SEMINAR

INFERENZMETHODEN

Model Elimination and Connection Tableau Procedures

Andreas Glaser 710909

INSTITUT FÜR INFORMATIK
UNIVERSITÄT POTSDAM

FACHGEBIET THEORETISCHE INFORMATIK
PROFESSOR CHRISTOPH KREITZ
SOMMERSEMESTER 2005

Model Elimination and Connection Tableau Procedures

Andreas Glaser (aglaser@rz.uni-postdam.de)

Institut für Informatik, Universität Potsdam

Zusammenfassung. Logisches Schließen hilft den Menschen unter anderem beim Lösen von Problemen, durch vorhandenes Wissen kann neues hergeleitet werden. Wir benutzen in unserem Leben eine Vielzahl von verschiedenen, nicht immer streng formal-logisch korrekten, Schlußweisen: die Deduktion, vom Allgemeinen auf das Spezielle, seine Umkehrung die Induktion oder auch die Abduktion, „Wo Rauch ist, da ist auch Feuer“. Das Gebiet der Inferenzsysteme, ein Teilgebiet der Künstlichen Intelligenz (KI), behandelt die verschiedenen Schlußweisen und stellt Methoden zur automatisierten Verarbeitung von Wissen mittels logischer Schlüsse (Inferenzmethoden) bereit. Basierend auf dem Vortrag vom 23.09.2005 im Rahmen des Seminars „Inferenzmethoden“ an der Universität Potsdam werden in diesem Dokument die Themen *Strukturelle Verfeinerung des Klausel Tableauxs* sowie *Globale Pruning Methoden in der Modellelimination* behandelt.

Keywords: Inferenz, Inference, Reasoning.

1 Grundlagen / Definitionen

Inferenzkalküle: Kalküle sind Hilfsmittel zur Beweissuche, es erfolgt eine syntaktische Manipulation formaler Ausdrücke unter Berücksichtigung der Semantik. Es werden Axiom-orientierte Kalküle (Frege-Hilbert-Kalkül), konnektivorientierte Kalküle (Tableaux) und maschinennahe Kalküle (Resolutions- / Konnektionskalküle) unterschieden.

Tableaux-Kalküle: Kompakte Variante des Sequenzenkalküls. Einführung der Polarität, Gruppierung der Regeln in Gruppen: Unterscheidung von Alpha-, Beta-, Gamma- und Delta-Regeln. Eigenschaften:

- Widerspruchsbeweis: Zeige, dass alle möglichen Konsequenzen von ¬F zum Widerspruch führen.
- Top-Down Verfahren: Analyse der Behauptung
- Endlichverzweigter, mit Formeln markierter Baum
- Tableaux ist geschlossen, wenn alle Zweige ein komplementäres Formelpaar enthalten

Klauseltableaux: Ein Klauseltableaux für eine Klauselmenge $S = \{C_1, \ldots, C_n\}$ ist ein Tableaux mit freien Variablen für S. Es dürfen nur die Beta-, Gamma- und Abschlussregel angewendet werden.

Konnektionstableaux: Ist ein Klauseltableaux, worin jeder innere mit L markierte Knoten (außer $\top$) ein mit L’ markiertes Blatt als unmittelbaren Nachfolger hat, so dass (L, L’) komplementäres Paar ist. Die für den Abschluss eines Zweiges nötigen Literale sind direkt hintereinander.

2 Motivation / Vom Kalkül zur Beweisprozedur

Kalküle sind keine Beweismethode: Die Vollständigkeit der Tableauxregeln garantiert nur die Existenz eines geschlossenen Tableauxs, benötigt wird aber eine Beweisprozedur, um den Beweis zu finden. Eine effiziente Implementierung dieser Prozedur ist unumgänglich, da die Rechenkapazität moderner Computer begrenzt ist. Während der Beweissuche stellen sich mehrere Fragen:

- Nächster zu betrachtender Ast?
- Abschluss oder Erweiterung eines Astes?
- Falls Abschluss: Mit welchen Literalpaar?
- Falls Erweiterung: Mit welcher Formel?

Bei den letzten drei Punkten entsteht ein Bevorzugungsproblem, bei den zu betrachtenden Ästen nicht, da jeder Ast, egal in welcher Reihenfolge, abgeschlossen werden muss. Eine schlechte Wahl kann die Beweissuche aufgrund der Entstehung von Redundanz wesentlich in die Länge ziehen. Auch vor der eigentlichen Beweissuche gibt es Möglichkeiten einer „Vorbearbeitung“ der Eingabe, um einen Beweis in möglichst geringer Zeit zu finden. Die in der Modellelimination entwickelten Methoden zur Eliminierung von Redundanz werden in drei Kategorien klassifiziert:

1 *Lokale Methoden:* Auch als strukturelle Methode bezeichnet: Betrachtet wird ein Tableaux als eine Deduktion. Einige Deduktionen können aufgrund ihrer internen Struktur als redundant identifizieren werden. Die Eigenschaft der Regularität eines Tableaux ist die wichtigste Technik.

2 *Globale Methoden:* Bestimmte Deduktionen sind in im Zusammenspiel mit anderen Deduktionen redundant. Ein Beispiel für den globalen Ansatz ist Failure Caching.

3 *Länge der Beweise:* Verglichen mit anderen Kalkülen können kleine Beweise trotzdem sehr umfangreich werden, da die Modell-elimination *cut-free* ist und es sich bei den Deduktionsobjekten um Bäume handelt. Ein Beweisbaum kann einen Unterbeweis öfters beinhalten. Diese Klasse wird in diesem Dokument nicht behandelt.

3 Strukturelle Verfeinerungen des Klauseltableauxs

Verfeinerungen sind Einschränkungen der Menge der Klauseltableauxs, so dass es trotzdem für jede unerfüllbare Klauselmenge ein geschlossenes Tableaux gibt. Es werden bestimmte Inferenzschritte verboten, wenn sie ein Tableaux bestimmter Struktur erzeugen, um so die Redundanz aus dem Suchbaum zu entfernen. Die Konnektionsbedingung des Klauseltableauxs ist eine solche Einschränkung mit der Auswirkung, dass die Verzweigung des Suchbaumes unter Beibehaltung von Vollständigkeit als auch von Korrektheit verringert wird. Die Gesamtanzahl der zu betrachtenden Tableauxs nimmt ab. Diese Methoden der Redundanzeliminierung werden lokale Pruning-Techniken genannt in dem Sinne, dass sie nur auf ein Tableaux angewendet werden können.

3.1 Regularität

In einen offenen Zweig soll eine Formel höchstens einmal vorkommen, so dass bei Klauseltableauxs durchaus nutzlose Einsetzungen für freie Variablen verhindert werden können.

Regulär: Ein Klauseltableaux heißt regulär, wenn auf keinem seiner Äste dasselbe Literal mehr als einmal vorkommt.

Beispiel: S = { {p(0)}, {¬p(x), p(s(x))}, {¬p(s(s(0)))} }

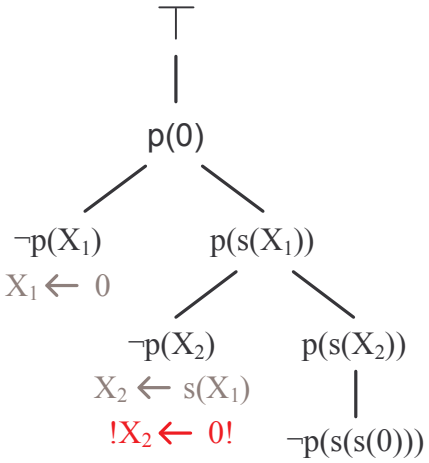

Abbildung 1: Regularität

Wenn man für die Variable X_2 die Substitution 0 benutzen würde, wäre die Eigenschaft der Regularität verletzt. Regularität verhindert redundante identische Instanziierung von Variablen.

Konsequenzen:

1. Jedes allgemeine Klauseltableaux von minimaler Größe ist auch regulär. Regularität wahrt minimale Beweislänge.
2. Mit Regularität wird der Suchraum einer Formel endlich.

Die zweite Bedingung gilt auch für Konnektionstableaux, die erste Bedingung, das minimale Konnektionstableauxs regulär sind, gilt nicht. Experimente zeigen aber, dass dieser

theoretische Nachteil von dem starken Vorteil bei der Beweissuche unter Nutzung der Regularität mehr als ausgeglichen wird. Regularität ist quasi für jede Modelleliminations-Prozedur unabdingbar.

3.2 Tautologie Elimination

Sinnvoll ist es, bestimmte Klauseln von der Eingabemenge zu löschen, da diese bei der Suche eines Widerspruches redundant sind. Tautologien werden in einer Preprocessing Phase erkannt bevor der eigentlicher Beweiser loslegt. Es kann aber passieren das Tautologien dynamisch erzeugt werden: ¬P(x, y) v ¬P(y, z) v P(x, z). Diese Klausel wird in einem Extensionsschritt verwendet. Die Variablen y und z werden mit dem gleichen Term t instanziiert. Dann entsteht eine Instanz mit einer Tautologie: ¬P(x, t) v ¬P(t, t) v P(x, t). Tautologische Klauseln sind in einem Formelset nicht relevant, Konnektionstableaux mit tautologischen Tableaux-Klauseln brauchen bei der Beweissuche nicht berücksichtigt werden. Die jeweiligen Tableauxs sowie jede Erweiterung davon können vernachlässigt werden.

3.3 Tableaux Klausel Subsumierung

Klauseln, die von anderen subsumiert werden, können gelöscht werden.

Subsumierung: Gegeben sind zwei Klauseln c_1 und c_2, c_1 subsumiert c_2 wenn es eine Substitution σ gibt, so dass die Literale in $c_1\sigma$ eine Teilmenge der Menge der Literale von c_2 sind.

Beispiel: Gegeben sind ¬P(x, y) v ¬P(y, z) v P(x, z) sowie P(a, b). x wird durch a und z durch b instanziiert. Daraus folgt: ¬P(a, y) v ¬P(y, b) v P(a, b). Das Ergebnis wird von P(a, b) subsumiert.

3.4 Starke Konnektivität

Wenn man Formeln in Klauselform umwandelt, werden manchmal neue Prädikate eingeführt um die Formeln zu kürzen. Beispiel: Wir ersetzen die Konjunktion der Literale a ∧ b durch d, also a ∧b <=> d. So erhält man ¬a v ¬b v d, a v ¬d, b v ¬d. Jede Resolvente zwischen den drei Klauseln ist eine Tautologie. Bezogen auf Tableauxs: Wenn eine der Klauseln unter einer der anderen ist, handelt es sich um eine „versteckte“ Form der Tautologie.

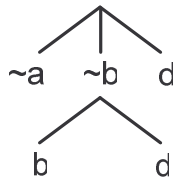

Abbildung 2: Versteckte Tautologien im Tableaux

Starke Konnektivität: Zwei Klauseln c_1 und c_2 sind stark konnektiert wenn es eine Substitution σ gibt, so dass $c_1\sigma$ genau ein Literal enthält dessen Komplement in $c_2\sigma$ ist.

3.5 Relevanzinformationen

Relevanzinformationen minimieren die möglichen Startklauseln.

Essentiell: Eine Formel F ist in einem Set S essentiell falls S unerfüllbar ist und S \ {F} erfüllbar.

Relevant: Eine Formel F ist in einem Set S relevant falls F in einer Teilmenge von S essentiell ist.

Minimal Unerfüllbar: Ein unerfüllbares Set S von Formeln ist minimal unerfüllbar, falls jede Formel in S essentiell ist.

4 Globale Pruningmethoden in der Modellelimination

Der Modelleliminationsbeweiser nutzt eine Art beschrifteter Klausel, genannt *Chain*. Diese Beschriftung enthält Informationen über bereits durchgeführte Inferenzen in einer Sequenz von Inferenzschritten. Mithilfe dieser Informationen kann man die Redundanz in einem Beweis senken.

Der Begriff *Pruning* ist eine andere Bezeichnung für *Backjumping* oder *intelligentes Backtracking*.

4.1 Matings Pruning

Ein *Mating* ist eine Menge von Konnektionen und kann mit mehreren verschiedenen Tableauxs in Verbindung gebracht werden. Dieses Konzept ermöglicht einen abstrakteren Blick auf die Beweissuche und erlaubt die Gruppierung von Tableauxs in Äquivalenzklassen. Die Konstruktion aller Tableauxs einer Klasse ist unter bestimmten Umständen nicht nötig, es reicht die Konstruktion eines einzigen Tableaux. Zur Veranschaulichung wird das folgende Set mit Klauseln betrachtet:

Beispiel 4.1: S = {¬P_1 v ¬P_2, ¬P_1 v P_2, P_1 v ¬P_2, P_1 v P_2}

Wie in der Abbildung 3 zu sehen hat das Set vier geschlossene reguläre Konnektionstableauxs mit der negativen Startklausel ¬P_1 v ¬P_2. Alle Tableauxs haben dasselbe *Mating* mit genau sechs Konnektionen. Bestimmte Tableauxs sind Permutationen in denen sich der Reihenfolge der Konnektionen unterscheidet. Es genügt die Betrachtung eines einzigen Tableauxs einer solchen Klasse, so dass eine mehrfache redundante Betrachtung vermieden wird.

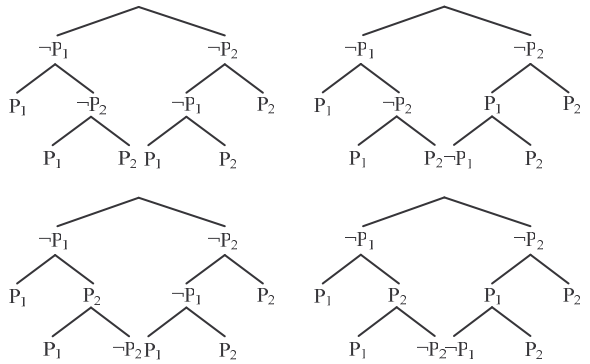

Abbildung 3: Vier geschlossene Tableauxs für dasselbe Mating

Es stellt sich die Frage, wie genau diese Form von Redundanz vermieden werden kann. Ein genereller Ansatz wäre das Speichern aller *Matings* bei der Beweissuche, so dass alle Tableauxs mit einem solchen *Mating* ignoriert werden könnten. Dieser Ansatz ist ineffizient, da er einen enormen Speicherplatz benötigt.

Bei näherer Betrachtung der Quelle kann die Redundanz identifiziert werden: Eine bestimmte Konnektion kann sowohl im Extensions- als auch im Reduktionsschritt benutzt werden. Die Kombinationen führen zu einer Explosion der Möglichkeiten. Durch das Einführen einer Ordnung für das Auftreten von Literalen im Eingabeset sollen Reduktionsschritte vermieden werden. Die Ordnung wird vom Literal des Eingabesets an den Tableauxknoten vererbt. Wie in [Letz 1998b] gezeigt erhält man eine super-exponentielle Reduktion des Suchraumes, nachteilig sind Probleme bei der gleichzeitigen Verwendung von anderen Pruningtechniken.

Matings Pruning ist nicht vereinbar mit der Bedingung der starken Konnektivität.

Gegenbeispiel: S = {P v Q(a), P v ¬Q(a), ¬P v Q(a), ¬P v ¬Q(x)}

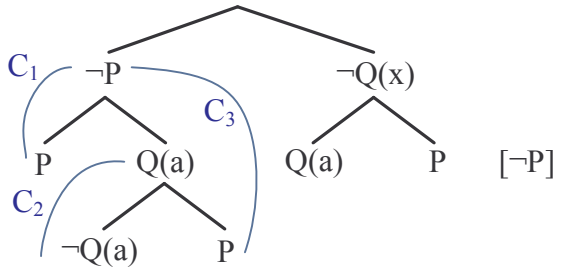

Abbildung 4: Deduktion für Beispiel 4.1

Wir nehmen die relevante Formel ¬P v ¬Q(x) als Startklausel, gehen in die erste Klausel des Sets S (C_1), danach in die zweite durch Extension und führen danach einen Reduktionsschritt durch. Das geschlossene Subtableaux hat das Mating {C_1, C_2, C_3}. Jeder Extensionsschritt auf der rechten Seite mit dem Subgoal ¬Q(x) verletzt die Bedingung der starken Konnektivität. (Begründung: P v Q(a) sowie ¬P v ¬Q(x) haben mehr als genau ein Literal, was in der einen Klausel komplementär zur anderen Klausel ist.) Für das in den eckigen Klammern stehende Literal ¬P wäre die Eigenschaft der starken Konnektivität erfüllt.

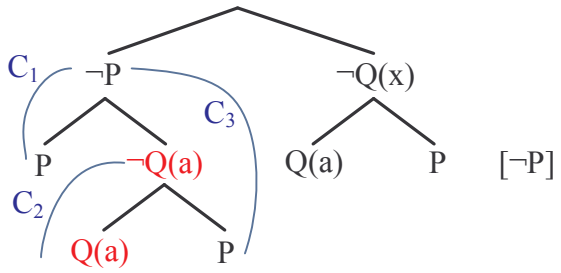

Abbildung 5: Deduktion für Beispiel 4.1

Es erfolgt ein Backtracking bis zur obersten Klausel. Es wird nun nicht mit der ersten Klausel des Sets S angefangen, sondern mit der zweiten und gehen dann mit einem Extensionsschritt in die erste Klausel des Sets S. Das Matings Pruning verhindert einen Reduktionsschritt mit P, da man ein Subtableaux mit dem Mating {C_3, C_2, C_1} erhalten würde. Der Extensionsschritt ist ebenfalls wegen der Regularitätsbedingung nicht möglich, also würde die Reduktion fehlschlagen und fälschlicherweise ausgeben, dass das Set erfüllbar ist. Matings Pruning ist daher nicht vereinbar mit der Bedingung der starken Konnektivität.

4.2 Tableaux Subsumierung

Die bisher besprochene Subsumierung bezog sich auf Startklauseln und Klauseln in einem Tableaux, nun wird die Subsumierung zwischen Klauseln in verschiedenen Tableauxs betrachtet.

Formelbaum Kontraktion: Ein (Formel-) Baum T wird Kontraktion eines (Formel-) Baumes T' genannt, wenn T' durch T erzeugt wird in dem man an T n (Formel-) Bäume anhängt. Die n (Formel-) Bäume dürfen nicht an die Blätter von T angehängt werden. N >= 0.

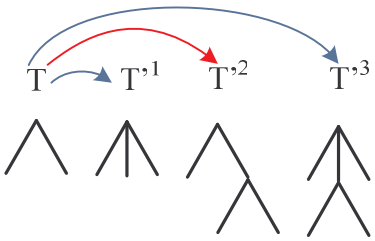

Abbildung 6: Illustration von drei Kontraktionen

In Abbildung 6 ist der Baum T eine Kontraktion von sich selbst, von T'^{1} sowie von T'^{3}, aber nicht von T'^{2}. T'^{2} selber ist eine Kontraktion von T'^{3}. Der letzte Baum hat also zu allen anderen eine Kontraktionsrelation.

Formelbaum Subsumierung: Ein Formelbaum T subsumiert einen Formelbaum T', wenn eine Formelbaum Kontraktion von T' eine Instanz von T ist.

Da die Subsumierung von Tableaux nicht genügend Potential für die Reduzierung des Suchraumes bietet, wird die folgende Form der Subsumierung favorisiert:

Subsumption Deletion: Für jedes Paar von verschiedenen Knoten N und N' in einem Suchbaum $\mathcal{T}$ gilt: Wenn der Zielbaum des Tableauxs bei N den Zielbaum des Tableauxs bei N' subsumiert, dann wird der ganze Unterbaum des Suchbaumes mit der Wurzel N' aus $\mathcal{T}$ gelöscht.

Ähnlich wie bei der Resolution führen bestimmte Verfeinerungen des Kalküls zusammen mit der *Subsumption Deletion* zu Problemen. So auch beim Tableauxkalkül:

Kompatibilität mit Subsumierung: Ein Tableauxkalül ist genau dann Kompatibel mit Subsumierung, wenn jeder seiner Tableaux-Suchbäume $\mathcal{T}$ die folgende Eigenschaft besitzt: N dominiert einen Nachfolgeknoten. Für frei wählbare Paare N und N' in $\mathcal{T}$, wenn der Zielbaum S des Tableauxs T bei N den Unterzielbaum S' des Tableauxs T' bei N' subsumiert und wenn N' den Nachfolgeknoten dominiert, dann dominiert auch N einen Nachfolgeknoten.

Das (Konnektions-) Tableauxkalkül ist mit Subsumierung kompatibel, Probleme können bei gleichzeitiger Verwendung der Regularität auftreten.

4.3 Failure Caching

Caching ist ein Mechanismus, welcher normale Suchmechanismen bei günstigeren Kosten unter Beibehaltung der Ergebnisse ersetzt. Caching reduziert die Anzahl von Inferenzschritten durch das ersetzten von Suchschritten, indem bereits erreichte Ziele angeschaut werden.

Beobachtungen zeigen, dass in der Praxis Fälle von Subsumierung unvermeidlich auftauchen. Dies legt die Nummerierung von Tableaux nahe, so dass Fälle von Subsumierung auch tatsächlich entdeckt werden können. Das kann mit einer Beweisprozedur erreicht werden, die Tableauxs erzeugt und den Suchbaum mittels Breitensuche durchsucht. Die explizite Nummerierung von Tableauxs ist in der Praxis aufgrund der Begrenzung von Ressourcen nicht möglich. Eine implizite Nummerierung durch die Benutzung von iterativer Tiefensuche ist vorteilhafter, da nur ein Tableaux im Speicher verwaltet werden muss. Die Implementierung der Alternative ist schwer, deswegen werden zwei Konzepte diskutiert wie man eingeschränkte *Subsumption Deletion* implementieren kann.

Die erste Möglichkeit wird durch Nutzung von intelligentem Backtracking realisiert. Diese Verfeinerung des Standard Prolog Backtracking sucht den Zweig nach der Substitution ab, welche ein Subgoal davon abhält, gelöst zu werden. Dies gestaltet sich aber schwierig bei nicht Hornklauseln bzw. bei der Benutzung anderer Pruningtechniken.

Die zweite Möglichkeit wird als Failure Caching bezeichnet. Man will die wiederholte Lösung von Subgoals vermeiden, welche für einen Zweig die gleiche oder eine speziellere Substitution verwenden. Es gibt zwei Ansätze, einen mit dauerhaften und einen mit temporärem Speicher. Die letzte Methode (auch lokales *Failure Caching* genannt) wird beschrieben, weil es in der Praxis erfolgreicher ist. Annahme: Es werden nur Funktionen (Depth-First-Branch-Selection-Tiefensuche) genutzt, welche Zweige zuerst wählen, die tiefer liegen, also nicht in die Breite gehen.

Solution-, Failure Substitution: Gegeben sind ein Tableaux-Suchbaum $\mathcal{T}$ sowie eine Funktion mit Tiefensuche. N ist ein Knoten in $\mathcal{T}$, T ist das Tableaux bei N und N das gewählte Subgoal in T.

1. N' ist ein Knoten im Tableaux T' in einem Suchbaum $\mathcal{T}$. N dominiert N' so dass alle Zweige durch N geschlossen sind. $\sigma' = \sigma_1 \ldots \sigma_n$ ist eine Komposition von Substitutionen angewandt auf dem Weg durch das Tableaux T von N zu N'. Dann wird die Substitution $\sigma = \{x/x\sigma' \in \sigma' : x \text{ kommt in T vor}\}$ als Solution Substitution von N, Lösung von N bei N über N', genannt.
2. Ist $\mathcal{T}'$ ein initialer Teil des Suchbaumes $\mathcal{T}$, welcher keinen Beweis bei N' oder darunter hat, dann wird die Lösung σ eine Fehlersubstitution für N bei N über N' in $\mathcal{T}'$ genannt.

Wenn also eine Lösung des Subgoals N mit der Substitution σ den Rest des Tableauxs unter einer bestimmten Größe nicht lösen kann, dann wird die „*Solution Substitution*" „*Failure Substitution*" genannt.

Generation, Application und Deletion einer Failure Substitution: $\mathcal{T}$ ist ein endliches initiales Segment eines Tableaux-Suchbaumes.

1. Wenn ein Subgoal in einem Tableaux T bei dem Knoten N in $\mathcal{T}$ über einen Knoten N' geschlossen wurde, dann wird die Lösung σ im Tableauxknoten N gespeichert. Wenn das Tableaux bei N' in $\mathcal{T}'$ nicht geschlossen werden kann und die Beweisprozedur per Backtracking rückwärts geht über N', dann wird σ in eine Fehlersubstitution umgewandelt.
2. Für jede alternative Lösung des Tableauxs T unter dem Knoten N, für die eine Substitution $r = r_1 \ldots r_n$ errechnet wird, so dass eine Failure Substitution im Knoten N allgemeiner als r ist, wird sofort Backtracking ausgeführt.
3. Wenn per Backtracking über einen Knoten N zurückgegangen wird, werden alle Failure Substitutionen bei N gelöscht.

Beispiel: S = { ¬P(x) v ¬Q(x) v ¬R(x), P(a), P(z) v ¬Q(z), Q(a), Q(b), R(b) }

Eine mögliche Tableauxkonstruktion wird vorgestellt, wobei die Wahl der Subgoals durch folgende Regel geschieht: Tiefensuche, links vor rechts.

Legende:

UF – Unifikationsfehler

Step x – Backtracking Schritt x

	Aktion	Subgoal	Sub	FSub
T_0	Start	¬P(x), ¬Q(y), ¬R(x)	-	-
T_1	P(a)	¬Q(y), ¬R(x)	{x/a}	-
T_2	Q(a)	¬R(a)	{y/a, x/a}	-
	UF	¬R(a)	{y/a, x/a}	-
	Step 2	¬Q(y), ¬R(a)	{x/a}	{y/a}
T_3	Q(b)	¬R(a)	{x/a, y/b}	{y/a}
	UF	¬R(a)	{x/a, y/b}	{y/a}
	Step 3	¬Q(y), ¬R(a)	{x/a}	{y/a}
	Step 1	¬P(x), ¬Q(y), ¬R(x)	-	{x/a}
T_4	P(x) v ¬Q(x)	¬Q(x), ¬Q(y), ¬R(x)	{z/x}	{x/a}
T_5	Q(a)	¬Q(y), ¬R(x)	{z/a, x/a}	{x/a}
T_5	T_5 subsumiert T_1	¬Q(y), ¬R(x)	{z/a, x/a}	{x/a}
	Step 5	¬Q(x), ¬Q(y), ¬R(x)	{z/x}	{x/a}
T_6	Q(b)	¬Q(y), ¬R(b)	{z/b, x/b}	{x/a}
T_7	Q(a)	¬R(b)	{z/b, x/b, y/a}	{x/a}
T_8	R(b)		{z/b, x/b, y/a}	{x/a}

Tabelle 1: Beweissuche durch Failure Substitution

In Schritt 1 wird das Subgoal ¬P(x) gelöst durch P(a), mit dieser Substitution können die anderen Subgoals aber nicht mehr gelöst werden. Also kommt es zum Backtracking von Schritt eins, die Fehlersubstitution {x/a} wird beim Subgoal ¬P(a) gespeichert. Der Vorteil der Fehlersubstitution zeigt sich im Schritt 5, wo die Fehlersubstitution allgemeiner als die errechnete Tableauxsubstitution ist (rot markiert): T_1 subsumiert T_5. Ohne diese Methoden müssten Schritt zwei bis vier wiederholt werden.

In Schritt drei wird eine Fehlersubstitution gelöscht. Die beschriebene Lösung kann aber Vollständigkeit verhindern. Deswegen wird folgende Annahme gemacht:

Annahme: $\mathcal{T}$ ist das initiale Segment eines Konnektionstableaux-Suchbaumes definiert durch eine Subgoal Selektionsfunktion und einer Depth-Bound (Tiefenbegrenzung) oder einer Klausel abhängigen Depth-Bound mit der

Größenlimitierung k. Angenommen eine Fehlersubstitution wurde am Knoten N in einem Tableaux T generiert. Ist T_c ein geschlossenen Konnektionstableaux unter einem Suchknoten N und r die Komposition von Substitutionen angewandt bei der Generierung von T_c aus T, dann ist die Fehlersubstitution σ nicht allgemeiner als r.

Die Failure Caching Methode muss daher angepasst werden, wenn man sie unter der obigen Annahme durchführt. Während für die Klausel abhängigen Depth-Bounds die oben beschriebene Methode genutzt werden kann, muss man bei der Inference-Bound (Begrenzung der Inferenzen) aufpassen, um nicht die Vollständigkeit zu verlieren. Es kann passieren, dass die Lösung eines Subgoal (mit den dazugehörigen Substitution σ) fast alle möglichen Inferenzen aufbraucht, so dass für die verbleibenden Subgoals nicht genügend Inferenzen übrig bleiben. Es könnte auch ein anderer, kleinerer Lösungsbaum des Subgoales einer Substitution, welche die Lösung der verbleibenden Subgoals ermöglicht, existieren. Die Failure Substitution würde diese Lösung verhindern.

Um also Vollständigkeit zu gewähren muss man die Zahl der Inferenzen, die für die Lösung eines Subgoals gebraucht wurde, an die Failure Substitution knüpfen. Nur wenn der Lösungsbaum, welcher später generiert wird, größer oder gleich dem mit σ assoziierten ist, kann σ für das Pruning verwendet werden.

Des Weiteren können vergleichbare Phänomene wie oben genannt auftreten, wenn man Failure Caching zusammen mit lokalen Pruningmethoden wie Regularität, Tautologie Elimination oder Tableaux Klausel Subsumierung anwendet. Eine Lösung wäre die Beschränkung der lokalen Methoden auf die Subgoal-Bäume des Tableauxs. Trotzdem kann es zu Unvollständigkeit kommen.

Als Lösung kann man die zweite Regel ändern:

2. In jedem alternativen Lösungsprozess des Subgoales N unterhalb des Suchknotens N: gibt es eine Substitution $r = r_1 \ldots r_n$ so dass eine der Fehlersubstitutionen die gespeichert sind im Knoten N allgemeiner ist als r, dann beginnt die Beweisprozedur sofort mit Backtracking.

Mit anderen Worten: Die Failure Substitution bei einem Subgoal muss deaktiviert werden, wenn das Subgoal gelöst wurde. Diese Einschränkung der Nutzung der Failure Substitution hat Vollständigkeit.

5 Schlussfolgerung

Tableauxverfeinerungen schränken den Suchraum ein, sie haben aber auch Nachteile [2]:

1. Falls nicht beweiskonfluent erfordert die Implementierung Backtracking, keine Gegenbeispiele
2. Kombination von für sich genommen vollständigen Verfeinerungen kann unvollständig sein
3. Es gibt Formelklassen Φ_n, deren kürzeste Beweise mit verfeinerten Tableauxs exponentiell länger sind als ihre kürzesten Beweise mit Klauseltableaus.

Während die vorgestellt Methode des Caching Pruningtechniken verwendet, reduziert Lovelands Substitution den aktuellen Subgoal Baum. Das Konzept basiert auf Beweistransformation, welche es erlaubt, bestimmte Subgoals zu ignorieren, wenn das Set von Literalen bei aktuellem Subgoal von einer Eingabeklausel subsumiert wird.

Literatur

[1] Reinhold Letz, Gernot Stenz, *Model elimination and Connection Tableau Procedures*, 2001.

[2] Dr. Bernhard Beckert, *Vorlesungsskript Analytisches Beweisen*, 2002.

[3] Owen L. Astrachan, Mark E. Stickel, *Caching and Lemmaizing in Model Elimination Theorem Provers*,

[4] O.L. Astrachan and D.W. Loveland, *The Use of Lemmas in the Model Elimination Procedure,*

[5] Professor Jörg Siekmann, *Inferenzmethoden der Künstlichen Intelligenz*,

[6] Martin Giese, *Proof Search Without Backtracking for Free Variable Tableaux*, 2002