GERT SMOLKA

EINIGE ERGEBNISSE ZUR VOLLSTÄNDIGKEIT DER BEWEISPROZEDUR VON KOWALSKI

DIPLOMARBEIT BETREUT VON P. DEUSSEN UND J.H. SIEKMANN
FAKULTÄT FÜR INFORMATIK, UNIVERSITÄT KARLSRUHE, JULI 1982

I N H A L T

Binet lächelte mit gesenktem Kopf und geweiteten Nasenflügeln und schien versunken in einen jener vollkommenen Glückszustände, die zweifellos nur anspruchslose Beschäftigungen vermitteln können, die den Geist durch leicht zu überwindende Schwierigkeiten anregen und ihn völlig befriedigen mit einem Resultat, das keine weiteren Wünsche mehr schafft.

Aus Gustave FLAUBERTs
"Madame Bovary"

E I N L E I T U N G

Das automatische Beweisen mathematischer Sätze durch einen Computer wird seit etwa zwanzig Jahren intensiv erforscht und ist heute eine eigene Diziplin auf dem Gebiet der Künstlichen Intelligenz. Zwar sind seit HERBRAND /He29/ Algorithmen bekannt, die zu jedem gültigen Satz einer formalisierbaren Theorie einen Beweis berechnen, aber diese Beweisprozeduren benötigen für interessante Theoreme so ungeheuer viele Rechenschritte, daß sie auch auf zukünftigen Rechenanlagen nicht realisierbar sein werden. Nach den ersten enttäuschenden Erfahrungen mit diesen HERBRAND-Prozeduren wurde die Entwicklung ab 1965 durch den von J.A. ROBINSON entwickelten Resolutionskalkül /Ro65/ dominiert. Dieser Kalkül ist einer der ersten Logikkalküle, die speziell für die Implementierung auf einem Rechner entworfen wurden. In der folgenden Dekade wurden Dutzende von Verfeinerungen für den Resolutionskalkül entwickelt und implementiert. Da keiner dieser Versuche zu einem Beweissystem führte, das die euphorischen Erwartungen der frühen sechziger Jahre erfüllte, setzte sich die Erkenntnis durch, daß ein leistungsfähiges Beweisprogramm neben der Deduktionskomponente, die den Logikkalkül implementiert, über eine Wissenskomponente verfügen muß, die inhaltliches Wissen und spezifische Beweismethoden über das mathematische Gebiet einbringt, aus dem das Theorem stammt. Während für die Deduktionskomponente viele Konzepte und Ergebnisse vorliegen (Resolution, Paramodulation, Matrixkalküle usw.), findet man für die Wissenskomponente eines Automatischen Beweisers jedoch bis heute vor allem Absichtserklärungen.

Die Connection Graph Proof Procedure, die der Deduktionskomponente des Karlsruher Beweissystems /BEHSSW81/ zugrunde liegt, wurde 1975 von Robert KOWALSKI publiziert /Ko75/. Diese Beweisprozedur unterscheidet sich von den anderen Verfeinerungen des Resolutionskalküls besonders dadurch, daß mit dem Fortschreiten des Ableitungsprozesses viele der vorher erzeugten Klauseln wieder gelöscht werden können. Trotz der Versuche von BROWN /Br76/, BIBEL /Bi81a, Bi81b/ und von SIEKMANN und STEPHAN /SS76, SS80/ ist das Vollständigkeitsproblem der Con-

nection Graph Proof Procedure bisher nicht gelöst: Es ist offen, ob für den zugrunde liegenden "KOWALSKI-Kalkül" eine Suchstrategie existiert, die für jede unerfüllbare Klauselmenge mit der Erzeugung der leeren Klausel terminiert. Die vorliegende Arbeit untersucht dieses Problem und liefert die folgenden Lösungsbeiträge:

(1) Für die Klasse der unär widerlegbaren Klauselmengen, die im Automatischen Beweisen von großer Bedeutung ist und die die HORN-Mengen mit einschließt, konnten wir alle offenen Fragen klären. Unter anderem geben wir ein Kriterium für die Vollständigkeit von Strategien bei unär widerlegbaren Klauselmengen an. Dieses Kriterium liefert insbesondere, daß jede unär erschöpfende (d.h., jeder Link, der eine unäre Resolution repräsentiert, wird nach endlich vielen Schritten wieder gelöscht) Strategie für unär widerlegbare Klauselmengen vollständig ist. Dies ist auch dann der Fall, wenn die Tautologie-Eliminationsregel uneingeschränkt und die Subsumtions-Eliminationsregel mit der Einschränkung aus /Bi81c/ angewendet werden. Wir zeigen die obigen Ergebnisse mit einer neuen, transparenten Beweistechnik, die sich voraussichtlich so verallgemeinern läßt, daß mit ihr auch das Vollständigkeitsproblem für allgemeine Klauselmengen gelöst werden kann.

(2) Für die Tautologieelimination geben wir Gegenbeispiele an, die zeigen, daß die bisher für den allgemeinen Fall erwarteten und von uns für den unären Fall nachgewiesenen Eigenschaften schon bei aussagenlogischen Klauselmengen nicht mehr erfüllt sind.

(3) BIBEL zeigt in /Bi81c/, daß der KOWALSKI-Kalkül, wenn man bestimmte Restriktionen für die Tautologie- und die Subsumtionselimination vorsieht, für aussagenlogische Klauselmengen konsistent ist, d.h., daß jeder Graph, der aus dem initialen Graphen einer unerfüllbaren aussagenlogischen Klauselmenge abgeleitet werden kann, wieder unerfüllbar ist. Wir zeigen in dieser Arbeit die eigentlich interessante, stärkere Eigenschaft der Konfluenz, d.h., daß jeder Graph, der aus dem initialen Graphen einer unerfüllbaren aussagenlogischen Klauselmenge abgeleitet werden kann, mit der Resolutionsregel für Klauselgraphen widerlegbar ist.

Nach den Vorbereitungen in Kapitel 1 führen wir in Kapitel 2 den KOWALSKI-Kalkül und das Konzept des Filters ein. Dabei haben wir Wert darauf gelegt, alle Konzepte präzise und formal zu definieren und einen breiten begrifflichen Rahmen für unsere und weitere Untersuchungen zu schaffen. In dieser Arbeit betrachten wir eine neue Variante der Klauselgraph-Resolutionsregel, die i.a.

weniger Links erzeugt als die bisher vorgeschlagen Versionen von KOWALSKI /Ko75/, BRUGNOOHE /Br75/, /Ko79/ und von BIBEL /Bi81a, Bi81c/. Natürlich gelten unsere Ergebnisse auch für die bisherigen Versionen der Resolutionsregel. Neben der Tautologieelimination (die Gegenbeispiele (2) finden sich in Abschnitt 2.7) betrachten wir auch ausführlich die Subsumtionselimination. In Kapitel 3 sammeln wir die Ergebnisse für aussagenlogische Klauselmengen und zeigen den Konfluenzsatz (3). Schließlich betrachten wir in Kapitel 4 den Sonderfall der unären Klauselmengen und entwickeln die Lösung des unären Vollständigkeitsproblems. Die Kapitel 3 und 4 sind voneinander unabhängig, so daß der am Hauptergebnis interessierte Leser nach dem Studium von Kapitel 2 sofort zu Kapitel 4 übergehen kann.

sel. Die leere Klausel ist die leere Menge und wird mit $\square$ bezeichnet. Die Menge aller Klauseln bezeichnen wir mit $\mathfrak{C}$. Metasymbole für Klauseln sind C, D und E. Mit $|C|$ bezeichnen wir die Anzahl von Literalen in der Klausel C. Eine Klausel heißt Tautologie, wenn sie zwei Literale enthält, die zueinander komplementär sind. Eine Klausel heißt unär, wenn sie genau ein Literal enthält. Als Metasymbol für Mengen von Klauseln benutzen wir S.

1.1.9 Ausdrücke. Ein Ausdruck ist ein Term oder ein Literal oder eine Klausel. Als Metasymbol für Ausdrücke benutzen wir A.

1.1.10 Enthaltene Variablen. Sei A ein Ausdruck. Mit Var(A) bezeichnen wir die Menge der Variablensymbole, die in A vorkommen. Ein Ausdruck, in dem kein Variablensymbol vorkommt, heißt variablenfrei (engl: ground expression). Für eine Klauselmenge S sind Var(S) und Variablenfreiheit entsprechend definiert.

1.1.11 Varianten. Die Varianten einer Klausel C sind rekursiv wie folgt definiert: C ist eine Variante von C; wenn D eine Variante von C ist und X ein Variablensymbol, das in D vorkommt, dann ist die Klausel, die man aus D erhält, indem man alle Auftreten von X durch ein Variablensymbol Y, das in D nicht vorkommt, ersetzt, eine Variante von C.

1.1.12 Aussagenlogische Klauselmengen. Die nullstelligen Prädikatensymbole heißen aussagenlogische Atome. Ein Literal L heißt aussagenlogisch, wenn $|L|$ ein aussagenlogisches Atom ist. Eine Klausel C heißt aussagenlogisch, wenn alle Literale in C aussagenlogisch sind. Eine Klauselmenge S heißt aussagenlogisch, wenn alle Klauseln in S aussagenlogisch sind.

1.1.13 Schreibweise. Bei Funktions- und Prädikatensymbolen lassen wir den die Stelligkeit angebenden Hochindex weg und sorgen dafür, daß er sich aus dem Kontext ergibt. Für ein negatives Literal $\neg Pt_1 \ldots t_n$ schreiben wir $\bar{P}t_1 \ldots t_n$. Gelegentlich bringen wir die Stelligkeit von Funktions- oder Prädikatensymbolen dadurch zum Ausdruck, daß wir die Unterterme in Klammern einschließen und durch Kommata trennen, z.B. f(a,b) statt fab Eine Klausel $\{L_1,\ldots,L_n\}$ schreiben wir kurz als Zeichenreihe $L_1 \ldots L_n$. Wenn C eine Klausel und L ein Literal ist, dann ist $C - L := C - \{L\}$

1.2 Unifikationen

1.2.1 Substitutionen. Eine Abbildung σ: TERME $\longrightarrow$ TERME heißt Substitution, wenn gilt:

(a) Für jeden Term $ft_1 \ldots t_n$ ist $\sigma(ft_1 \ldots t_n) = f\sigma(t_1) \ldots \sigma(t_n)$.

(b) Für jede Konstante a ist $\sigma(a) = a$.

(c) Für höchstens endlich viele Variablen X ist $\sigma(X) \neq X$.

Als Metasymbole für Substitutionen verwenden wir σ, τ, θ und auch andere kleine griechische Buchstaben. Statt $\sigma(t)$ schreiben wir kurz σt. Die Identität TERME $\longrightarrow$ TERME ist eine Substitution und heißt die leere Substitution. Als Symbol für die leere Substitution benutzen wir ε. Wenn σ und τ Substitutionen sind, dann ist die Komposition $\tau\sigma$, $\tau\sigma(t) := \tau(\sigma(t))$ wieder eine Substitution. Die Komposition von Substitutionen ist eine assoziative, aber i.a. nicht kommutative Operation. Die Menge der Substitutionen ist mit der Komposition eine Halbgruppe mit dem Einselement ε.

1.2.2 Darstellung von Substitutionen. Die Zeichenreihe X/t heißt Komponente der Substitution σ, wenn X eine Variable ist mit $\sigma X = t$ und $X \neq t$. Eine Substitution hat nur endlich viele Komponenten. Die leere Substitution ist die einzige Substitution, die keine Komponente hat. Eine Substitution ist durch die Menge ihrer Komponenten eindeutig bestimmt, da sie dies durch ihre Restriktion auf VAR ist. Wie üblich identifizieren wir deshalb eine Substitution mit der Menge ihrer Komponenten. Damit sind für Substitutionen die Mengenoperationen Vereinigung, Durchschnitt und Differenz definiert.

1.2.3 Ber(σ) und Var(σ). Sei σ eine Substitution. Die Menge der Variablen X mit $\sigma X \neq X$ heißt Bereich von σ und wird mit Ber(σ) bezeichnet. Die Menge aller Variablen, die in der Komponentendarstellung von σ auftreten, wird mit Var(σ) bezeichnet. Beispielsweise ist für $\sigma = \{x/fy\}$ Ber(σ)$=\{x\}$ und Var(σ)$= \{x,y\}$.

1.2.4 Normale Substitutionen. Eine Substitution σ heißt normal, wenn sie idempotent (d.h. $\sigma\sigma = \sigma$) ist und für zwei verschiedene Variablen X und Y mit $\sigma X = Y$ stets $X < Y$ im Sinne der alphabetischen Ordnung aus 1.1.1 gilt.

1.2.5 Allgemeinste Substitution. Sei M eine Menge von Substitutionen. Eine Substitution $\sigma \in M$ heißt allgemeinste Substitution in M, wenn für jede Substi-

tution $\tau \in M$ eine Substitution θ existiert mit $\tau = \theta\sigma$.

1.2.6 Restriktionen von Substitutionen. Sei σ eine Substitution und V eine Menge von Variablen. Die Restriktion von σ auf V ist definiert durch $\sigma|_V := \{X/t \mid X/t \in \sigma \wedge X \in V\}$ und ist offensichtlich wieder eine Substitution.

1.2.7 Kanonische Fortsetzung von Substitutionen. Sei σ eine Substitution. Für ein Atom $Pt_1 \ldots t_n$ ist $\sigma(Pt_1 \ldots t_n) := P\sigma t_1 \ldots \sigma t_n$. Für ein negatives Literal $\sim L$ ist $\sigma(\sim L) := \sim\sigma(L)$. Für eine Klausel $\{L_1, \ldots, L_n\}$ ist $\sigma(\{L_1, \ldots, L_n\}) := \{\sigma L_1, \ldots, \sigma L_n\}$. Für eine Klauselmenge $\{C_1, \ldots, C_n\}$ ist $\sigma(\{C_1, \ldots, C_n\}) := \{\sigma C_1, \ldots, \sigma C_n\}$.

1.2.8 U-Substitutionen. Sei V eine Menge von Variablen. Eine Substitution θ mit $Ber(\theta) \subset V$ heißt U-Substitution (Umbenennungssubstitution) für V, wenn θ auf V injektiv ist und θ Variablen in Variablen abbildet. Eine Substitution θ heißt U-Substitution für einen Ausdruck A, wenn θ eine U-Substitution für Var(A) ist. Für eine U-Substitution θ heißt $\theta^{-1} := \{Y/X \mid X/Y \in \theta\}$ die inverse Substitution. Offensichtlich ist eine Klausel D genau dann eine Variante einer Klausel C, wenn es eine U-Substitution θ für C mit $D = \theta C$ gibt. Wenn θ eine U-Substitution für C mit $D = \theta C$ ist, dann gilt $C = \theta^{-1} D$.

1.2.9 Lemma. σ, τ, θ seien Substitutionen und A ein Ausdruck. Dann gilt:

(a) $(\tau\sigma)A = \tau(\sigma A)$.

(b) $(\theta\tau)\sigma = \theta(\tau\sigma)$.

(c) Wenn für jeden Term t $\sigma t = \tau t$ ist, dann ist $\sigma = \tau$.

(d) Wenn C eine Klausel mit $|\sigma C| = |C|$ und $L \in C$ ein Literal ist, dann ist $\sigma(C - L) = \sigma C - \sigma L$.

Beweis. Für (a) - (c) siehe /Ro65/. (d) ist evident. [ ]

1.2.10 Unifikation. Sei M eine Menge von Ausdrücken. M heißt unifizierbar, wenn es eine Substitution σ mit $|\sigma M| \leq 1$ gibt. σ heißt dann Unifikator für M.

1.2.11 Unifikationssatz. Zu jeder unifizierbaren Menge von Ausdrücken existiert genau ein normaler allgemeinster Unifikator. Die Unifizierbarkeit ist mit linearem Aufwand entscheidbar. Ebenso läßt sich der normale allgemeinste Unifikator, wenn er existiert, mit linearem Aufwand berechnen. Wenn σ der normale allgemeinste Unifikator für eine Menge M von Ausdrücken ist, dann gilt

$Var(\sigma) \subset Var(M)$.

Beweis. Für die Existenz und Eindeutigkeit des normalen allgemeinsten Unifikators verweisen wir auf /Ro65/ und /He82/. Ein linearer Unifikationsalgorithmus wird von PATERSON und WEGMAN in /PW78/ angegeben. []

1.1.12 nau($L_1,\ldots,L_n$). Seien $L_1,\ldots,L_n$ Literale. Den normalen allgemeinsten Unifikator von $L_1,\ldots,L_n$ bezeichnen wir, falls er existiert, mit $nau(L_1,\ldots,L_n)$.

1.1.13 Potentiell komplementäre Literale. Zwei Literale L und K heißen potentiell komplementär, wenn es für K eine U-Substitution θ gibt, so daß $\bar{L}$ und θK unifizierbar sind.

1.2.14 Lemma. L und K seien Literale und σ sei eine Substitution. Wenn σL und K potentiell komplementär sind, dann sind auch L und K potentiell komplementär.

Beweis. Siehe /He82/. []

1.2.15 Kombinationssubstitutionen. Seien $\sigma = \{X_1/t_1,\ldots,X_n/t_n\}$ und $\tau = \{Y_1/s_1,\ldots,Y_m/s_m\}$ Substitutionen. Dann heißt der Unifikator $\sigma*\tau := nau(PX_1\ldots X_nY_1\ldots Y_m,\ Pt_1\ldots t_ns_1\ldots s_m)$, wenn er existiert, die Kombinationssubstitution von σ und τ. Die Kombinationsoperation * ist kommutativ und assoziativ. Wenn $M = \{\sigma_1,\ldots,\sigma_n\}$ eine Menge von Substitutionen ist, dann heißt die Substitution $(*\sigma \mid \sigma \in M) := \sigma_1*\ldots*\sigma_n$, falls sie existiert, die Kombinationssubstitution von M. Eine Menge M von Substitutionen heißt konsistent, wenn die Kombinationssubstitution von M existiert.

1.2.16 Lemma.

(a) Wenn σ eine normale Substitution ist, dann ist $\sigma = \sigma*\sigma$.

(b) Für zwei konsistente Substitutionen σ und τ gilt:
$\sigma*\tau = (\sigma*\tau)\sigma = (\sigma*\tau)\tau = \sigma(\sigma*\tau) = \tau(\sigma*\tau)$.

(c) Für zwei normale Substitutionen σ und τ mit $Ber(\tau) \cap Var(\sigma) = \emptyset$ gilt:
$\sigma*\tau = \sigma\tau$.

(d) Wenn L und K variablendisjunkte Literale sind, und σ ein normaler Unifikator von L und K ist, dann gilt $\sigma = \sigma * nau(L,K)$.

Beweis. Siehe /He82/. []