Geordnete Resolution

Seminararbeit

von

Kerstin Susewind
(susewind@uni-koblenz.de)

Universität Koblenz
Fachbereich 4: Informatik

Wintersemester 2004/2005

Seminar: Entscheidungsverfahren
für logische Theorien

Dozent: Peter Baumgartner

Datum: 01.03.2005

Inhaltsverzeichnis

1. Einleitung

Seit Jahren beschäftigt sich die Forschung im Bereich „Automatische Beweisverfahren“ mit der Gewinnung neuer Entscheidungsverfahren für spezielle Theorien, die für computerunterstützte praktische Anwendungen erforderlich sind.
Im Bereich der Prädikatenlogik 1. Ordnung (d.h. keine Quantifizierungen von Prädikats- und Funktionssymbolen) wurde das *Resolutions*verfahren zu Genüge untersucht. Es ist jedoch nur ein Semi-Entscheidungsverfahren: Wendet man es auf eine unerfüllbare Formel an, erhält man garantiert daraus die leere Klausel und beweist somit die Unerfüllbarkeit; angewendet auf eine erfüllbare Formel ist es möglich, dass der Algorithmus niemals stoppt. In diesem Sinne gibt es kein allgemeingültiges Verfahren, das für eine beliebige prädikatenlogische Formel terminiert und die Frage der Erfüllbarkeit *oder* Unerfüllbarkeit entscheidet.
Um diesem Problem näher zu kommen, wurden die prädikatenlogischen Formeln anhand einfacher Eigenschaften - wie beispielsweise Reihenfolge und Anzahl von Existenz- und Allquantoren – in Klassen eingeteilt. Für einige dieser Klassen wurde deren Entscheidbarkeit bereits nachgewiesen und sie wurden somit als „lösbar“ bezeichnet.
Das allgemeine Resolutionsverfahren ist *kein* Entscheidungsverfahren für die meisten dieser lösbaren Klassen. Es ist erforderlich, weitere Techniken mit dem Resolutionsalgorithmus zu kombinieren um ein vollständiges Resolutionsverfahren zu gewinnen.
Zu *geordneter Resolution* zählt das in den nächsten Kapiteln vorgestellte Entscheidungsverfahren, das eine Kombination von Resolution und zwei weiteren Techniken namens A-*ordering* und *condensing* ist. Der Beweis zur Vollständigkeit dieses Verfahrens erfolgt über so genannte *semantische Bäume*, die auch gleichzeitig ein Modell für als erfüllbar ausgewiesene Formeln liefern.

2. Grundlagen

Zum besseren Verständnis der neu vorgestellten Techniken in den nächsten Kapiteln ist es erforderlich, einige Fachbegriffe zunächst zu wiederholen, da wichtige Definitionen je nach Quelle variieren können. Die folgenden Definitionen entstammen, leicht modifiziert, hauptsächlich aus [2] (s.u.).

2.1 Umwandlung einer Formel in Klauselform

Damit der Resolutionsalgorithmus auf eine Formel angewendet werden kann, wird sie zunächst in die so genannte Klauselform umgewandelt. Für eine gegebene Formel *F* mit

$F = \neg\exists x(P(x,z) \vee \forall y Q(x,f(y)))$

- wird diese bereinigt, d.h. hinter jedem Quantor muss eine andere Variable stehen;
- freie Variable gebunden (hier *z*): $F = \exists z(\neg\exists x(P(x,z) \vee \forall y Q(x,f(y))))$;
- *F* umgeformt in Pränexform: $F = \exists z \forall x \exists y(\neg P(x,z) \wedge \neg Q(x,f(y)))$;
- *F* umgeformt in Skolemform: $F = \forall x(\neg P(x,a) \wedge \neg Q(x,f(h(x))))$;
- *F* in KNF umgewandelt und schließlich
- in *Klauselform* ***c(F)*** geschrieben: $c(F) = \{\{\neg P(x,a)\},\{\neg Q(x,f(h(x)))\}\}$.

Die *Matrix F** einer Formel *F* ist die Formel *F* ohne Quantoren und der an sie gebundenen Variablen, für das obige Beispiel ist die Matrix also $F^* = (\neg P(x,a) \wedge \neg Q(x,f(h(x))))$.

2.2 Herbrand-Universum, Herbrand-Basis

Das *Herbrand-Universum D(F)* einer Formel *F* in Skolemform ist induktiv wie folgt definiert:

- Alle in *F* vorkommenden Konstanten sind in *D(F)*. Falls *F* keine Konstante enthält, so ist *a* in *D(F)*.
- Für jedes in *F* vorkommende *n*-stellige Funktionssymbol *f* und Terme $t_1, t_2, \ldots, t_n$ in *D(F)* ist der Term $f(t_1, t_2, \ldots, t_n)$ in *D(F).*

Beispiel: Für die Formel $F = \forall x \forall y P(x,f(y))$ ist $D(F) = \{a, f(a), f(f(a)), f(f(f(a))), ...\}$.

Die *Herbrand-Basis B(F)* einer Formel *F* in Skolemform ist die Menge von Atomen, die aus den Prädikatssymbolen von *F* und den Termen des Herbrand-Universums *D(F)* erzeugt werden.

Beispiel: Für $F = \forall x \forall y P(x,f(y))$ ist $B(F) = \{P(a,f(a)), P(f(a),f(a)), P(a,f(f(a))), ...\}$.

2.3 Grundsubstitution, Grundinstanz, Instanz

Eine Substitution, die alle freien Variablen (d.h. keine an einen Quantor gebundenen Variablen) in *F** durch variablenfreie Terme ersetzt, heißt *Grundsubstitution*. Auch die Elemente von *B(F)* sind Grundsubstitutionen, da alle Variablen durch variablenfreie Terme aus *H(F)* ersetzt wurden. Wenn alle Variablen in F* durch eine Grundsubstitution ersetzt werden, heißt das Resultat eine *Grundinstanz* von *F**.

Wenn allgemein alle Variablen in einer Formel *F* durch Substitution ersetzt werden, heißt die resultierende Formel eine *Instanz* von *F*.

2.4 Allgemeinster Unifikator, Resolution

Eine Substitution *sub* ist ein *Unifikator* einer (endlichen) Menge von Literalen **L** = {L_1, L_2, …, L_k}, falls L_1*sub* = L_2*sub* = … = L_k*sub*, d.h. durch Anwenden von *sub* auf jedes Literal in **L** entsteht ein und dasselbe Literal. Ein Unifikator *sub* einer Literalmenge **L** heißt *allgemeinster*

Unifikator von **L**, falls für jeden Unifikator *sub'* von **L** gilt, dass es eine Substitution *s* gibt mit *sub'* = *sub s*.

R ist ein *prädikatenlogischer Resolvent* zweier prädikatenlogischer Klauseln K_1 und K_2, falls folgendes gilt:

- Es gibt Substitutionen s_1 und s_2, die Variablenumbenennungen sind, so dass K_1s_1 und K_2s_2 keine gemeinsamen Variablen enthalten.
- Es gibt eine Menge von Literalen $L_1,...,L_m \in K_1s_1, m \geq 1$, und $L'_1,...,L'_n \in K_2s_2, n \geq 1$, so dass $\mathbf{L} = \{\overline{L}_1,...,\overline{L}_m, L'_1,...,L'_n\}$ unifizierbar ist. Es sei *sub* allgemeinster Unifikator von **L**.
- *R* hat die Form $R = ((K_1s_1 - \{L_1,...,L_m\}) \cup (K_2s_2 - \{L'_1,...,L'_n\}))sub$.

Des Weiteren ist für eine prädikatenlogische Klauselmenge *F* definiert:

Res(F) = $F \cup \{R \mid R$ *ist ein prädikatenlogischer Resolvent zweier Klauseln* $K_1, K_2 \in F\}$

$Res^0(F)$ = F

$Res^{n+1}(F)$ = $Res(Res^n(F))$ für $n \geq 0$

Res(F)* = $\bigcup_{n \geq 0} \mathrm{Res}^n(F)$

Die Anwendung von *Res* auf *F* ist die Konstruktion der endlichen Klausemenge $Res^t(F)$ für *t=0,1,….* Das Ziel ist die Herleitung der leeren Klausel, geschrieben □. Die Vollständigkeit und Korrektheit des *Resolutionsverfahrens Res* wird wie folgt ausgedrückt:

- $\square \in \mathrm{Res}^n(c(F))$ für ein $n \geq 0 \Leftrightarrow$ F ist unerfüllbar

Ein Resolutionsverfahren kann aber auch stoppen, ohne dass die leere Klausel erzeugt wird, z.B. wenn $Res^n(F) = Res^{n+1}(F)$ für ein beliebiges *n*. Sei $F \approx G$ für Klauselmengen *F* und *G*, falls jede Klausel in *F* eine Variante in *G* hat und umgekehrt. Dies bedeutet, dass die Anwendung eines Resolutionsverfahrens *Res* ohne Produktion der leeren Klausel hält, falls $\mathrm{Res}^n(F) \approx \mathrm{Res}^{n+1}(F)$ und $\square \notin \mathrm{Res}^n(F)$ für ein $n \geq 0$. Somit gilt folgende Implikation:

- [Für ein $n \geq 0 : (\mathrm{Res}^{n+1}(c(F)) \approx \mathrm{Res}^n(c(F)) \wedge \square \notin \mathrm{Res}^n(c(F)))] \Rightarrow$ F ist erfüllbar

Der Umkehrung der Implikation ist das bekannte Halteproblem: Es ist *möglich*, dass das Verfahren für eine erfüllbare Formel stoppt, da keine neuen Klauseln erzeugt werden, aber es ist nicht *garantiert*.

Wie bereits erwähnt, sind bestimmte Klassen von Formeln entscheidbar, d.h. für sie gilt die Implikation in beide Richtungen. In den folgenden Kapiteln wird die Herleitung eines Entscheidungsverfahrens für eine dieser Klassen unter Verwendung verschiedener Techniken vorgestellt. Eine wichtige Rolle spielen hierbei die so genannten *semantischen Bäume*, die im nächsten Kapitel behandelt werden.

3. Semantische Bäume

Semantische Bäume werden bei der Gewinnung eines Entscheidungsverfahrens für bestimmte Formelklassen zum Beweis der Vollständigkeit und der Herleitung eines Modells für erfüllbare Formeln verwendet.

Für eine beliebige Klauselmenge F sei $A_1, A_2, A_3,...$ eine sich nicht wiederholende Aufzählung ihrer Herbrand-Basis $B(F)$. Der *semantische Baum* T passend zu dieser Aufzählung ist ein (in der Regel unendlicher) binärer Baum, dessen Kanten mit den Elementen der Aufzählung und deren Negationen wie folgt beschriftet sind:

Die zwei die Wurzel verlassenden Kanten sind A_1 und $\neg A_1$. Falls A_i oder $\neg A_i$ die eingehende Kante eines Knotens kennzeichnet, so wird dieser Knoten mit A_{i+1} und $\neg A_{i+1}$ verlassen. Jedem Knoten von T kann man eine *Belegung* zuordnen, indem man die Menge der Literale bildet, die die Kanten von der Wurzel bis zum Knoten kennzeichnen. Die Belegung an der Wurzel ist die leere Klausel. Des Weiteren gilt für einen semantischen Baum folgendes:

- Eine Klausel C der Klauselmenge F *scheitert* an einem Knoten in T, falls das Komplement jedes Literals in einer Grundinstanz C_0 von C in der Belegung dieses Knotens enthalten ist.
- Ein Knoten heißt *failure node* für F, falls eine Klausel von F an diesem Knoten scheitert, aber keine Klausel in F näher der Wurzel auf diesem Pfad.
- T heißt *geschlossen* für F, falls es *failure nodes* an jedem Zweig von T gibt. Somit ist dann F unerfüllbar.
- Ein *inference node* ist ein Knoten, dessen zwei Söhne *failure nodes* sind.

Beispiel:

Die Formel $F = P(u) \wedge (\neg P(x) \vee Q(y)) \wedge \neg Q(f(z))$

hat die Klauselform $\{\{P(u)\},\{\neg P(x), Q(y)\},\{\neg Q(f(z))\}\}$

Klausel1 Klausel2 Klausel3

und die Herbrandbasis $B(F) = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a))),...\}$.

Aufzählung: A_1 A_2 A_3 A_4 A_5 A_6 …

Der semantische Baum T zu dieser Formel F sieht wie folgt aus:

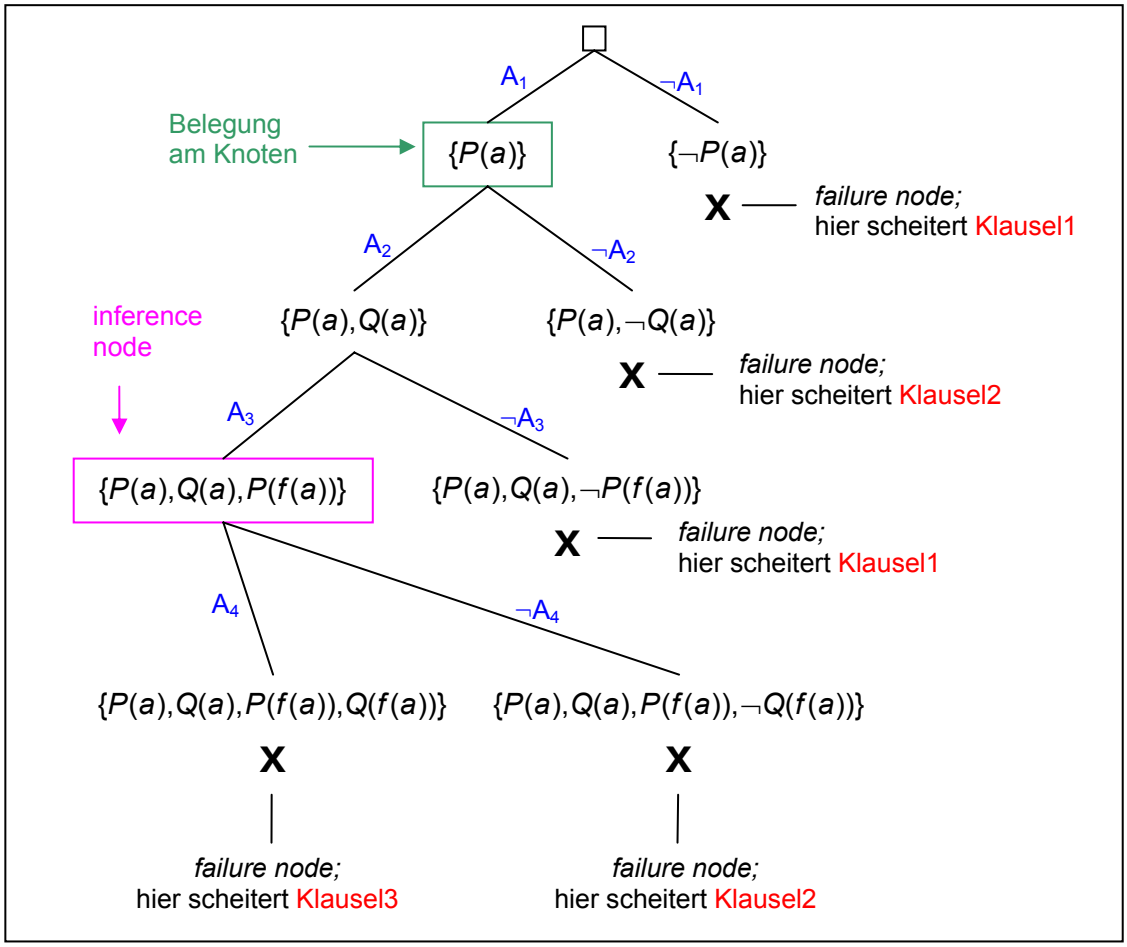

Abb. 1: *Beispiel für einen semantischen Baum*

Dieser semantische Baum *T* ist *geschlossen*, die zugehörige Formel *F* also unerfüllbar, sonst würde ein Pfad ohne *failure node* ein Modell für *F* bestimmen. Die zwei Klauseln, die unter dem *inference node* scheitern, formen eine Resolvente, die am *inference node* scheitert. Diese Resolvente kann keine Klausel der ursprünglichen Klauselmenge sein, da diese ja sonst schon am *inference node* gescheitert wäre. In diesem Fall wäre die Resolvente $E = \{\neg P(f(a))\}$ (entstanden aus Klausel2 und Klausel3) die am *inference node* scheitende Resolvente.

T hat somit auch weniger *failure nodes* für *Res(F)* als für *F* (diejenigen unterhalb eines *inference node*), da der *inference node* oder ein Knoten näher der Wurzel sie ersetzt hat. Wendet man *Res* wiederholt an, wird die Anzahl an *failure nodes* fortschreitend reduziert, wobei *T* aber immer geschlossen bleibt. Letztendlich muss es einen *failure node* für ein $Res^p(F)$ geben, wobei es sich dabei um die Wurzel handeln muss, da *T* geschlossen bleibt. Da die leere Klausel die einzige an der Wurzel scheiternde Klausel ist, muss $\square \in Res^p(F)$ sein. Somit ist *Res* ein über semantische Bäume vollständiges Resolutionsverfahren.

4. Entscheidungsverfahren und Resolution

Wie bereits erwähnt, ist es möglich, für bestimmte Formelklassen ein Entscheidungsverfahren basierend auf Resolution zu gewinnen. Einige dieser lösbaren Klassen, die Teilmengen des *reinen* Prädikatenkalküls 1. Ordnung (d.h. Formeln ohne Funktionssymbole) sind, werden wie folgt definiert:

Klasse	Struktur
Herbrand	Matrix = Konjunktion von Literalen
Bernays-Schonfinkel	Präfix $\exists^* \forall^*$
Ackermann	Präfix $\exists^* \forall \exists^*$
Gödel	Präfix $\exists^* \forall \forall \exists^*$

Das in Kapitel 2.4 vorgestellte allgemeine Resolutionsverfahren *Res* entscheidet diese Formelklassen (bis auf die Herbrand-Klasse) nicht. Es ist notwendig, die zwei in den nächsten Abschnitten vorgestellten Probleme in den Griff zu bekommen, um – unter Erhaltung der allgemeinen Vollständigkeit, d.h. Semi-Entscheidbarkeit für die Klasse *aller* Formeln – ein Entscheidungsverfahren für einige dieser Klassen zu gewinnen.

4.1 Wachsende funktionale Verschachtelung

Bei erfüllbaren Formeln aus den oben vorgestellten Klassen kann man bei fortschreitender Anwendung des allgemeinen Resolutionsverfahrens *Res* eine steigende funktionale Schachtelung in den Resolventen feststellen. Durch diese funktionale Schachtelung ist keine dieser produzierten Resolventen eine Variante einer vorherigen und das Verfahren kann nicht terminieren. Es ist somit auch kein Entscheidungsverfahren. Beispiel:

$F = \forall x \exists y((P(x) \vee P(y)) \wedge (\neg P(x) \vee \neg P(y)))$ ist eine erfüllbare Formel mit Klauselform $c(F) = \{\{P(x), P(f(x))\}, \{\neg P(x), \neg P(f(x))\}\}$. Durch wiederholte Resolventenbildung mit den Ausgangsklauseln werden folgende Klauseln produziert:

$\{P(x), \neg P(f(f(x)))\}, \{P(x), \neg P(f(f(f(f(x))))\}, \{P(x), \neg P(f(f(f(f(f(f(x))))))\}, \ldots$

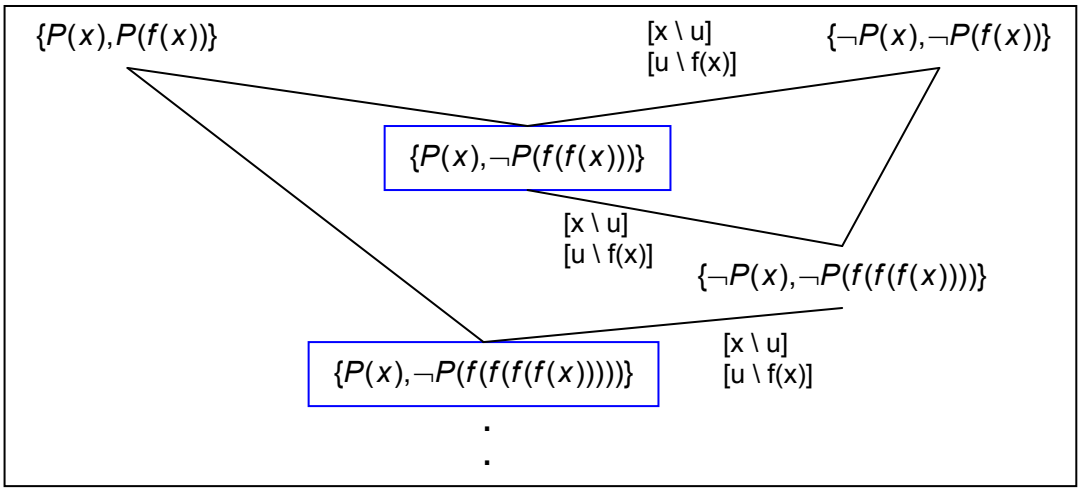

Abb. 2: *wachsende funktionale Verschachtelung*

4.2 Wachsende Klauselgröße

Neben der wachsenden funktionalen Verschachtelung kann es bei wiederholter Anwendung von *Res* zu einem Wachstum der Klauselgröße kommen, wie das folgende Beispiel für die Klausel F mit $c(F) = \{\{P(x_1, x_2), Q(x_2, x_3), R(x_1, x_3)\}, \{\neg R(y_1, y_2), Q(y_2, y_3), \neg P(y_1, y_3)\}\}$ illustriert: Es werden Klauseln der Form $\{P(x_1, x_2), Q(x_2, x_3), ..., Q(x_{n-1}, x_n), R(x_1, x_n)\}$ mit $n = 5,7,9...$ erzeugt.

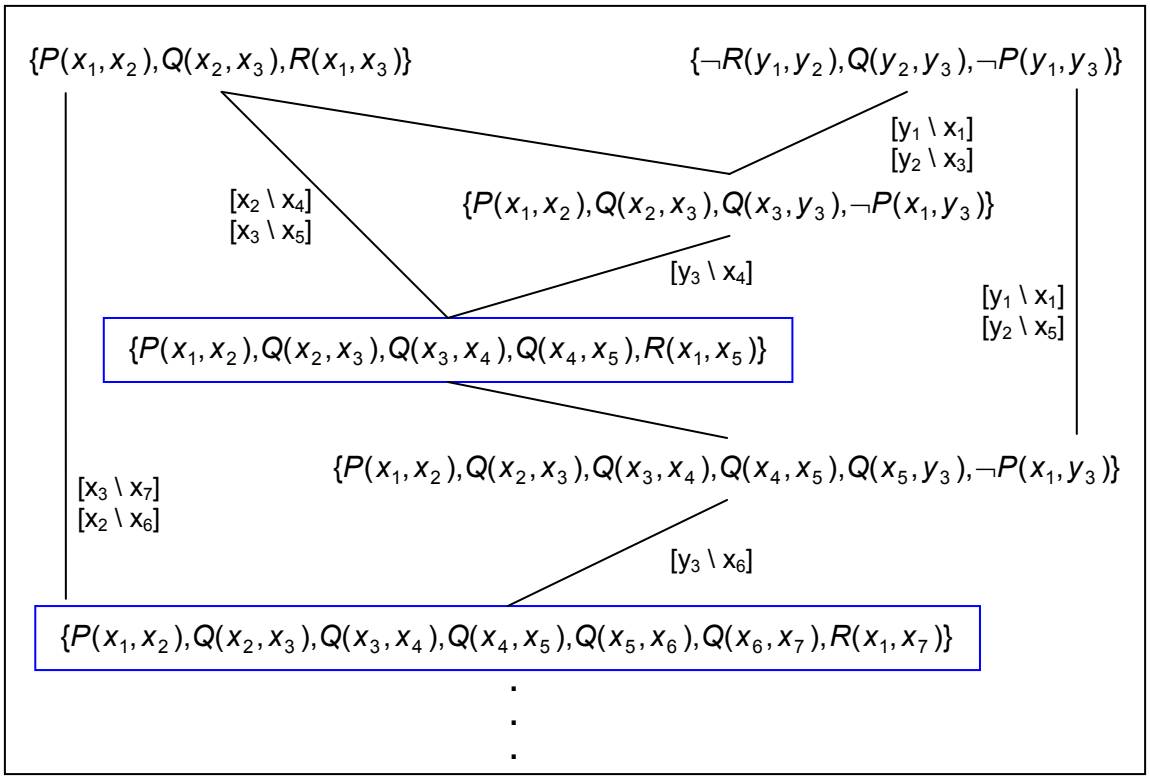

Abb. 3: *wachsende Klauselgröße*

Um zu zeigen, dass beispielsweise ein neues Resolutionsverfahren ***Res₂*** ein Entscheidungsverfahren für eine Formel *F* aus der Ackermann-Klasse ist, muss bewiesen werden, dass Wachstumsgrenzen für Schachtelung und für Klauselgröße für Formeln aus den obigen Klassen berechnet werden können. Die Existenz dieser Grenzen wird in den Beweisen benutzt, dass dieses Verfahren für eine Klauselmenge *c(F)* immer terminiert. Um diese zwei „Wachstumstypen“ zu beschränken, werden die Techniken *a-ordering* und *condensing* verwendet.

5. A-ordering

Um das Anwachsen der funktionalen Schachtelung zu beschränken, wird eine so genannte ***a-order*** (**A-Ordnung**) verwendet:

Eine **A-Ordnung $<_a$** ist eine irreflexive und transitive binäre Relation auf atomaren Formeln, so dass

(a) für jede Klauselmenge gilt: Es gibt eine Aufzählung $A_1, A_2, \ldots$ ihrer Herbrand-Basis, so dass für $A_i <_a A_j$ gelten muss: $i < j$

(b) wenn $A <_a B$, dann $A\theta <_a B\theta$ für jede Substitution θ.

Eine Resolvente E von Klauseln C und D mit resolviertem Atom L *gehorcht der* $<_a$*-Restriktion*, falls es kein in E auftretendes Atom M gibt mit $L <_a M$.

Um Variablen, Terme und Atome vergleichen und aufzählen zu können, beschreibt man sie durch ihre funktionale Größe:

- Größe einer Variable: 1
- Größe eines Terms $f(t_1,...,t_n) = 1$ + Maximum der Größen von $t_1,..,t_n$.
- Größe eines Atoms: Maximum der Größen seiner Argumente

Eine Aufzählung der Herbrand-Basis entspricht dann der Aufzählung der Atome nach steigender funktionaler Größe.

Um eine Grenze für die funktionale Schachtelung ermitteln zu können, werden zwei spezielle A-Ordnungen definiert:

- Die **A-Ordnung $<_1$** auf *Termen* ist folgendermaßen definiert:
  Zwei Terme sind *gleich*, wenn sie sich nur in ihren äußersten Funktionssymbolen unterscheiden. Für zwei Terme r, s gilt $r <_1 s$, falls entweder
  (1) r in s vorkommt oder
  (2) r von s *dominiert* wird, d.h. falls $r = f(t_1,...,t_m)$, $s = g(t_1,...,t_n)$ und $0 \le m < n$ oder
  (3) ein Term u, der r dominiert oder gleich r ist, in s vorkommt.
- Um *Atome* vergleichen zu können, wird aufbauend auf $<_1$ eine weitere **A-Ordnung $<_2$** definiert:
  Für zwei Atome A, B gilt $A <_2 B$ genau dann, wenn es ein Argument u in B gibt, so dass für jedes Argument t aus A gilt: $t <_1 u$.

Beispiele:

- $a <_1 f(x)$
- $f(x) <_1 g(x,y)$ (f(x) wird von g(x,y) dominiert)
- $P(x,f(x)) <_2 Q(g(x,y))$ ($x <_1 g(x,y)$ und $f(x) <_1 g(x,y)$)

Für Resolventen muss laut obiger Definition nun die $<_2$-Restriktion gelten, d.h. kein in einer Resolvente auftretendes Atom M darf in der dieser Ordnung entsprechenden Aufzählung der Herbrand-Basis *nach* dem resolvierten Atom L kommen ($L <_2 M$ darf nicht gelten). Auf diese Weise werden größere Atome, d.h. Atome mit höherer funktionaler Verschachtelung, bei der Resolution zuerst eliminiert.

Für gegebene Klauseln $C = \{P(x,f(x)), Q(g(x,y))\}$ und $D = \{P(u,v), \neg Q(w)\}$ ist somit

- Resolvente $E = \{P(x, f(x)), P(u, v)\}$ unter der $<_2$-Restriktion gültig, denn *P(x,f(x)) <₂ Q(g(x,y))* und *Q(g(x,y))* wurde eliminiert,
- Resolvente $F = \{Q(g(x, y)), Q(w)\}$ nicht unter der $<_2$-Restriktion gültig, denn *P(x,f(x)) <₂ Q(g(x,y))* und *Q(g(x,y))* tritt in der Resolvente auf.

6. Condensing

Die zweite Technik namens *condensing* wird verwendet, um den Wachstum der Klauselgröße bei fortschreitender Resolution zu beschränken.
Die *Verdichtung (condensation)* einer Klausel ist die kleinste Untermenge dieser Klausel, die auch gleichzeitig eine Instanz ist. Eine Verdichtung einer Klausel erzeugt man, indem man entweder

- all ihre Untermengen daraufhin untersucht, ob sie Instanzen sind oder
- Paare von Literalen sukzessiv unifiziert, bis keine Klausel mehr erzeugt werden kann, die eine Variante einer Untermenge dieser Klausel ist.

Eine Klausel lässt sich somit leicht durch einen Algorithmus verdichten. Ist eine Klausel bereits ihre eigene Verdichtung, so heißt sie *verdichtet (condensed)*.
Durch die Verdichtung können Klauseln schon vor der Resolution von Literalen befreit werden, die sonst zum Anwachsen der Klauselgröße beitragen könnten. Zwei Beispiele für Verdichtungen sind:

- Für *{P(x), P(f(a)), Q(z)}* ist die Verdichtung *{P(f(a)), Q(z)}*.
- *{P(w,x),P(x,y),P(y,z)}* ist bereits verdichtet.

Für ein Resolutionsverfahren *Res* ist ***Res*$_{cond}$** ein Resolutionsverfahren wie *Res* mit dem Zusatz, dass alle Anfangsklauseln und jede erzeugte Resolvente durch ihre Verdichtung ersetzt werden:

- *cond(F)* = Klauselmenge, die aus *F* entsteht, wenn man jede Klausel in *F* verdichtet
- $\mathrm{Re}\, s^0_{cond}(F) = cond(\mathrm{Re}\, s^0(F))$, $\mathrm{Re}\, s_{cond}(F) = F \cup cond(\mathrm{Re}\, s(F) - F)$

7. Entscheidungsverfahren für die Ackermann-Klasse

7.1 Vollständigkeit von Res_2

Die in den letzten beiden Kapiteln vorgestellten Techniken werden nun verwendet, um ein Entscheidungsverfahren basierend auf Resolution für die Ackermann-Klasse zu gewinnen. Hierzu muss zunächst gezeigt werden, dass das neue Resolutionsverfahren **Res_2** =

$Res_{<_2,cond}$, basierend auf A-Ordnung und Verdichtung, ein vollständiges Resolutionsverfahren ist:

- Es ist vollständig unter der $<_2$-Restriktion, da es für jede Klauselmenge eine zur A-Ordnung $<_2$ passende Aufzählung der Herbrand-Basis gibt: Atome mit geringerer Größe stehen in der Aufzählung vor den größeren. Haben Atome die gleiche Größe, werden sie durch $<_2$ geordnet, ansonsten lexikographisch.
- Ist *Res⁰(F)* unerfüllbar, so ist auch $Res^0_{cond}(F) = cond(Res^0(F))$ unerfüllbar, denn *cond(Res⁰(F))* enthält eine Teilmenge jedes Elements aus *Res⁰(F).* Durch die Definition des Scheiterns an einem Knoten gilt weiterhin folgendes:
  Auch Verdichtungen *C'* und *D'* zweier genau unterhalb eines *inference node w* scheiternden Klauseln *C* und *D* scheitern genau unterhalb von *w*, da sie Untermengen von *C* und *D* sind. Eine aus *C'* und *D'* erzeugte Resolvente *E* scheitert also am Knoten *w*. Die Verdichtung *E'* von *E*, die von Res_{cond} erzeugt wird, scheitert auch an *w*. Somit ist Res_{cond} vollständig über semantische Bäume und somit vollständig.
- Insgesamt ist damit $Res_{<_2,cond}$ vollständig.

7.2 v-Partitionen

Um eine Klauselgrößengrenze zu ermitteln, teilt man die Literale einer Klausel in Blöcke mit gemeinsamen Variablen ein, eine so genannte ***v-Partition***:

- Alle Grundliterale, d.h. Literale ohne Variablen, stehen im so genannten Grundblock,
- zwei verschiedene Blöcke enthalten keine gemeinsamen Variablen,
- die Partitionierung muss *feinstmöglich* sein.

Beispiel: Für die Klausel *{P(a), Q(f(b)), R(x,y), R(y,w), Q(f(z))}* mit Konstanten *a,b* und Variablen *w,x,y,z* ist

- {P(a), Q(f(b))} (Grundblock), {R(x,y), R(y,w)}, {Q(f(z))} eine akzeptierbare v-Partition
- {P(a), Q(f(b))} (Grundblock), {R(x,y), R(y,w), Q(f(z))} nicht akzeptierbar, da nicht feinstmöglich.

7.3 Akzeptierbare Eigenschaften

Der Beweis zur Existenz von Schachtelungs- und Klauselgrößengrenzen für Res_2 erfordert zusätzlich noch ein weiteres Werkzeug, die Definition bestimmter Eigenschaften von Klauseln:

Eine Eigenschaft **K** von Klauseln ist eine *akzeptierbare Eigenschaft*, wenn gilt:

(a) Ist **K** für eine Klausel gültig, so auch für jede Variante oder Teilmenge dieser Klausel. *(Also auch für Verdichtungen von Klauseln.)*

(b) Ist **K** für zwei Klauseln gültig, die sich keine Variablen teilen, so gilt **K** auch für die Vereinigung dieser Klauseln. *(K gilt auch für die Vereinigung zweier Blöcke.)*

(c) Ist eine endliche Menge von Prädikats- und Funktionssymbolen gegeben, so kann nur eine endliche Anzahl von nicht varianten, verdichteten Klauseln, für welche **K** gilt, erzeugt werden.

7.4 Das Entscheidungsverfahren Res_2

Damit **Res_2** ein Entscheidungsverfahren ist, muss insgesamt folgendes gelten:

- Es gibt eine Schachtelungsgrenze *g* derart, dass keine Klausel in $\mathrm{Re}\, s_2^p(c(F))$ für jedes $p \geq 0$ Terme größer als *g* besitzt.
- Es gibt eine Grenze *s* der Klauselgröße derart, dass keine Klausel in $\mathrm{Re}\, s_2^p(c(F))$ für jedes $p \geq 0$ mehr Literale als *s* besitzt.
- Falls *F* erfüllbar ist, muss es ein $p \geq 0$ geben, so dass alle nach Level *p* erzeugten Klauseln Varianten der bereits vorher generierten Klauseln sind. Dann gilt:

  (1) $\mathrm{Re}\, s_2^p(c(F)) \approx \mathrm{Re}\, s_2^{p+1}(c(F))$

  (2) *Res_2* wird halten ohne die leere Klausel zu erzeugen

  (3) *F* wird als erfüllbar ausgewiesen

Um zu *beweisen*, dass *Res_2* ein Entscheidungsverfahren für die Ackermann-Klasse ist, muss man dementsprechend:

1) eine akzeptierbare Eigenschaft **K** definieren,

2) beweisen, dass alle Klauseln in *c(F)* für jede Formel *F* der Ackermann-Klasse unter **K** gültig sind,

3) zeigen, dass jede von *Res_2* aus unter **K** gültigen Klauseln erzeugte Resolvente ebenfalls gültig unter **K** ist.

Bedingung **3** kann umgeschrieben werden, wenn man die Restriktionen der A-Ordnung für die Resolventenbildung betrachtet:

3') Es muss folgendes bewiesen werden:

Für eine unter **K** gültige Klausel *C* mit Literalen *L* und *M,* die mit dem allgemeinsten Unifikator θ unifizierbar sind, darf es *kein* Literal *N* in *C* geben, so dass $L\theta = M\theta <_2 N\theta$ gilt. Gibt es kein solches Literal *N*, dann ist auch $C\theta$ unter **K** gültig.

Nun zum Beweis:

Wie in Kapitel 4 vorgestellt, haben Formeln der Ackermann-Klasse den Präfix $\exists^*\forall\exists^*$. Für eine Formel *F* dieser Klasse mit Präfix $\exists^m\forall\exists^n$ besitzt c(*F*) also

- *eine* Variable *v* (denn es gibt nur *einen* Allquantor)

- m Konstanten, z.B. $a_1,...,a_m$
- n einstellige (da nur eine Variable) Funktionssymbole $f_1,...,f_n$

Nun wird zunächst gemäß **1)** eine Eigenschaft **ACK** definiert:

(i) Es kommt kein Literal vor mit einer Größe >2.

(ii) Genau *eine* Variable kommt in jedem Block der v-Partition vor (außer dem Grundblock selbstverständlich).

(iii) Jedes Argument eines Literals in einem Block (außer Grundblock) mit Variable v muss ein Element der Menge $\{a_1,...,a_m,v,f_1(v),...,f_n(v)\}$ sein.

ACK ist akzeptierbar:
Die Bedingungen (i)-(iii) sind selbstverständlich für jede Variante oder Teilmenge einer Klausel gültig. Auch vereinte Klauseln besitzen nur Literale mit Größe $\leq$2, und da es nur eine Variable gibt, kann man auch vereinte und verdichtete Klauseln entsprechend leicht in Blöcke partitionieren mit Literalen, deren Argumente aus der in (iii) gegebenen Menge entstammen.
Für Klauseln der Ackermann-Klasse in Klauselform *c(F)* gilt, dass die einzigen auftretenden Funktionen Skolemfunktionen sein müssen, da wir vom *reinen* Prädikatenkalkül 1. Ordnung ausgehen (siehe Kapitel 4). Dementsprechend können Klauseln in *c(F)* auch nur Literale mit einer Größe $\leq$2 besitzen und in jedem Block (außer Grundblock) kommen nur Literale aus der in (iii) definierten Menge vor. Da in *c(F)* auch nur eine Variable vorkommt, kann jeder Block der v-Partition höchstens eine Variable enthalten.

Nun zum Problem der wachsenden Klauselgröße. Man kann bestimmen, aus wie vielen Elementen ein Block maximal bestehen kann. Für m Konstanten, n einstellige Funktionssymbole, *eine* Variable und k verschiedene Prädikatssymbole gilt:
Ein Prädikatssymbol hat (wie Terme) eine bestimmte *Stelligkeit deg(P)*, die angibt, wie viele Argumente es besitzt. Beispielsweise hat P(f(x)) die Stelligkeit 1, P(a,x) die Stelligkeit 2. Diese Argumente können bei einer Formel aus der Ackermann-Klasse eine von m Konstanten, *die* Variable oder eine von n Skolemfunktionen sein. Dies macht insgesamt $m+n+1$ Möglichkeiten für ein Argument. Für das gesamte Prädikatssymbol P mit Stelligkeit $deg(P)$ sind dies alle möglichen Argumentkombinationen, also $(m+n+1)^{deg(P)}$. Berechnet man dies für alle in einem Block auftretenden k Prädikatssymbole P_i, so erhält man die Summe $\sum_{i=1}^{k}(m+n+1)^{\deg(P_i)}$. Da Literale auch negiert vorkommen können, kann ein Block maximal aus $2\sum_{i=1}^{k}(m+n+1)^{\deg(P_i)}$ Elementen bestehen. Nimmt man den extremsten Fall an, dass

jedes Prädikatssymbol die maximale Stelligkeit q aller auftretenden Prädikatssymbole hat, erhält man folgende Grenze der Klauselgröße: $\sum_{i=1}^{k}(m+n+1)^{\deg(P_i)} \leq 2k(m+n+1)^q$.

Da eine Formel der Ackermann-Klasse nur eine Variable und einstellige Funktionssymbole besitzt, kann nicht mehr als eine Variable in einem Literal auftreten. Ist $f_i(v)$ ein Argument eines Literals (dies ist dann automatisch eine Instanz eines Literals in $c(F)$), so kann entsprechend kein Term der Form $f_j(a_k)$ in diesem Literal auftreten.
Damit erfüllen alle von Res_2 produzierten Klauseln die Bedingungen (ii) und (iii) von ACK.

Nun zu (i): Es bleibt zu zeigen, dass in keinem Resolutionsschritt funktionale Schachtelung auftreten kann.
Angenommen, bei der Unifikation zweier Literale L und M einer Klausel C solle funktionale Schachtelung auftreten, wobei C unter ACK gültig ist. Literal L müsste z.B. als Argument eine Variable v besitzen und M dann entsprechend einen Term $f_j(t)$, t eine andere Variable (z.B. durch Umbenennung) oder Konstante. Soll nun funktionale Schachtelung auftreten, muss ein Term $f_i(v)$ in L oder einem anderen Literal N in C vorkommen.
Wenn $f_i(v)$ ein Argument von N ist, widerspricht das der $<_2$-Restriktion: Bei der Unifikation von L und M mit der Substitution $[v \setminus f_j(t)]$ entstehen Terme $f_i(f_j(t))$ in N und $f_j(t)$ in L und somit wäre $L\theta <_2 N\theta$, was laut ***3')*** nicht erlaubt ist. Also muss L zum Ausgleich ein Argument $f_k(v)$ besitzen. In diesem Fall kann aber keine Unifikation stattfinden, denn die einzigen möglichen Argumente in M außer $f_j(t)$ sind t, $f_p(t)$ oder eine Konstante (= Terme aus der Menge in (iii)), aber nichts dergleichen kann mit $f_k(f_j(t))$ unifiziert werden.
Somit ist auch keine funktionale Schachtelung möglich und es kommen nur Literale mit der Größe ≤2 vor.

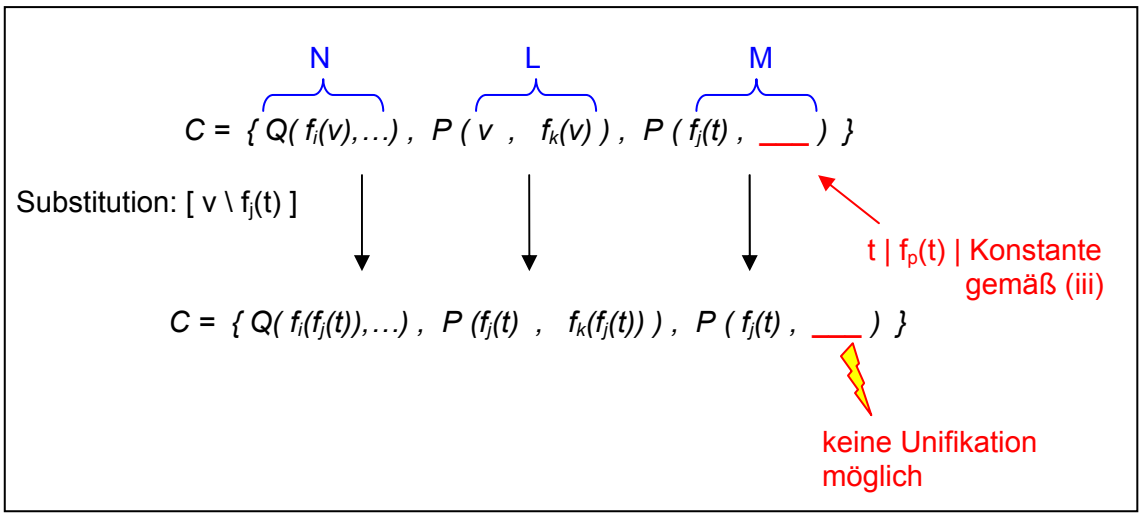

Abb. 4: *Funktionale Schachtelung und Unifikation*

Insgesamt wurde damit bewiesen, dass Res_2 ein Entscheidungsverfahren für die Ackermann-Klasse ist.

8. Modellkonstruktion

Wir wissen nun, dass es für *c(F)* einen semantischen Baum *T* gibt basierend auf der Aufzählung, für die Res_2 vollständig über semantische Bäume ist. Wenn *c(F)* erfüllbar ist, dann ist *T* nicht geschlossen.
Es gibt somit im Baum mindestens einen Zweig, auf dem es keinen *failure node* gibt. Die Literale auf diesem Zweig bestimmen ein Modell für *c(F)*. Gibt es mehrere solcher Zweige, wird willkürlich der linkste genommen. Die Zweige sind in der Regel unendlich.
Die Spezifikation eines unendlichen Modells erhält man über das Herbrand-Universum über *F*, wobei für jedes *Grundatom Q* entschieden wird, ob es *TRUE* oder *FALSE* ist im Modell. Dies geschieht, indem der offene Zweig ab der Wurzel abgelaufen wird. Schließlich muss man an eine mit *Q* oder $\neg Q$ beschriftete Kante gelangen. Dort wird gestoppt und *Q* entsprechend *TRUE* oder *FALSE* zugewiesen.

9. Quellen

[1] W.H. Joyner Jr. Resolution Strategies as Decision Procedures. *Journal of the ACM*, 23(3):398-417, 1976.

[2] Schöning, Uwe (2000): „Logik für Informatiker“ 5. Auflage. Heidelberg / Berlin: Spektrum Akademischer Verlag.