

Fachhochschule Wedel

Seminararbeit

Zum Thema

Automatisches Beweisen

Erstellt von: Dennis Ziska
Tinsdaler Weg 127
22880 Wedel
04103 - 900210

betreut von: Prof. Dr. Iwanowski
Fachhochschule Wedel
Feldstraße 143
22880 Wedel
04103 - 80 48 – 63

Übersicht:

1. allgemeine Problemstellung	3
2. Grundlagen	5
2.1 Prädikatenlogik	5
2.2 Substitution und Unifikation	5
2.3 Normalformen	6
2.4 Skolemisierung	7
2.5 Klauselsprache	8
3. Methoden	
3.1 Tableaumethode	9
3.2 Resolutionskalkül	15
Abbildungs- und Literaturverzeichnis	17

1. Allgemeine Problemstellung:

Die Anfänge im Bereich automatisches Beweisen lassen sich in die Mitte der 50er Jahre zurückverfolgen. Damals gab es erste Versuche, mathematische Resultate durch Computerprogramme zu berechnen. Daraus ergibt sich die Frage, was automatisches Beweisen eigentlich bedeutet oder leisten soll. Eine einfache Definition ist die Aussage, dass automatisches Beweisen bedeutet, mathematische Beweise mittels Computerprogrammen zu führen.¹ Etwas ausführlicher heißt das, „mathematische Wahrheiten formal zu modellieren und“ mit einer endlichen Anzahl von „Axiomen und Regeln formal zu beweisen“.² Im Einzelnen werden dazu zwei Aktionen unternommen: Zuerst wird ein Problem modelliert und anschließend zum Beweis abgeleitet.

Bei der Modellierung wird als erstes ein Problem aus der natürlichen Sprache über Formalisierung nach einer festgelegten Syntax zu Zielformeln transformiert. Anschließend können den formalen Formeln Wahrheitswerte zugeordnet werden, wodurch semantische Korrektheit hergestellt wird. Syntax und Semantik zusammen bilden eine Logik. Für das automatische Beweisen wird für gewöhnlich die Prädikatenlogik 1. Stufe verwendet. Allerdings können auch schwächere oder stärkere Logiken verwendet werden, die für die Umsetzung aber gegebenenfalls nicht mächtig genug sind oder zu kompliziert werden können. Die Prädikatenlogik wird bei den Grundlagen in Kapitel 2.1 ganz kurz angesprochen werden.

Auf die Modellierung wird in diesem Beitrag allerdings nur ansatzweise eingegangen. Hier wird das Hauptaugenmerk auf dem Bereich der Deduktion liegen.

Deduktion nennt man das ableiten der syntaktischen Formeln auf beweisbare Formeln. Das geschieht unter Berücksichtigung der Korrektheit und Vollständigkeit. Vollständigkeit bedeutet, dass jede gültige Formel abgeleitet werden kann, Korrektheit besagt, dass jede abgeleitete Formel wahr ist. Die Menge aller Ableitungsregeln heißt Kalkül. Zur Veranschaulichung hilft die Grafik Abbildung 1 auf der folgenden Seite.

Das automatische Beweisen beschäftigt sich nun hauptsächlich damit, diese Ableitungsregeln automatisch zu finden und zu nutzen.

¹ Vergl. Handbuch der KI, S.199

² Zitat: Vorlesungsscript Hähnle/Beckert, <http://i12www.ira.uka.de/~beckert/Lehre/Automatisches-Beweisen/>

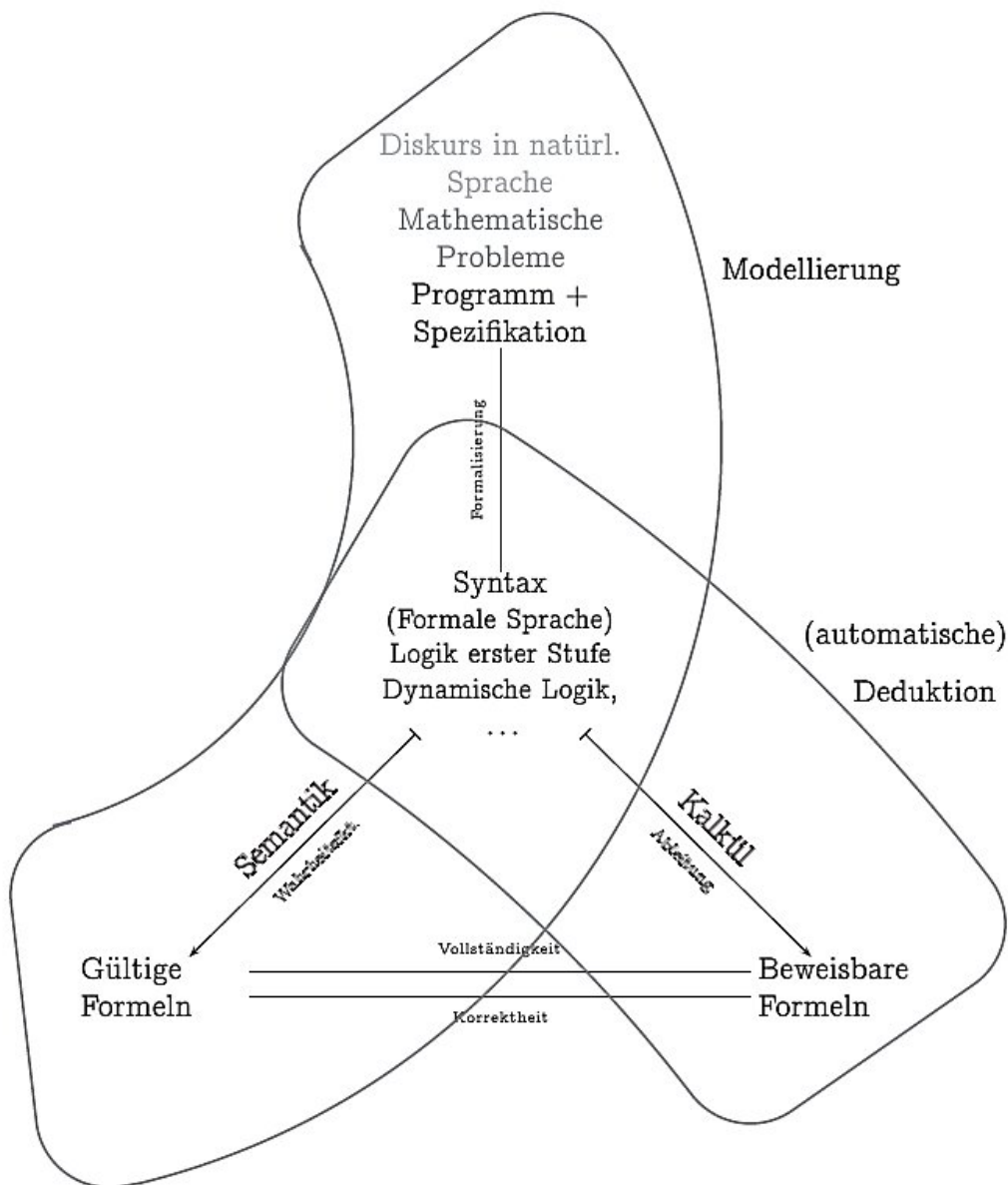


Abbildung 1, Zusammenhang der Modellierung und Deduktion

2. Grundlagen:

2.1 Prädikatenlogik:

Da eine Grundkenntnis der Prädikatenlogik beim Leser vorausgesetzt wird, soll hier nur noch einmal auffrischend das Prinzip dargestellt werden.

Als Operatoren gibt es \neg ..Negation, \wedge ..Konjunktion, \vee ..Disjunktion, \leftrightarrow ..Äquivalenz, \rightarrow ..Implikation und die Quantoren \forall ..Allquantor, \exists ..Existenzquantor. Dazu gibt es Klammern (,) zur Priorisierung von Auswertungen.

Vorhandene Elemente sind Prädikate (variablenabhängige Aussagen), Variablen (Literale) und Funktionen (Zuordnungen). Nullstellige Funktionen werden als Konstanten gewertet.

Eine prädikatenlogische Formel besteht aus Verknüpfungen der Elemente durch die Operatoren.

In einer Formel können freie und gebundene Variablen auftreten.

Bsp.: $\forall x((f(x)\wedge y) \vee (x\wedge z))$ alle Vorkommen der Variable x sind durch den Allquantor gebunden, die Variablen y und z sind frei.

Weitere Verwendungen können bei Anwendungen in Beispielen der folgenden Kapitel eingesehen werden.

2.2 Substitution und Unifikation:

Da die Substitution, vor allem in Form von Unifikation, bei beiden in dieser Arbeit vorgestellten Methoden des Beweises angewendet wird, soll hier die grundlegende Technik vorgestellt werden. Der Begriff „Substitution“ kommt aus dem lateinischen und bedeutet „Ersetzung“. Genau das wird in den prädikatenlogischen Formeln vollzogen. Dabei wird eine endliche Menge von Variablen in einem Ausdruck durch eine festgelegte Substitutionsvorschrift verändert. Bei Anwendung auf eine Konstante, verändert sich diese nicht, bei Anwendung auf komplexe Terme wird sie auf jedes Element dieses Terms angewendet.

Zum Beispiel: ein Ausdruck $t = g(xh(yz))$ wird über die Substitution $\sigma=\{x/a, y/f(ax)\}$ abgebildet. In diesem Beispiel wird jedes Vorkommen von x im Ausdruck t durch die Variable a ersetzt, entsprechend für y mit f(ax). Also ist das Ergebnis $\sigma(t) = \sigma(g(xh(yz))) = g(\sigma(x) \sigma(h(yz))) = g(ah(\sigma(y) \sigma(z))) = g(ah(f(ax)z))$. Dabei ist zu be-

achten, dass die Ersetzungen gleichzeitig getätigt werden, so dass es nicht zu zyklischen Abhängigkeiten kommen kann.

Die Unifikation ist eine spezielle Substitution. Das Prinzip darin besteht, eine Substitution zu finden, die auf eine Menge von Termen angewendet, jeweils zum gleichen Ergebnis führt. Zum Beispiel ist $\sigma = \{x/c, y/a, z/c\}$ ein Unifikator für die Terme $D = \{f(x,a), f(z,y)\}$. Das Ergebnis der Substitution ist $D' = \{f(c,a), f(c,a)\}$. Wie man sehen kann, gibt es eine große Menge von Unifikatoren. Nun ist das Ziel, den „allgemeinsten Unifikator“ (most general unifier - mgu) zu finden. Der mgu ist der Unifikator, für den gilt: $\sigma = \sigma^\circ \mu$, wobei σ eine beliebige Substitution ist und μ der mgu. $^\circ$ steht für die Funktionskomposition. Entsprechend angewendet auf einen Term t gilt: $(\sigma^\circ \mu)(t) = \sigma(\mu(t))$. Der mgu führt keine neuen Variablen ein, sondern benutzt die, die bereits in den Termen vorkommen. Bezüglich der Funktionskomposition ist der mgu idempotent, d.h. auf sich selbst angewendet führt die Unifikation zu keiner Änderung des Ergebnis. Jede Problemstellung hat einen allgemeinsten Unifikator, oder sie kann nicht unifiziert werden.

2.3 Normalformen:

Die Ausgangsform für Skolemisierung und damit grundlegend wichtig ist die pränex Normalform (PNF). Eine Formel F ist in PNF, falls sie in folgender Form vorliegt: $F = \Delta_1 x_1 (\Delta_2 x_2 (\dots \Delta_n x_n B))$. Δ steht hier stellvertretend für einen der Quantoren \forall oder \exists . B definiert die Matrix von F und ist selbst eine quantorenfreie Formel. $\Delta_1 x_1 (\Delta_2 x_2 (\dots \Delta_n x_n$ heißt der Präfix von F .

Die PNF kann über die Negations- Normalform (NNF) errechnet werden. Eine Formel liegt in NNF vor, wenn Negation nur noch vor Atomen steht, und alle Implikationen und Äquivalenzen ersetzt wurden. Um die PNF zu erreichen, werden dann schrittweise alle Quantoren über festgelegte Umformungen „nach Außen“ gezogen. Begonnen wird mit dem ersten Vorkommen eines Quantors in der Formel.

Die Matrix liegt für gewöhnlich in konjunktiver Normalform (KNF) vor.

Beispiel: $F = [\exists x \forall y \forall z \exists w [P(xz) \rightarrow Q(yzw)]]$

[Präfix [Matrix]]

2.4 Skolemisierung:

Ziel der Skolemisierung ist es, Existenzquantoren in Formeln zu eliminieren. Voraussetzung dafür ist, dass die zu verändernden Formeln in PNF vorliegen. Dann werden nacheinander alle durch Existenzquantoren gebundenen Variablen durch Skolemterme ersetzt. Begonnen wird mit dem zu erst auftretenden Existenzquantor. Ein Skolemterm ist eine k -stellige Funktion, wobei nullstellige Skolemfunktionen auch Skolemkonstanten genannt werden. Gibt es in einem zu ersetzenden Ausdruck noch andere Abhängigkeiten, so werden die abhängigen Variablen als Parameter in die Skolemfunktion übergeben. Die wird anschaulicher an einem Beispiel: die Formel F liegt in PNF vor. $F = [\exists x \forall y \forall z \exists w [P(xz) \rightarrow Q(yzw)]]$. Im ersten Schritt der Skolemisierung wird $\exists x$ eliminiert. Dazu werden alle gebundenen Vorkommen der Variable x ersetzt, so dass sich F' ergibt. $F' = [\forall y \forall z \exists w [P(a z) \rightarrow Q(yzw)]]$. a ist eine Skolemkonstante. Im nächsten Schritt wird $\exists w$ eliminiert. Da die gebundene Variable w auch von y und z abhängig ist, werden diese in die Skolemfunktion übernommen. Daraus ergibt sich $F'' = [\forall y \forall z [P(az) \rightarrow Q(yz f(yz))]]$. Da jetzt keine Existenzquantoren mehr vorkommen ist die Skolemisierung abgeschlossen und $F^* = F''$ ist das Ergebnis. Es ist zu beachten, dass die Skolemisierung nicht äquivalenzerhaltend ist. Das ist für das automatische Beweisen auch nicht nötig, da die Erfüllbarkeit, bzw. die Unerfüllbarkeit sehr wohl erhalten bleibt.

Zwei Nachteile, die bei der Skolemisierung auftreten können, sollen hier noch anhand von Beispielen erläutert werden:

1. Die Formel $F = \forall x ((\exists y p(y)) \vee q(x))$ wird durch Skolemisierung zu $F^* = \forall x (p(f(x)) \vee q(x))$. Die Skolemvariable x hätte aber nicht eingeführt werden müssen, da x gar nicht im Wirkungsbereich von $\exists y$ vorkam.
2. Aus $F = (\exists x p(x)) \vee (\exists y p(y))$ wird $F^* = p(c) \vee p(d)$. Damit werden unnötigerweise zwei verschiedene Skolemkonstanten eingeführt. Gleiche Existenzquantifizierte Unterformeln können das gleiche Skolemsymbol erhalten: $F^* = p(c) \vee p(c)$

Diese Probleme lassen sich vermeiden, indem Änderungen in der Skolemisierung vorgenommen werden: Loslösung von der PNF, feste Zuordnung von Skolemsymbolen zu Unterformeln und damit wiederholbare Nutzung, definitorische (strukturertehende) KNF-Transformationen. Auf diese möglichen Verbesserungen wird im Rahmen dieser Arbeit nicht näher eingegangen werden. (Siehe dazu Vorlesungsskriptum Beckert/Hähnle, <http://i12www.ira.uka.de/~beckert/Lehre/Automatisches-Beweisen/>)

2.5 Klauselsprache:

Die Klauselsprache ist eine Teilsprache der Prädikatenlogik. Um Klauseln zu erzeugen, müssen die Formeln in skolemisierter Form vorliegen. Dann werden allquantifizierte Disjunktionen durch Mengen von Atomen bzw. negierten Atomen dargestellt, Konjunktionen werden als Mengen dieser Mengen abgebildet. Eine Klausel besteht also aus einer Mengendarstellung von Ausdrücken in KNF. Das bedeutet, dass eine Klausel selbst auch eine Menge ist. Beispielklauseln: $C1=\{P\}$, $C2=\{\neg P,Q\}$, $C3=\{P(f(x)),\neg Q\}$, $C4=\{ \{p(x),\neg q(y)\} , \{\neg p(x)\} \}$, $C5=\square$.

Das Symbol \square steht für die leere Klausel, also die Klausel, die kein Literal enthält.

Zur Veranschaulichung sei hier noch einmal beispielhaft die Klausel C4 vor der Umformung in Klauseln dargestellt: $F^* = (\forall x \forall y (p(x) \vee \neg q(y))) \wedge (\forall z \neg q(z))$

3. Methoden:

3.1 Tableaumethode:

Die Tableaumethode benutzt einen Widerspruchsbeweis über Fallunterscheidung. Dafür wird ein Satz Formeln als Beweisgrundlage genommen. Die zu beweisende Formel wird als Annahme ins Gegenteil verkehrt und diesem Satz hinzugefügt. Anschließend wird durch Fallunterscheidung anhand der Formeln und der Annahme versucht, Widersprüche in den Aussagen zu finden. Dazu wird jede Formel genau einmal untersucht. Die Implementierung und die Darstellung eines Tableaus können als endlich verzweigte Baumstruktur erfolgen. Der Baum besitzt M Knoten, wobei M die Anzahl der Elemente der Formelmenge ist. Ein Blatt steht dabei für einen gefundenen Widerspruch, und der entsprechende Zweig heißt damit geschlossen. Sind alle Zweige geschlossen, dann gilt der Baum als und geschlossen und damit auch das Tableau.

Zur Verdeutlichung hier ein Beispiel aus der mathematischen Mengenlehre.¹ Die ersten vier Formeln aus Abbildung 2 werden als Grundlage verwendet. Die fünfte Aussage lässt sich implizit aus den anderen herleiten.

$$\begin{array}{l} (1) \quad S \cap Q = \emptyset \\ (2) \quad P \subseteq Q \cup R \\ (3) \quad P = \emptyset \rightsquigarrow Q \neq \emptyset \\ (4) \quad Q \cup R \subseteq S \\ \hline (5) \quad P \cap R \neq \emptyset \end{array}$$

Nun gilt es, dieses zu beweisen. Hierfür wird

Abbildung 2, Beispiel Aussagen

zunächst die zu beweisende Aussage ins Gegenteil überführt. In diesem Beispiel wird also aus Formel (5) die Annahme $P \cap R = \emptyset$ gezogen. Nun gilt es zu beweisen, dass diese Annahme in Verbindung mit jeder anderen Aussage einen Widerspruch ergibt. Dazu wird eine beliebige Aussage hergenommen und eine Fallunterscheidung getroffen. Hier wird mit Formel (3) begonnen. Diese besagt, dass entweder $P \neq \emptyset$ oder $Q \neq \emptyset$ gelten muss. Das ergibt die erste Fallunterscheidung. Ein beliebiges Element c muss also der Menge P angehören, falls $P \neq \emptyset$; analog dazu ist ein Element $d \in Q$, falls $Q \neq \emptyset$. In diesem Beispiel wird nur der erste Fall ausführlich dargestellt. Es gilt mittlerweile der Fall $c \in P$. Jetzt wird hier die Annahme herangezogen, die besagt, dass ein Element nicht in den Mengen P und R gleichzeitig vorkommen kann, da diese keine Schnittmenge besitzen. Also entstehen hier die beiden Fälle $c \notin P$ und $c \notin R$. Der erste Fall führt direkt zu einem Widerspruch, denn es ist nicht möglich, dass das Element c

¹ entnommen aus dem Vorlesungsskriptum von Beckert/Hähnle,
<http://i12www.ira.uka.de/~beckert/Lehre/Automatisches-Beweisen/>

in der Menge P auftreten und gleichzeitig nicht auftreten kann. Damit braucht nur noch der zweite Zweig verfolgt zu werden. Als nächstes wird Aussage (2) behandelt. Diese besagt, dass ein in P auftretendes Element auch in mindestens einer der beiden Mengen Q und R auftreten muss. Es ist bereits belegt, dass $c \in P$ und $c \notin R$. Damit bleibt als einziger widerspruchsfreier Fall $c \in Q$. Wird nun Formel (1) ausgewertet, ergibt sich, dass ein Element c nur in einer der beiden Mengen S und Q auftreten kann. Für den Fall, dass $c \notin Q$ ergibt sich ein Widerspruch, also bleibt der Fall $c \in S$. Zuletzt bleibt Formel (4), aus der abgeleitet werden kann, dass ein Element, welches nicht in S auftritt auch nicht in $Q \cup R$ auftritt. Das führt jeweils zu Widersprüchen mit bisher getroffenen Feststellungen. Da keine Formeln zu Auswertung übrig sind, ist der Beweis für diese Seite abgeschlossen. Der zweite Fall bei Formel (3) wird genauso behandelt. Als Ergebnis ist die Annahme ist widerlegt, und damit ist die Aussage (5) bewiesen. Zur Veranschaulichung ist der entstandene Teilbaum in Abbildung 3 dargestellt.

Reihenfolge der
benutzen Formeln:

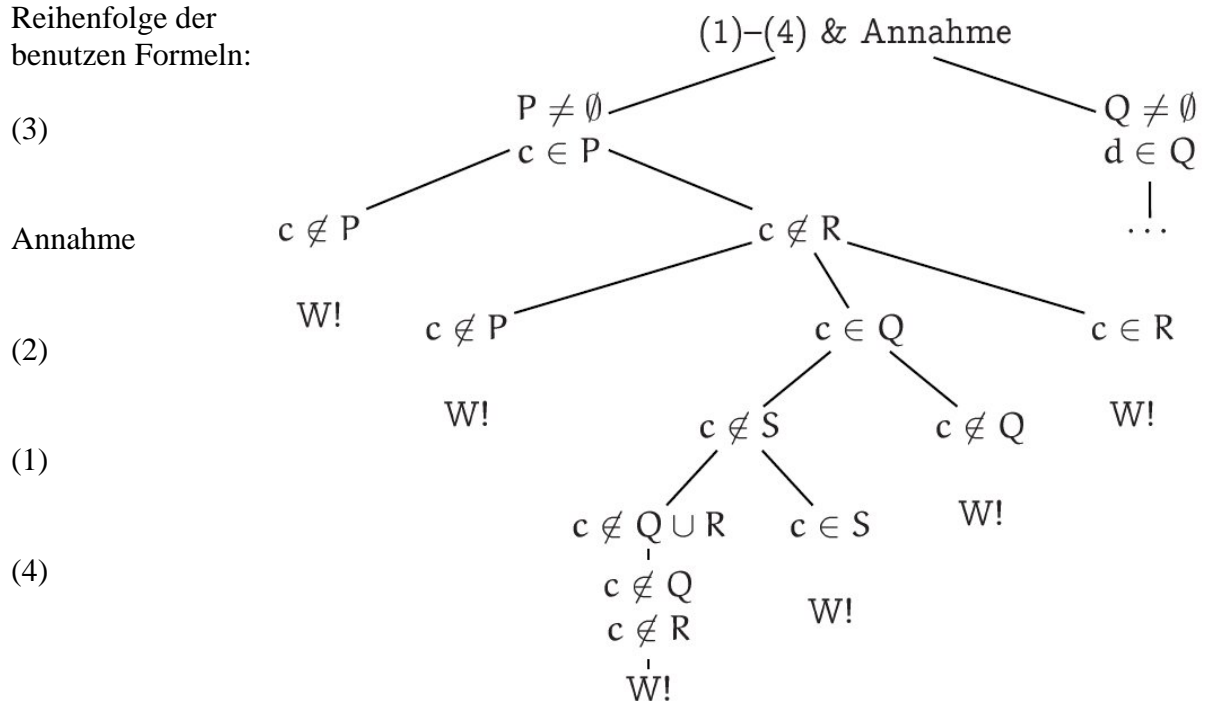


Abbildung 3, Baumstruktur des Beispielbeweises

Die oben getroffenen Aussagen in mathematischer Syntax können auch in Prädikatenlogik 1.Stufe aufgeschrieben werden. Das automatische Beweisen anhand der Tableaumethode arbeitet genau mit dieser Form. Die Umsetzung auf Tableaus heißt dann Grundtableau.

In einem Grundtableau werden gebundene Variablen durch Grundterme ersetzt. Das Vorgehen entspricht der Skolemisierung (siehe Kapitel 2.4). Als Folge davon kommen auf Grundtableaus nur geschlossene Formeln vor. Größtes Problem dabei ist allerdings, dass die richtige Instanz, also die Ersetzung, die zum Abschluss des

Tableaus führt, zum Zeitpunkt der Anwendung der Ersetzungsregeln nicht bekannt ist. Resultierend daraus kann es extrem große und redundante Grundtableaus geben. In Abbildung 4 sind die Aussagen aus dem obigen Beispiel in Prädikatenlogik formalisiert und in dem entsprechenden Grundtableau dargestellt. Die Reihenfolge der untersuchten Formeln entspricht der oben genannten. Wie aus der Abbildung ersichtlich, werden die Quantoren und die durch sie gebundenen Variablen ersetzt. Die Bögen zeigen die Widersprüche bzw. die Abschlusspaare an. Der grau hinterlegte Zweig ist noch nicht abgeschlossen, dieses kann aber analog zum vorhandenen Beweis vollzogen werden.

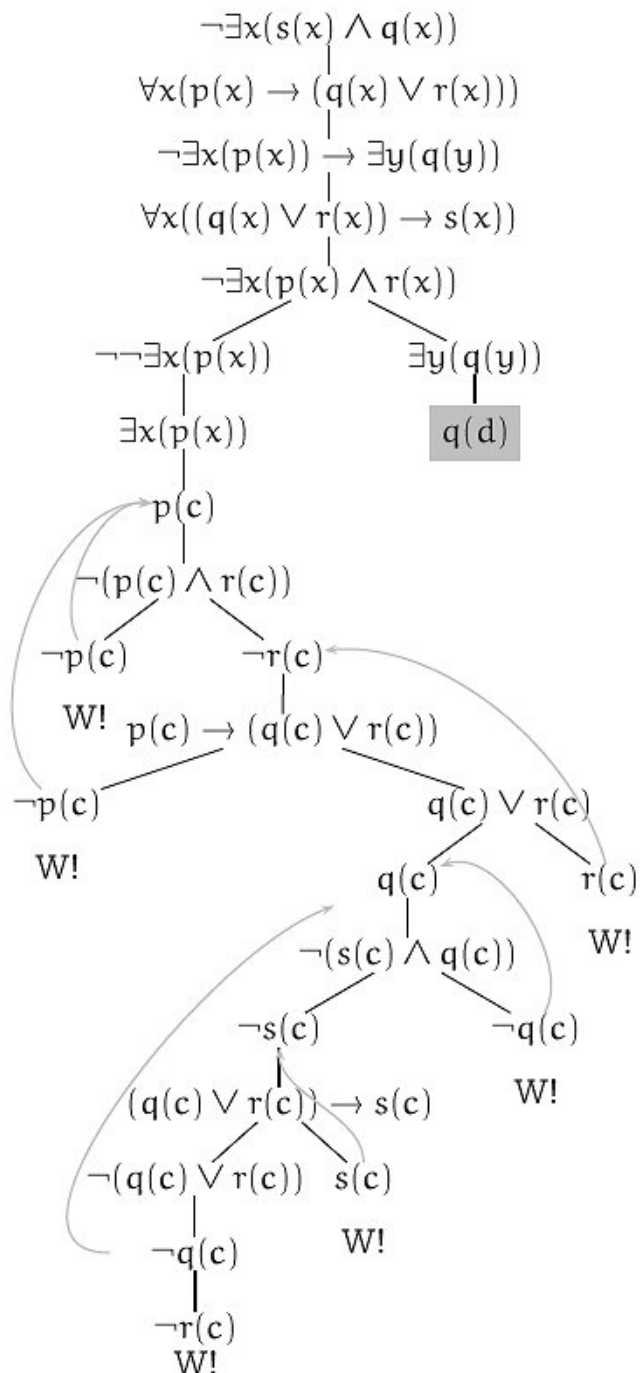


Abbildung 4, partieller formalisierter Grundtableaubeweis

Um den Nachteil der Redundanz zu beseitigen wurden Grundtableaus zu Freie-Variablen Tableaus (FV-Tableaus) erweitert. Während Grundtableaus mittels unspezifizierten Grundtermen ersetzen, werden bei FV-Tableaus freie Variablen eingeführt, die bei Bedarf über Unifikation (siehe Kapitel 2.2) instanziiert werden können. Zur Veranschaulichung soll Abbildung 5 dienen. Diese zeigt ein FV-Tableau für das oben begonnene Beispiel. Die durchgezogenen Bögen zeigen Widersprüche bzw. Abschlusspaare an, die gestrichelten Bögen markieren Elternformeln. Dass FV-Tableaus auch vollständig sind, kann aus der Vollständigkeit der zugrunde liegenden Grundtableaus abgeleitet werden. Ein Beweis soll hier nicht weiter geführt werden, kann aber zum Beispiel bei Melvin C. Fitting „First-Order Logic and Automated Theorem Proving“ Springer-Verlag, NY, 2nd ed. 1996 nachgelesen werden. Die Vollständigkeit der Tableauregeln garantiert aber nur, dass ein geschlossenes Tableau tatsächlich existiert. Ein automatischer Beweiser muss dieses allerdings auch finden können. Ein ähnliches Problem ergibt sich dadurch, dass für erfüllbare Formelmengen kein geschlossenes FV-

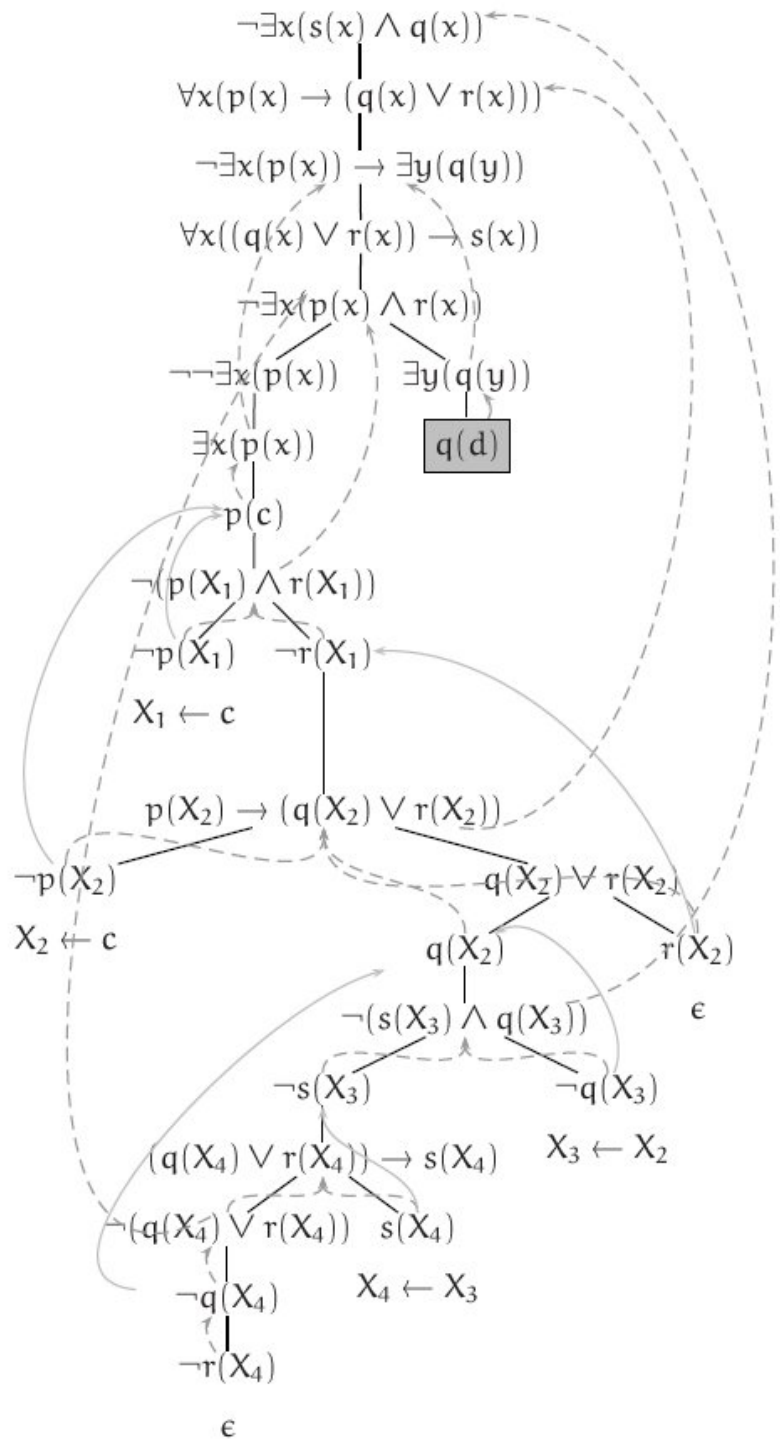


Abbildung 5, FV-Tableau

Tableau existieren kann. Daher müssen bei der Umsetzung folgende Fragen zum Indeterminismus betrachtet werden:

1. Welcher Zweig soll als nächstes zur Betrachtung ausgewählt werden?
2. Soll der gewählte Zweig abgeschlossen oder erweitert werden?
3. Falls abgeschlossen wird: mit welcher benutzten Formel abschließen?
4. Falls erweitert wird: über welche Elternformel?

Hier eine mögliche Umsetzung dafür: Es wird immer der erste Zweig (im Baum ganz links) betrachtet; Zweige in denen unifizierbare Literale vorkommen immer abschließen, sonst expandieren; erst eine Ebene komplett untersuchen.

Diese mögliche Umsetzung kann, genau wie jede andere, bei speziellen Fällen zu Problemen führen. Diese werden Bevorzugungsprobleme (Fairness) genannt und wie folgt unterteilt:

1. Bevorzugung eines bestimmten Literals:
Es wird immer das gleiche Literal zu einem Abschluss benutzt, als Folge kann es passieren, dass die übrigen Äste nicht abgeschlossen werden können.
2. Bevorzugung einer bestimmten Formel:
Es wird immer die gleiche Formel zum Expandieren genutzt, als Folge können Unvollständigkeiten auftreten.
3. Bevorzugung eines Modus (Abschluss/Erweiterung):
Es kann zu Unvollständigkeiten durch Vernachlässigen des anderen Modus kommen.

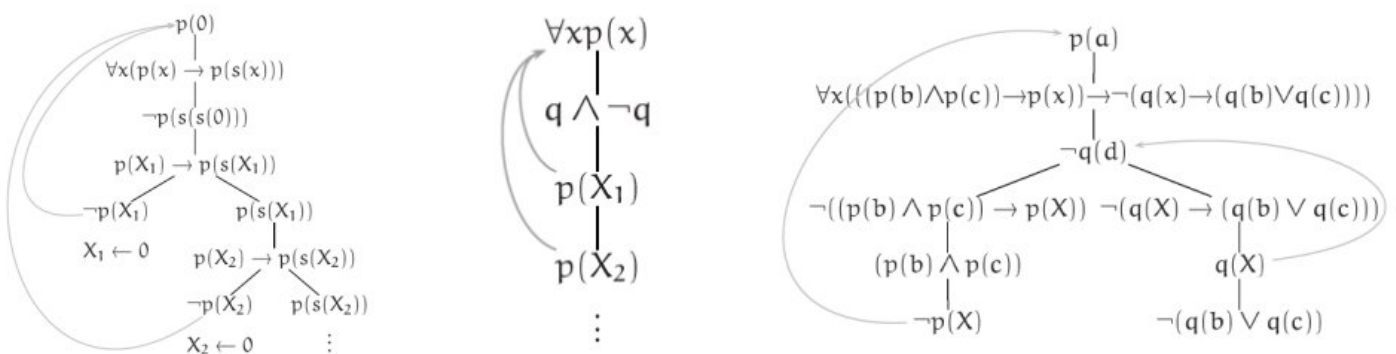


Abbildung 6, Beispiele für Bevorzugungsprobleme: 1.-3. v.l.n.r

Lösung für diese Probleme kann es durch faire Implementierungen geben. Dabei wird stets das ganze Tableau betrachtet, die Bearbeitungsschritte erfolgen also nicht destruktiv. Ein endgültiger Abschluss wird erst getätigt, wenn eine passende Substitution gefunden wurde, d.h. wenn eine Substitution alle Zweige abschließen kann. Suchverfahren, die angewendet werden können, um einen Baum zu traversieren

wurden bereits in einem vorangehenden Beitrag erläutert. Mit der Verwendung von Fairness werden die Kalküle beweiskonfluent, das heißt, dass ein angefangener Beweis für unerfüllbare Formelmengen vervollständigt werden können. Allerdings hat die Implementierung mit Fairness auch Nachteile. So muss zum einen das gesamte Tableau mit allen möglichen Ausprägungen zwischengespeichert werden. Zum anderen müssen bei jedem weiteren Schritt alle Knoten geprüft werden, auch die bisher behandelten, weil sich die gesuchte Substitution auf das gesamte Tableau auswirken soll.

Weiterhin können Optimierungen für die Formelwahl getroffen werden. Eine gängige Methode ist dabei das Setzen von so genannten „Links“. Ein Link besteht, wenn es an einem Zweig ein Paar von Formeln gibt, die eine unterschiedliche Polarität aufweisen (durch Vorkommen von Negationen) und über eine Substitution zu einem komplementären Paar, also einem Abschlusspaar, gemacht werden können. Falls kein Link besteht, sollte eine andere Formel gewählt werden, da die Formel ohne Link nicht zu einem Abschluss gebracht werden kann.

3.2 Resolution:

Die Resolutionsmethode benötigt Eingaben in Form von Klauseln (siehe Kapitel 2.5). Grundidee hinter der Resolutionsmethode ist es, mittels Anwendung von Schlussregeln auf diese Klauseln einen Widerspruch herzuleiten. Die Schlussregeln teilen sich in Resolution und Faktorisierung und sollen hier als Kernstück des Resolutionsbeweises eingehender betrachtet werden.

Die Resolutionsregel ist von der Logik gesehen nichts anderes, als die Schnittregel bei Mengen. Allerdings gibt es den gravierenden Unterschied, dass die Schnittmenge für gewöhnlich erst gebildet werden muss, während sie bei der Resolution immer bekannt ist. Von daher kann die Darstellung der Resolution als azyklisch gerichteter Graph (DAG) erfolgen. Die bekannte Schnittmenge wird aus zwei Elternklauseln resolviert, d.h. ein Resolvent wird aus den Klauseln über günstige Substitution gewonnen. Eine geeignete Substitution erzeugt dabei mindestens ein komplementäres Mengenpaar in den Elternklauseln, d.h. die Clauselemente sind bis auf ihr Vorzeichen identisch. Der Resolvent hebt sich somit auf, und als neue Klausel ergibt sich die Vereinigung der beiden ursprünglichen Klauseln mit den getätigten Substitutionen aber ohne den Resolventen. Der Resolvent wird durch folgende Formel definiert: $\text{Res}(C,L,D,K,\sigma)$, wobei C und D die Elternklauseln benennen, L ein Element der Klausel C und K ein Element der Klausel D ist, und σ den mgu (siehe Kapitel 2.2) für die beiden Klauseln bildet. Für die Notation werden L und K oftmals auch vereinfachend durch ihre Position in der Klausel, also ihren Index ersetzt: $\text{Res}(C,i,D,j,\sigma)$. Die neue Klausel ist definiert nach: $C_{\text{neu}} = \sigma(C-L)\sigma \cup (D-K)$. Anschaulicher wird das Vorgehen anhand eines Beispiels:

Seien $C1 = \{P(xb), P(by), \neg Q(xy)\}$ und $C2 = \{\neg P(az), Q(zz)\}$ die Elternklauseln, die resolviert werden sollen. Dann ergibt sich als eine Möglichkeit der Resolution: $R1 = \{P(by), \neg Q(ay), Q(bb)\} = \text{Res}(C1, P(xb), C2, \neg P(az), \{x/a, z/b\})$. $P(xb)$ aus $C1$ wird über die Substitution zu $P(ab)$, $\neg P(az)$ aus $C2$ wird zu $\neg P(ab)$. Dieses komplementäre Paar hebt sich auf, übrig bleiben nach der Ersetzung $P(by), \neg Q(ay)$ aus $C1$ und $Q(bb)$ aus $C2$. Vereinfacht mit Index geschrieben: $\text{Res}(C1, 1, C2, 1, \{x/a, z/b\})$.

Die Faktorisierungsregel betrachtet eine einzelne Klausel. Dabei wird ebenfalls eine günstige Substitution auf die Elternklausel angewendet. Mit einer geeigneten Substitution wird versucht, einen Faktor innerhalb dieser Klausel zu finden. Ein Faktor bestimmt ein äquivalentes Mengenpaar, d.h. die Clauselemente sind komplett iden-

tisch. Da die Klausелеlemente disjunktiv verknüpft sind können die durch die Substitution entstandenen äquivalenten Elemente zu einem zusammengefasst werden. Die neue Klausel ergibt sich aus der alten Klausel, mit allen durch die Substitution verlangten Änderungen, inklusive des faktorisierten Elements. Der Faktor wird über folgende Formel definiert: $Fak(C,\sigma)$, wobei C die zu faktorisierende Klausel benennt und σ den mgu darstellt. Zur Veranschaulichung wird hier dasselbe Beispiel wie für die Resolution herangezogen: $C1 = \{P(xb),P(by),\neg Q(xy)\}$ und $C2 = \{\neg P(az),Q(zz)\}$ sind zwei Elternklauseln. Eine Möglichkeit der Faktorisierung besteht nun in: $F1=\{P(bb),\neg Q(bb)\} = Fak(C1,\{x/b,y/b\})$. Durch die Anwendung der Substitution $\{x/b,y/b\}$ auf $C1$ entsteht zunächst eine Menge $M=\{P(bb),P(bb),\neg Q(bb)\}$. Die ersten beiden Elemente werden nun zusammengefasst.

Es gibt die Möglichkeit, diese beiden Schlussregeln, Resolution und Faktorisierung, zusammenzufassen. Dadurch wären weniger Arbeitsschritte zur Lösungsfindung nötig. Im Gegenzug dazu würde allerdings die Übersichtlichkeit leiden und sich somit die Fehlerquote erhöhen.

Grundsätzlich wird bei der Resolutionsmethode voraus gesetzt, dass die einzelnen Klauseln variablenfremd sind. Das bedeutet, dass unterschiedliche Klauseln keine gemeinsamen Variablensymbole benutzen und damit auch nicht über die Variablen voneinander abhängen. Diese Voraussetzung kann leicht durch Variablenumbenennung erreicht werden. Das geschieht, indem in einer Klausel die zweifelhaften Variablensymbole durch bisher nicht verwendete ersetzt werden. Dieses Vorgehen ist auch korrekt, da sich durch die Umbenennung nicht die Aussage ändert. Trotzdem ist sie nötig, da es beim automatischen Beweisen sonst zu Unifikationsproblemen kommen kann. So können z.B. Herleitungen nicht geschlossen werden, weil kein Unifikator gefunden werden kann, oder es kann zu einem falschen mgu führen, der je nach Reihenfolge der Regelbetrachtung variiert. Durch Variablenumbenennung wird also garantiert, dass ein bestimmter mgu gewählt wird.

Abschließend hier noch ein Beispiel für eine komplette Resolution:

(C1) $\{\neg P(x1),P(f(x1))\}$	(C5) $\{\neg Q(y2),P(a)\}$	$Fak(C3,\{z1/y1\})$
(C2) $\{P(a),Q(b)\}$	(C6) $\{P(a)\}$	$Res(C2,2,C5,1\{y2/b\})$
(C3) $\{\neg Q(y1), \neg Q(z1),P(a)\}$	(C7) $\{P(f(a))\}$	$Res(C1,1,C6,1\{x1/a\})$
(C4) $\{\neg P(f(a))\}$	(C8) \square	$Res(C4,1,C7,1,\varepsilon)$

Abbildungsverzeichnis:

Abbildung 1: Zusammenhang von Modellierung und Deduktion	4
Abbildung 2: Beispiel Aussagen über Mengen	9
Abbildung 3: Baumstruktur eines Beispielbeweises	10
Abbildung 4: Beispielbeweis als Grundtableau	11
Abbildung 5: Beispielbeweis als FV-Tableau	12
Abbildung 6: Beispiele für Bevorzugungsprobleme	13

Alle Abbildungen wurden entnommen aus dem Vorlesungsscript von Beckert und Hähnle von der Seite: <http://i12www.ira.uka.de/~beckert/Lehre/Automatisches-Beweisen/>

Quellenverzeichnis:

- Görz/Rollinger/Schneeberger: Handbuch der Künstlichen Intelligenz, 3. Auflage, Oldenbourg Wissenschaftsverlag, 2000
- Beckert/Hähnle: Vorlesungsskriptum „Automatisches Beweisen“, <http://i12www.ira.uka.de/~beckert/Lehre/Automatisches-Beweisen/>, 3. Auflage, Karlsruhe, 2001

im Internet:

- <http://wikipedia.org>
- <http://www.computerbase.de/lexikon/>