722 in:* Bundy, Alan (Hrsg.), *Proc. 12th Conference on Automated Deduction CADE, Nancy/France.* LNAI 814. Springer.

Kowalski, R. & Hayes, P. J. 1969. Semantic Trees in Automatic Theorem Proving. *Seiten 87–101 in: Machine Intelligence*, vol. 4. Edinburgh University Press. Nachdruck in (Siekmann & Wrightson, 1983a).

Leitsch, A. 1996. *The Resolution Calculus.* Monographs in Theoretical Computer Sciences. Springer-Verlag. Noch unveröffentlicht.

Letz, R., Schumann, J., Bayerl, S. & Bibel, W. 1992. SETHEO: A High Performance Theorem Prover. *Journal of Automated Reasoning*, **8**(2), 183–212.

LETZ, REINHOLD. 1993 (Juni). *First-order calculi and proof procedures for automated deduction.* Ph.D. thesis, TH Darmstadt.

LOVELAND, D. 1968. Mechanical Theorem Proving by Model Elimination. *Journal of the ACM*, **15**(2).

MARTELLI, A. & MONTANARI, U. 1982. An Efficient Unification Algorithm. *ACM Trans. Programming Languages and Systems*, 258–282.

MASLOV, S. JU. 1971. Proof-Search Strategies for Methods of Resolution Type. *Seiten 77–90 in: Machine Intelligence*, vol. 6. Elsevier.

NIEUWENHUIS, R. & RUBIO, A. 1995. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, **19**, 321–351.

PETERMANN, U. 1992. How to Build In an Open Theory into Connection Calculi. *Journal on Computers and Artificial Intelligence*, **2**, 105–142.

PETERSON, G. 1983. A technique for establishing completness results in theorem proving with equality. *SIAM Journal of Computation*, **12**(1), 83–100.

REIF, W. 1992. Verification of Large Software Systems. *In:* SHYAMASUNDAR (Hrsg.), *Foundations of Software Technology and Theoretical Computer Science, New Dehli, India.* Springer, LNCS.

ROBINSON, J. A. 1965a. Automatic Deduction with Hyper-Resolution. *Int. Journal of Computer Math.*, **1**, 227–234. Nachdruck in (Siekmann & Wrightson, 1983b).

ROBINSON, J. A. 1965b. A Machine-Oriented Logic Based on the Resolution Principle. *JACM*, **12**(1), 23–41. Nachdruck in (Siekmann & Wrightson, 1983b).

SCHRIJVER, ALEXANDER. 1986. *Theory of Linear and Integer Programming.* Wiley-Interscience series in discrete mathematics. John Wiley & Sons.

SIEKMANN, JÖRG & WRIGHTSON, GRAHAM (Hrsg.). 1983a. *Automation of Reasoning: Classical Papers in Computational Logic 1967–1970.* Bd. 2. Springer.

SIEKMANN, JÖRG & WRIGHTSON, GRAHAM (Hrsg.). 1983b. *Automation of Reasoning: Classical Papers in Computational Logic 1957–1966.* Bd. 1. Springer.

SIEKMANN, JÖRG H. 1989. Universal Unification. *Journal of Symbolic Computation*, **7**(3/4), 207–274. Earlier version in *Proceedings, 7th International Conference on Automated Deduction (CADE), Napa, FL. USA*, LNCS 170, Springer, 1984.

SMULLYAN, RAYMOND M. 1995. *First-Order Logic.* Zweite überarbeitete Auflage. Dover Publications, New York. Erste Auflage 1968 im Springer-Verlag.

STEINBACH, J. & ZEHNTER, M. 1990. *Vade-mecum of Polynomial Orderings.* Report SR-90-03. Fachbereich Informatik, Universität Kaiserslautern, Germany.

STICKEL, MARK E. 1985. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, **1**, 333–355.

STICKEL, MARK E. 1988. A Prolog Technology Theorem Prover. *Seiten 752–753 in:* LUSK, E. & OVERBEEK, R. (Hrsg.), *9th International Conference on Automated Deduction (CADE)*. LNCS. Argonne, IL, USA: Springer.

SULZMANN, MARTIN. 1995 (Juli). *Ausnutzung von KIV-Spezifikationen in 3TAP*. Studienarbeit, Fakultät für Informatik, Universität Karlsruhe.

SUTCLIFFE, GEOFF, SUTTNER, CHRISTIAN & YEMENIS, THEODOR. 1994. The TPTP Problem library. *Seiten 252–266 in:* BUNDY, ALAN (Hrsg.), *Proc. 12th Conference on Automated Deduction CADE, Nancy/France*. LNAI. Springer.

Index