5 Resolution

Ein Kalkül soll die semantische Folgerungsbeziehung durch syntaktische Manipulation nachbilden, d.h. genau dann wenn $F \models G$, soll es möglich sein, entweder G aus F durch syntaktische Manipulation abzuleiten ($F \vdash G$, positiver Beweis) oder $F \wedge \neg G$ durch syntaktische Manipulation zu widerlegen ($F \wedge \neg G \vdash false$, Widerlegungsbeweis). Für jeden Kalkül muß die Korrektheit gezeigt werden, d.h. wann immer $F \vdash G$, dann $F \models G$. Die Vollständigkeit, d.h. wann immer $F \models G$, dann $F \vdash G$ ist nicht notwendig. Was möglichst gelten sollte (aber bei manchen Logiken nicht möglich ist), ist die Widerlegungsvollständigkeit, d.h. wann immer $F \models G$ dann $F \wedge \neg G \vdash false$. Für PL_1 gibt es eine ganze Reihe unterschiedlicher Kalküle. Ein wichtiger und gut automatisierbarer ist der 1963 von John Alan Robinson entwickelte Resolutionskalkül. Er arbeitet in erster Linie auf Klauseln.

5.1 Grundresolution: Resolution ohne Unifikation

Im folgenden schreiben wir Klauseln teilweise als Folge von Literalen: $L_1, \dots L_n$ und behandeln diese als wären es Multimengen. (teilweise auch als Mengen).

Definition 5.1. *Resolution im Fall direkt komplementärer Resolutionsliterale.*
Elternklausel 1: $L, K_1, \dots, K_m$
Elternklausel 2: $\neg L, N_1, \dots, N_n$
Resolvente: $\overline{K_1, \dots, K_m, N_1, \dots, N_n}$

Hierbei kann es passieren, daß die Resolvente keine Literale mehr enthält. Diese Klausel nennt man die *leere Klausel*; Bezeichnung: $\Box$. Sie wird als "falsch" interpretiert und stellt i.a. den gesuchten Widerspruch dar.

Beispiel 3.2.14 weiter fortgesetzt:

Beispiel 5.2. siehe Beispiel 4.34
A1: Dieb(Anton), Dieb(Ede), Dieb(Karl)
A2: ¬ Dieb(Anton), Dieb(Ede), Dieb(Karl)
A3: ¬ Dieb(Karl), Dieb(Ede), Dieb(Anton)
A4a: ¬ Dieb(Ede), ¬ Dieb(Anton)
A4b: ¬ Dieb(Ede), ¬ Dieb(Karl)
A5: ¬ Dieb(Anton),¬ Dieb(Karl)
Resolutionsableitung:

A2,2 & A4a,1 ⊢ R1: ¬ Dieb(Anton), Dieb(Karl)
R1,2 & A5,2 ⊢ R2: ¬ Dieb(Anton)
R2 & A3,3 ⊢ R3: ¬ Dieb(Karl), Dieb(Ede)
R3,2 & A4b,1 ⊢ R4: ¬ Dieb(Karl)
R4 & A1,3 ⊢ R5: Dieb(Anton), Dieb(Ede)
R5,1 & R2 ⊢ R6: Dieb(Ede)
Also, Ede wars.

Aussage 5.3. *Die Grund-Resolution ist korrekt:*

$$\begin{array}{ll} C_1 & := L, K_1, \ldots, K_m \\ C2 & :=: \neg L, N_1, \ldots, N_n \\ \hline R & = K_1, \ldots, K_m, N_1, \ldots, N_n \end{array}$$

Dann gilt $C_1 \wedge C_2 \models R$.

Beweis. Wir müssen zeigen: Jede Interpretation, die die beiden Elternklauseln wahr macht, macht auch die Resolvente wahr. Das geht durch einfache Fallunterscheidung:

Falls L wahr ist, muß $\neg L$ falsch sein. Da C_2 wahr ist, muß ein N_i wahr sein. Da dieses Literal in R vorkommt, ist auch R (als Disjunktion betrachtet) wahr. Falls L falsch ist, muß ein K_j wahr sein. Da das ebenfalls in der Resolvente vorkommt, ist R auch in diesem Fall wahr. □

Bemerkung 5.4. Im Sinne der Herleitbarkeit ist Resolution unvollständig: Nicht jede Formel, die semantisch folgt, läßt sich durch Anwenden der Resolution ableiten: Denn $P \models P \vee Q$. aber auf P alleine kann man keine Resolution anwenden.

Für den eingeschränkten Fall, daß die Klauselmenge keine Variablen enthält (Grundfall) können wir schon die Widerlegungsvollständigkeit der Resolution beweisen.

Satz 5.5. *(Widerlegungsvollständigkeit der Grundresolution) Jede endliche unerfüllbare Grundklauselmenge läßt sich durch Resolution widerlegen.*

Beweis. siehe Beweis von Satz 2.31 □

Resolution im allgemeinen Fall Für den allgemeinen Fall, wenn in den Literalen auch Variablen vorkommen, benötigt man eine zusätzliche Operation, um potentielle Resolutionspartner, d.h. Literale mit gleichem Prädikat und verschiedenem Vorzeichen durch Einsetzung von Termen für Variablen komplementär gleich zu machen. Dazu führen wir zunächst das Konzept der Substitution ein.

Definition 5.6. *(Substitution)*
Eine Substitution σ ist eine Abbildung, die Terme auf Terme abbildet und die nur endlich viele Variablen verändert. Zusätzlich gilt: $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$. D.h. die Substitution σ kann man sehen als gleichzeitige Ersetzung der Variablen x durch den Term $\sigma(x)$.

Substitutionen werden meist geschrieben wie eine Menge von Variable - Term Paaren:

$$\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_1\}$$

Mit Hilfe der rekursiven Definition:

$$\sigma(t) := \begin{cases} t_i & \text{falls } t \equiv x_i \text{ und } \sigma(x_i) = t_i \\ f(\sigma(t_1), \ldots, \sigma(t_n)) & \text{falls } t \equiv f(t_1, \ldots t_n)) \end{cases}$$

lassen sich solche Mengen von Variable - Term Paaren korrekt als (mit Datenstruktur verträgliche) Abbildung auf beliebigen Termen definieren. Entsprechend kann man die Anwendung von Substitutionen auf Literale und Klauseln definieren.

Beispiel 5.7.

$$\begin{array}{ll}
\sigma = \{x \mapsto a\} & \sigma(x) = a, \sigma(f(x,x)) = f(a,a) \\
\sigma = \{x \mapsto g(x)\} & \sigma(x) = g(x), \sigma(f(x,x)) = f(g(x), g(x)), \\
 & \sigma(\sigma(x)) = g(g(x)) \\
\sigma = \{x \mapsto y, y \mapsto a\} & \sigma(x) = y, \sigma(\sigma(x)) = a, \\
 & \sigma(f(x,y)) = f(y,a) \\
\sigma = \{x \mapsto y, y \mapsto x\} & \sigma(x) = y, \sigma(f(x,y)) = f(y,x)
\end{array}$$

Definition 5.8. *Für eine Substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ heißt*

$$\begin{array}{lll}
dom(\sigma) & := \{x_1, \ldots, x_n\} & \text{die Domain} \\
cod(\sigma) & := \{t_1, \ldots, t_n\} & \text{die Codomain}
\end{array}$$

Substitutionen σ mit $\sigma(\sigma(t)) = \sigma(t)$ für alle Terme t heißen idempotent*.*

Die Komposition *$\sigma\tau$ von Substitutionen σ und τ ist definiert als: $\sigma\tau(t) := \sigma(\tau(t))$.*

Gleichheit (modulo einer Menge von Variablen W) zwischen Substitutionen σ und τ wird folgendermaßen definiert: $\sigma = \tau[W]$ gdw. $\sigma x = \tau x$ für alle $x \in W$.

Eine Instantiierungrelation *$\leq [W]$ zwischen Substitutionen σ und τ (für eine Menge von Variablen W) ist folgendermaßen definiert:*

$\sigma \leq \tau[W]$ gdw. es existiert eine Substitution λ mit $\lambda\sigma = \tau[W]$. Zwei Substitutionen σ und τ sind äquivalent*, d.h. $\sigma \approx t[W]$ gdw. $\sigma \leq \tau[W]$ und $\tau \leq \sigma[W]$ gilt.Ist W die Menge aller Variablen, so kann man W weglassen.*

Bemerkung 5.9. Die Relationen $\leq [W]$ ist reflexiv und transitiv, aber nicht antisymmetrisch. D.h. es ist eine Präordnung.

Die Relation $\approx [W]$ ist eine Äquivalenzrelation.

Eine Substitution ist genau dann idempotent wenn die Variablen der Domain nicht in der Codomain vorkommen.

Beispiel 5.10.

Komposition:

- $\{x \mapsto a\}\{y \mapsto b\} = \{x \mapsto a, y \mapsto b\}$
- $\{y \mapsto b\}\{x \mapsto f(y)\} = \{x \mapsto f(b), y \mapsto b\}$
- $\{x \mapsto b\}\{x \mapsto a\} = \{x \mapsto a\}$

Instantiierungsrelation:

- $\{x \mapsto y\} \leq \{x \mapsto a\}[x]$
- $\{x \mapsto y\} \leq \{x \mapsto a\}[x, y]$ gilt nicht !
- $\{x \mapsto y\} \leq \{x \mapsto a, y \mapsto a\}$
- $\{x \mapsto f(x)\} \leq \{x \mapsto f(a)\}$

Äquivalenz: $\{x \mapsto y, y \mapsto x\} \approx Id$ (Id ist die identische Substitution)

Für die Äquivalenz von Substitutionen gilt: $\sigma \approx \tau[W]$ gdw. $\sigma = \rho\tau[W]$ für eine Variablenpermutation ρ. Eine Variablenpermutation ρ ist eine Substitution, die Variablen permutiert, d.h. $Dom(\rho) = Cod(\rho)$ und ρ bildet Variablen auf Variablen ab. (Beweis Übungsaufgabe)

Definition 5.11. *(Resolution mit Unifikation)*

$$
\begin{array}{lll}
\textit{Elternklausel 1:} & L, K_1, \ldots, K_m & \sigma \textit{ ist eine Substitution (Unifikator)} \\
\textit{Elternklausel 2:} & \neg L', N_1, \ldots, N_n & \sigma(L) = \sigma(L') \\
\hline
\textit{Resolvente:} & \sigma(K_1, \ldots, K_m, N_1, \ldots, N_n) &
\end{array}
$$

Die Operation auf einer Klauselmenge, die eine Klausel C auswählt, auf diese eine Substitution σ anwendet und $\sigma(C)$ zur Klauselmenge hinzufügt, ist korrekt. Damit ist auch die allgemeine Resolution als Folge von Variableneinsetzung und Resolution korrekt.

Beispiel 5.12. Dieses Beispiel (Eine Variante der Russelschen Antinomie) zeigt, daß noch eine Erweiterung der Resolution, die Faktorisierung, notwendig ist. Die Aussage ist: Der Friseur rasiert alle, die sich nicht selbst rasieren:

$$\forall x : \neg(rasiert(x, x) \Leftrightarrow rasiert(Friseur, x)$$

$$
\begin{array}{ll}
rasiert(x, x), rasiert(Friseur, x) & \sigma = \{x \mapsto Friseur, y \mapsto Friseur\} \\
\neg rasiert(Friseur, y), \neg rasiert(y, y) & \\
\hline
rasiert(Friseur, Friseur), \neg rasiert(Friseur, Friseur) &
\end{array}
$$

Die Klauseln sind widersprüchlich, was aber ohne eine Verschmelzung der Literale mittels Resolution nicht ableitbar ist. In der folgenden Herleitung werden die Variablen mit "Friseur"instantiiert, und dann die Literale verschmolzen.

$$rasiert(x, x), rasiert(Friseur, x) \vdash rasiert(Friseur, Friseur)$$
$$\neg rasiert(Friseur, y), \neg rasiert(y, y) \vdash \neg rasiert(Friseur, Friseur)$$

Danach ist es möglich, diese beiden Literale durch Resolution zur leeren Klausel abzuleiten.

Definition 5.13. (Faktorisierung)

$$
\begin{array}{lll}
\textit{Elternklausel:} & L, L', K_1, \ldots, K_m & \sigma(L) = \sigma(L') \\
\hline
\textit{Faktor:} & \sigma(L, K_1, \ldots, K_m) &
\end{array}
$$

Damit besteht der Resolutionskalkül jetzt aus Resolution und Faktorisierung.

Definition 5.14. *Der Resolutionskalkül transformiert Klauselmengen S wie folgt:*

1. *$S \rightarrow S \cup \{R\}$, wobei R eine Resolvente von zwei (nicht notwendig verschiedenen) Klauseln aus S ist.*
2. *$S \rightarrow S \cup \{F\}$, wobei F ein Faktor einer Klausel aus S ist.*

Der Resolutionskalkül terminiert mit Erfolg, wenn die leere Klausel abgeleitet wurde, d.h. wenn $\Box \in S$.

Bei Klauselmengen nehmen wir wie üblich an, daß die Klauseln variablendisjunkt sind.

Beispiel 5.15. Wir wollen die Transitivität der Teilmengenrelation mit Resolution beweisen. Wir starten mit der Definition von $\subseteq$ unter Benutzung von $\in$:

$$\forall x, y : x \subseteq y \Leftrightarrow \forall w : w \in x \Rightarrow w \in y$$

Das zu beweisende Theorem ist:

$$\forall x, y, z : x \subseteq y \wedge y \subseteq z \Rightarrow x \subseteq z$$

Umwandlung in Klauselform ergibt:

H1: $\neg x \subseteq y, \neg w \in x, w \in y$	($\Rightarrow$ Teil der Definition)
H2: $x \subseteq y, f(x, y) \in x$	(zwei $\Leftarrow$ Teile der Definition,
H3: $x \subseteq y, \neg f(x, y) \in y$	f ist die Skolem Funktion für w)
C1: $a \subseteq b$	(drei Teile der negierten Behauptung,
C2: $b \subseteq c$	a, b, c sind Skolem Konstanten für x, y, z)
C3: $\neg a \subseteq c$	

Resolutionswiderlegung:

H1,1 & C1,	$\{x \mapsto a, y \mapsto b\}$	$\vdash$ R1: $\neg w \in a, w \in b$
H1,1 & C2,	$\{x \mapsto b, y \mapsto c\}$	$\vdash$ R2: $\neg w \in b, w \in c$
H2,2 & R1,1,	$\{x \mapsto a, w \mapsto f(a, y)\}$	$\vdash$ R3: $a \subseteq y, f(a, y) \in b$
H3,2 & R2,2,	$\{y \mapsto c, w \mapsto f(x, c)\}$	$\vdash$ R4: $x \subseteq c, \neg f(x, c) \in b$
R3,2 & R4,2,	$\{x \mapsto a, y \mapsto c\}$	$\vdash$ R5: $a \subseteq c, a \subseteq c$
R5 & (Faktorisierung)		$\vdash$ R6: $a \subseteq c$
R6 & C3		$\vdash$ R7: $\Box$

5.2 Unifikation

Die Resolutions- und Faktorisierungsregel verwenden Substitutionen (Unifikatoren), die zwei Atome syntaktisch gleich machen. Meist will man jedoch nicht irgendeinen Unifikator, sondern einen möglichst allgemeinen. Was das bedeutet, zeigen die folgenden Beispiele:

Beispiel 5.16. Unifikatoren und allgemeinste Unifikatoren.

$$\begin{array}{ll} P(x), Q(x) & \\ \neg P(y), R(y) & \sigma = \{x \mapsto a, y \mapsto a\} \\ \hline Q(a), R(a) & \sigma \text{ ist ein Unifikator} \end{array}$$

$$\begin{array}{ll} P(x), Q(x) & \\ \neg P(y), R(y) & \sigma = \{x \mapsto y\} \\ \hline Q(y), R(y) & \sigma \text{ ist ein allgemeinster Unifikator} \end{array}$$

Fragen:

- Was heißt "allgemeinster" Unifikator?
- Wieviele gibt es davon? ($\sigma' = \{y \mapsto x\}$ im obigen Beispiel ist offensichtlich auch einer.)
- Wie berechnet man sie?

Ein allgemeinster Unifikator (von zwei Atomen) kann man intuitiv dadurch erklären, daß es eine Substitution ist, die zwei Terme oder Atome gleich macht, und möglichst wenig instantiiert. Optimal ist es dann, wenn alle Unifikatoren durch weitere Einsetzung in den allgemeinsten Unifikator erzeugt werden können.

Definition 5.17. *Unifikatoren und allgemeinste Unifikatoren.*

Seien s und t die zu unifizierenden Terme (Atome) und $W := FV(s,t)$. Eine Substitution σ heißt Unifikator *(von s,t), wenn $\sigma(s) = \sigma(t)$. Die Menge aller Unifikatoren bezeichnet man auch mit $U(s,t)$,*

Eine Substitution σ heißt allgemeinster Unifikator *für zwei Terme s und t wenn*

σ ein Unifikator ist	*(Korrektheit)*
für alle Unifikatoren τ gilt $\sigma \leq \tau[W]$	*(Vollständigkeit)*

Der allgemeinste Unifikator ist i.a. nur eindeutig bis auf $\equiv [W]$:

Beispiel 5.18. Es gibt (mindestens) zwei allgemeinste Unifikatoren für x und y: $\{x \mapsto y\}$, $\{y \mapsto x\}$, die sich aber nur durch Variablenumbenennung unterscheiden.

Die Beispiele legen die Vermutung nahe, daß es bis auf Variablenumbenennung immer einen allgemeinsten Unifikator gibt. Wir geben den Unifikationsalgorithmus in Form eines Regelpaketes an, das auf Mengen von Gleichungen operiert.

Definition 5.19. *Unifikationsalgorithmus $U1$:*

Eingabe: *zwei Terme oder Atome s und t:*
Ausgabe: *"nicht unifizierbar" oder einen allgemeinsten Unifikator:*

Zustände: *auf denen der Algorithmus operiert: Eine Menge* Γ *von Gleichungen.*

Initialzustand: $\Gamma_0 = \{s \stackrel{?}{=} t\}$.

Unifikationsregeln:

$$\frac{f(s_1,\ldots,s_n) \stackrel{?}{=} f(t_1,\ldots,t_n), \Gamma}{s_1 \stackrel{?}{=} t_1 \ , \ \ldots \ , \ s_n \stackrel{?}{=} t_n, \Gamma} \qquad \textit{(Dekomposition)}$$

$$\frac{x \stackrel{?}{=} x, \Gamma}{\Gamma} \qquad \textit{(Tautologie)}$$

$$\frac{x \stackrel{?}{=} t, \Gamma}{x \stackrel{?}{=} t, \{x \mapsto t\}\Gamma} \qquad x \in FV(\Gamma),\ x \notin FV(t) \qquad \textit{(Anwendung)}$$

$$\frac{t \stackrel{?}{=} x, \Gamma}{x \stackrel{?}{=} t, \Gamma} \qquad t \notin V \qquad \textit{(Orientierung)}$$

Abbruchbedingungen:

$$\frac{f(\ldots) \stackrel{?}{=} g(\ldots), \Gamma}{Fail} \qquad \textit{wenn } f \neq g \qquad \textit{(Clash)}$$

$$\frac{x \stackrel{?}{=} t, \Gamma}{Fail} \qquad \textit{wenn } x \in FV(t) \textit{ vorkommt (occurs check Fehler)}$$

und $t \neq x$

Steuerung:

Starte mit $\Gamma = \Gamma_0$*, und transformiere* Γ *solange durch (nichtdeterministische) Anwendung der Regeln, bis entweder eine Abbruchbedingung erfüllt ist oder keine Regel mehr anwendbar ist. Falls eine Abbruchbedingung erfüllt ist, terminiere mit "nicht unifizierbar". Falls keine Regel mehr anwendbar ist, hat die Gleichungsmenge die Form* $\{x_1 \stackrel{?}{=} t_1, \ldots, x_k \stackrel{?}{=} t_k\}$*, wobei keine der Variablen* x_i *in einem* t_j *vorkommt; d.h. sie ist in gelöster Form. Das Resultat ist dann* $\{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$*. (siehe Lemma 5.22).*

Beispiel 5.20. Unifikation von zwei Termen durch Anwendung der obigen Regeln:

$$\{k(f(x, g(a, y)), g(x, h(y)) \stackrel{?}{=} k(f(h(y), g(y, a)), g(z, z))\}$$

$$\to \{f(x, g(a, y)) \stackrel{?}{=} f(h(y), g(y, a)), g(x, h(y)) \stackrel{?}{=} g(z, z)\} \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), g(a, y) \stackrel{?}{=} g(y, a), g(x, h(y)) = g(z, z) \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), a \stackrel{?}{=} y, y \stackrel{?}{=} a, g(x, h(y)) \stackrel{?}{=} g(z, z) \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), y \stackrel{?}{=} a, g(x, h(y)) \stackrel{?}{=} g(z, z) \qquad \text{(Orientierung)}$$

$$\to x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, g(x, h(a)) \stackrel{?}{=} g(z, z) \qquad \text{(Anwendung, y)}$$

$$\to x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, x \stackrel{?}{=} z, h(a) \stackrel{?}{=} z \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, x \stackrel{?}{=} z, z \stackrel{?}{=} h(a) \qquad \text{(Orientierung)}$$

$$\to x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, z \stackrel{?}{=} h(a) \qquad (\text{Anwendung},z)$$

Unifizierte Terme: $k(f(h(a), g(a, a)), g(h(a), h(a)))$.

Satz 5.21. *Der Unifikationsalgorithmus U1 terminiert, ist korrekt und vollständig.*

Beweis. Seien s und t die zu unifizierenden Terme. Sei $W := FV(s, t)$.

Korrektheit: Wir müssen zeigen, daß die berechnete Substitution auch wirklich ein Unifikator für die beiden Eingabeterme ist. Die Idee dazu ist folgende: Man zeige, daß für jede Transformationsregel gilt: wenn die Gleichungsmenge Γ' nach einer Regelanwendung unifizierbar ist, dann ist auch die entsprechende Menge vor der Regelanwendung unifizierbar. Per Induktion nach der Anzahl der Regelanwendungen ergibt sich dann: Wenn die Lösungsgleichungsmenge unifizierbar ist, dann sind auch die Ausgangsterme unifizierbar. Man muß jetzt jede Unifikationsregel daraufhin untersuchen.
Wir machen das am Beispiel der Dekompositionsregel:
Angenommen, $\Gamma' = \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup \Gamma$ ist unifizierbar, d.h. es existiert eine Substitution $\tau \in U(\{s_1 \stackrel{?}{=} t_1, \ldots s_n \stackrel{?}{=} t_n\} \cup \Gamma)$. Dann gilt selbstverständlich $\tau f(s_1, \ldots, s_n) \stackrel{?}{=} \tau f(t_1, \ldots, t_n)$, d.h. das Gleichungssystem vorher ist unifizierbar. Für die anderen Regeln ist die Argumentation analog.

Terminierung Um zu zeigen, daß der Unifikationsalgorithmus immer terminiert, müssen wir ein (fundiertes) Maß für Gleichungssysteme finden, das nach jeder Regelanwendung kleiner wird. Ein Maß ist das Tupel (Anzahl der ungelösten Variablen, Anzahl der Symbole, Anzahl der falsch orientierten Gleichungen) mit lexikographischer Ordnung. (Eine Variable x heißt gelöst wenn sie nur noch in der Gleichung $x \stackrel{?}{=} t$, aber x nicht in t vorkommt.). Dieses Maß wird nach jeder Regelanwendung kleiner: Wenn die Dekomposition oder Tautologieregel feuert, wird die Anzahl der Symbole kleiner, wenn die Anwendungsregel feuert, wird die Anzahl der ungelösten Variablen kleiner, und die Orientierungsregel verkleinert die dritte Komponente und läßt die anderen fest.

Vollständigkeit: Wir zeigen, daß alle Regeln die Menge der Unifikatoren nicht verändert. Da die Korrektheit schon gezeigt ist, d.h. die Menge vergrößert sich nicht, müssen wir jetzt nur noch zeigen, daß die Menge der Unifikatoren sich nicht verkleinert. Sei τ eine Substitution mit $\tau \in U(\Gamma)$ und sei $\Gamma \to \Gamma'$ transformiert worden. Wenn die entsprechende Regel die Dekomposition, Tautologie oder die Orientierung war, dann sieht man leicht ein, daß $\tau \in U(\Gamma')$. Betrachtung wir die Regel (Anwendung). Dann gilt $\tau x = \tau t$. Hieraus folgt, daß $\tau = \tau\{x \mapsto t\}$ ist, denn für x gilt: $\tau(x) = \tau(t)$ und $\tau\{x \mapsto t\}(x) = \tau(t)$. Für andere Variablen $y \neq x$ gilt offenbar $\tau\{x \mapsto t\}(y) = \tau(y)$. Folglich gilt auch hier $\tau \in U(\Gamma')$.

□

Lemma 5.22. *Der Unifikationsalgorithmus U1 terminiert entweder mit einem Unifikator in gelöster Form, oder er bricht ab mit "nicht unifizierbar".*

Beweis. nach 5.21 terminiert U1 immer. Es gibt folgende Möglichkeiten: entweder bricht er ab mit "nicht unifizierbar" oder es ist keine Regel mehr anwendbar. Wenn keine Regel mehr anwendbar ist, dann gibt es nur noch Gleichungen $x \stackrel{?}{=} t$, Dekomposition ist nicht mehr anwendbar, die Gleichungen sind richtig orientiert und x kommt nur einmal in Γ vor, insbesondere nicht in t. □

Lemma 5.23. *Sei $\Gamma = \{x_1 \stackrel{?}{=} t_1, \ldots, x_k \stackrel{?}{=} t_k\}$ ein Gleichungssystem in gelöster Form. Dann ist $\sigma := \{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ ein allgemeinster Unifikator.*

Beweis. Offenbar ist σ ein Unifikator. Zum Nachweis der Allgemeinstheit sei τ ein Unifikator von Γ. Dann gilt $\tau\sigma = \tau$: $\tau\sigma x_i = \tau t_i = \tau x_i$ für $i = 1, \ldots, n$ und für eine Variable $x \notin \{x_1, \ldots, x_n\}$ gilt $\tau\sigma x = \tau x$. Also ist σ allgemeiner als τ. □

Satz 5.24. *Für jedes unifizierbare Gleichungssystem Γ gibt es bis auf Umbenennung von Variablen genau einen allgemeinsten Unifikator und U1 berechnet einen solchen.*

5.3 Komplexität des Unifikationsalgorithmus

Aussage 5.25. *Der Unifikationsalgorithmus U1 hat exponentielle Laufzeit in der Anzahl der Symbole der zu unifizierenden Terme:*

Beispiel 5.26. Zu unifizieren sei:

$$f(x_2, x_3, x_4, \ldots, x_n) \stackrel{?}{=} f(g(x_1, x_1), g(x_2, x_2), g(x_3, x_3), \ldots, g(x_{n-1}, x_{n-1}))$$

Der Unifikationsalgorithmus arbeitet dann folgendermaßen:

$$f(x_2, x_3, x_4, \ldots, x_n) \stackrel{?}{=} f(g(x_1, x_1), g(x_2, x_2), g(x_3, x_3), \ldots, g(x_n - 1, x_n - 1))$$

$$\to x_2 \stackrel{?}{=} g(x_1, x_1), x_3 \stackrel{?}{=} g(x_2, x_2), x_4 \stackrel{?}{=} g(x_3, x_3), \ldots, x_n \stackrel{?}{=} g(x_{n-1}, x_{n-1}) \qquad \text{(Dekomposition)}$$

$$\to x_2 \stackrel{?}{=} g(x_1, x_1), x_3 \stackrel{?}{=} g(g(x_1, x_1), g(x_1, x_1)), \ldots, x_n \stackrel{?}{=} g(x_{n-1}, x_{n-1}) \qquad \text{(Anwendung, } x_2\text{)}$$

$$\to x_2 \stackrel{?}{=} g(x_1, x_1), x_3 \stackrel{?}{=} g(g(x_1, x_1), g(x_1, x_1)),$$

$$x_4 \stackrel{?}{=} g(g(g(x_1, x_1), g(x_1, x_1)), g(g(x_1, x_1), g(x_1, x_1))), \ldots, x_n \stackrel{?}{=} g(x_{n-1}, x_{n-1}) \text{ (Anwendung, } x_3\text{)}$$

Da die Terme exponentiell anwachsen, benötigt man in diesem Beispiel exponentiell viel Platz.

Eine Verbesserung ist die Verwendung von Sharing, d.h.gerichteten Graphen statt Bäumen). Danach wird der Platzbedarf stark verbessert (linear), allerdings benötigt der (naive) occurs-check exponentiellen Aufwand. Auch diesen kann man verbessern durch Verwendung von optimierten Graph-algorithmen, so daß man nach Optimierung einen im schlimmsten Fall quadratischen Algorithmus erhält.

Es gibt jedoch Unifikationsalgorithmen, die durch geschickte Repräsentation der Terme eine lineare Komplexität haben [PM68]. Allerdings sind für praktische Zwecke die Varianten des Algorithmus von Martelli und Montanari geeigneter [MM82].

5.4 Der allgemeine Resolutionskalkül

Wir können nun zusammenfassend sagen, wie der Resolutionskalkül auf Klauselmengen arbeitet:

Definition 5.27. *Der* Resolutionskalkül *ist ein Kalkül, der Klauselmengen in Klauselmengen transformiert mit den folgenden beiden Regeln:*

1. *Sei C eine Menge von Klauseln. Sei R eine Resolvente von zwei (nicht notwendig verschiedenen) Klauseln in CS, wobei der zugehörige Unifikator ein allgemeinster ist. Dann bilde $C' := C \cup \{R\}$*
2. *Sei C eine Menge von Klauseln. Sei F ein Faktor einer der Klauseln in CS, wobei der zugehörige Unifikator ein allgemeinster ist. Dann bilde $C' := C \cup \{F\}$*

5.5 Schritte zum Vollständigkeitsbeweis der allgemeinen Resolution

Definition 5.28. *Herbrand Interpretation: Sei C eine Klauselmenge über der Signatur Σ und V. Eine* Herbrand Interpretation *ist eine Interpretation $(I_V, T(\Sigma, \emptyset), P)$ wobei*

– $T(\Sigma, \emptyset)$ *die Grundtermalgebra ist (ohne Variablensymbole) (auch* Herbrand Universum *genannt) und*
– I_V *eine Variablenbelegung in* $T(\Sigma, \emptyset)$*, d.h. eine Abbildung auf Grundterme*
– P *für jedes Prädikatensymbol eine Relation auf Grundtermen festlegt.*

(Für jedes Prädikatensymbol kann man die Relation dadurch festlegen, daß man eine Menge von Atomen angibt, die als wahr in der Interpretation angenommen werden sollen.)

Bemerkung: Für eine feste Klauselmenge unterscheiden sich die Herbrand-Interpretationen nur durch die Interpretation der Variablen und Prädikatensymbole.

Satz 5.29. *(Herbrand Theorem) Für eine unerfüllbare Menge C von Klauseln gibt es eine endliche unerfüllbare Menge von Grundinstanzen*

Beweis. 1.) Die Menge C_{gr} aller Grundinstanzen von C ist unerfüllbar. Wir nehmen an, daß die Menge C_{gr} aller Grundinstanzen von C erfüllbar ist, von einer Interpretation I. Wir konstruieren eine Herbrandinterpretation I_H für C_{gr} wie folgt: In jeder Grundklausel wird mindestens ein Literal von I wahr gemacht. All diese Literale zusammen ergeben dann genau eine (Herbrand) Interpretation der Prädikatensymbole, die in C vorkommen.

Um zum Widerspruch zu kommen, müssen wir jetzt zeigen, daß diese Herbrandinterpretation auch die ursprüngliche Klauselmenge C erfüllt. Das heißt, für jede Klausel D und für jede Belegung der Variablen $x_1, \ldots, x_n$ in der Klausel mit Grundtermen muß gelten, daß $I_H[t_1/x_1, \ldots, t_n/x_n] \models D$. Es ist jedoch $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}C$ gerade eine der Grundklauseln, die von I_H erfüllt wird. Da bei Herbrand-Interpretationen Variablenbelegungen dasselbe sind wie Substitutionen, gilt: $I_H[t_1/x_1, \ldots, t_n/x_n] \models C$. Also wird die Klauselmenge C von I_H erfüllt, und das ist der gesuchte Widerspruch.
2.) Es gibt eine endliche unerfüllbare Teilmenge von C_{gr}:

Annahme: jede endliche Teilmenge von C_{gr}: erfüllbar ist. Wir konstruieren damit eine Interpretation von von C_gr. Sei $A_1, A_2, \ldots$ eine Aufzählung der Atome einer Herbrandbasis (alle Grundatome). Sei $\mathcal{B} := \bigcup \mathcal{B}_n$ wobei $\mathcal{B}_n := \{B \mid B : \{A_1, A_2, \ldots, A_n\} \to \{0, 1\}\}$. Das ist die Menge aller möglichen (Teil-) Belegungen von Atomen. Dies ergibt einen Baum mit den Belegungen als Knoten; von B nach B' geht genau dann eine gerichtete Kante, wenn $B \in \mathcal{B}_n$, $B' \in \mathcal{B}_{n+1}$ und B und B' auf $\{A_1, A_2, \ldots, A_n\}$ übereinstimmen. Ein Knoten $B \in \mathcal{B}_n$ in diesem Baum heißt *abgeschlossen*, wenn es eine Klausel in C_{gr} gibt, so daß alle in C_{gr} vorkommenden Atome in $\{A_1, A_2, \ldots, A_n\}$ sind und B diese Klausel zu falsch evaluiert. Offenbar ist auch jeder Nachfolgeknoten eines abgeschlossenen Knotens wieder abgeschlossen. Für jedes n gibt es einen nicht abgeschlossenen Knoten in $\mathcal{B}_n$, denn anderenfalls gibt es für jeden Knoten in $\mathcal{B}_n$ eine Klausel in C_{gr}, die zu falsch evaluiert wird. Dies sind endlich viele Klauseln, deren Konjunktion könnte dann von keiner Interpretation erfüllt werden, was ein Widerspruch zur obigen Annahme ist. Der Teilbaum der nicht abgeschlossenen Knoten ist ein unendlicher Baum mit endlicher Verzweigungsrate. Nach König's Lemma gibt es einen unendlichen Pfad in diesem Baum. Dieser unendliche Pfad stellt gerade eine Interpretation von C_{gr} dar. Dies ist ein Widerspruch zur Annahme, daß C_{gr} unerfüllbar ist. Damit haben wir gezeigt, daß es eine *endliche* unerfüllbare Teilmenge von C_{gr} gibt. □

Das hier nicht bewiesene Kompaktheitstheorem für PL_1 sagt aus, daß jede unerfüllbare unendliche Menge von Aussagen eine endliche, unerfüllbare Teilmenge hat. Unter Ausnutzung dieses Satzes kann man im zweiten Teil des obigen Beweises sofort schließen, daß es eine unerfüllbare endliche Teilmenge von C_{gr} geben muß.

Das Herbrand Theorem können wir jetzt für folgende Argumentation benutzen: Für eine unerfüllbare Klauselmenge gibt es offensichtlich eine endliche Menge von Grundinstanzen, so daß diese Grundklauselmenge unerfüllbar ist. Diese Grundklauselmenge können wir durch Resolution ohne Unifikation widerlegen, das sagte gerade das Theorem 5.5. Was wir jetzt noch brauchen ist die Möglichkeit, diese Widerlegung auf der variablenfreien Ebene zu einer Widerlegung auf der Ebene der Originalklauseln zu "liften", d.h. wir müssen zeigen, daß es für jeden Grundresolutionsschritt eine entsprechende allgemeine Resolution gibt, so daß die Grundresolvente eine Instanz der allgemeinen Resolvente ist. Das leistet das sogenannte Lifting Lemma

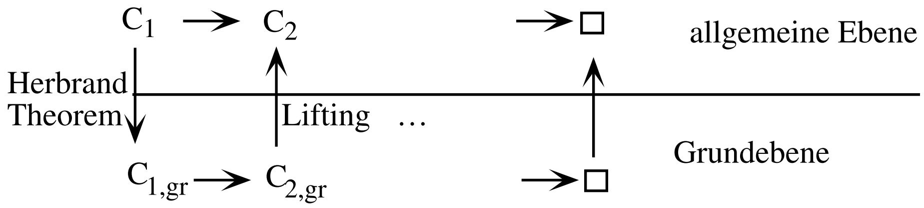
Lemma 5.30. Lifting Lemma

Seien C und D zwei beliebige Klauseln ohne gemeinsame Variablen, C_{gr} und D_{gr} zwei Grundinstanzen davon. Für jede Resolvente R_{gr} zwischen C_{gr} und D_{gr} gibt es eine entsprechende Resolvente R von Faktoren (siehe Definition. 5.13) von C und D, so daß R_{gr} eine Grundinstanz von R ist.

Beweis. siehe einschlägige Literatur. □

Satz 5.31. *Vollständigkeit der Resolution und Faktorisierung mit allgemeinsten Unifikatoren: Für jede unerfüllbare Klauselmenge gibt es einen Widerlegungsbeweis mittels Resolution und Faktorisierung.*

Beweis. Wir sammeln einfach alle bisherigen Ergebnisse auf: Nach dem Herbrand Theorem (5.29) gibt es eine endliche unerfüllbare Grundklauselmenge, für die man nach Theorem 5.5 eine Resolutionswiderlegung findet. Durch Induktion nach der Anzahl der Grundresolutionen bekommt man jetzt mittels des Lifting Lemmas, daß es für jede dieser Grundresolutionen eine entsprechende Sequenz von Faktorisierungen und Resolution mit allgemeinsten Unifikatoren gibt, so daß die Resolvente allgemeiner ist als die Grundresolvente. Für die allerletzte Grundresolvente, die leere Klausel, ist die einzige allgemeinere Klausel ebenfalls die leere Klausel. Daher wird auf der allgemeineren Ebene ebenfalls die leere Klausel produziert. □



5.6 Ein Beispiel für Resolutionsbeweise

Wir formulieren in PL_1 Aussagen über eine andere Logik, nennen wir sie MINI. Dies ist eine Implementierung einer Variante des Hilbertkalküls für Aussagenlogik. Die Syntax dieser Logik MINI lasse nur nullstellige Prädikatensymbole und einen Implikationsjunktor IMP zu. Wir betrachten einen Kalkül für MINI mit zwei logischen Axiomen

$$X\ IMP\ (Y\ IMP\ X)$$

$$(X\ IMP\ (Y\ IMP\ Z))\ IMP\ ((X\ IMP\ Y)\ IMP\ (X\ IMP\ Z))$$

und der einzigen Schlußregel Modus Ponens

$$\frac{X, \quad (X\ IMP\ Y)}{Y}$$

D.h. der Kalkül arbeitet auf Mengen von Formeln, und fügt entsprechend der Schlußregel Modus ponens neue Formeln hinzu. Allerdings ist diese Schlußregel mit Instanziierung verbunden, so daß diese korrekter lauten sollte:

$$\frac{X_1, \quad (X_2\ IMP\ Y)}{\sigma(Y)} \quad \sigma \text{ ist allgemeinster Unifikator von } X_1, X_2$$

Dabei stehen X_1, X_2, Y, Z jeweils für beliebige MINI-Formeln. Wir wollen zeigen, daß im MINI-Kalkül die Formel $(X\ IMP\ X)$ für beliebige MINI-Formeln X ableitbar ist.

Um die Aussagen über MINI in PL_1 zu formulieren, benutzen wir ein PL_1-Prädikatensymbol "ableitbar" und kodieren den Junktor als ein PL_1-Funktionssymbol "imp" (in Präfixschreibweise). Dadurch werden MINI-Formeln als PL_1-Terme dargestellt. Der MINI-Kalkül läßt sich dann folgendermaßen durch PL_1-Formeln beschreiben:

F1: $\forall x, y: \quad ableitbar(imp(x, imp(y, x)))$

F2: $\forall x, y, z: ableitbar(imp(imp(x, imp(y, z)), imp(imp(x, y), imp(x, z))))$

F3: $\forall x, y: \quad ableitbar(x) \wedge ableitbar(imp(x, y)) \Rightarrow ableitbar(y)$

Wir wollen zeigen, daß daraus folgende Formel folgt:

$$B : \forall x : ableitbar(imp(x,x))$$

Die Klauselform von $F1 \ \wedge \ F2 \ \wedge \ F3 \ \wedge \ \neg B$ ist eine Menge von vier Klauseln, wobei die Variable in $\neg B$ durch eine nullstellige Skolemfunktion, also eine Skolemkonstante c ersetzt ist:

$$\text{C1} = ableitbar(imp(x_1, imp(y_1, x_1)))$$

$$\text{C2} = ableitbar(imp(imp(x_2, imp(y_2, z_2)), imp(imp(x_2, y_2), imp(x_2, z_2))))$$

$$\text{C3} = \neg ableitbar(x_3), \neg ableitbar(imp(x_3, y_3)), ableitbar(y_3)$$

$$\text{B} \ \ = \neg ableitbar(imp(c, c))$$

Im folgenden bezeichne C, n das n-te Literal einer Klausel C, das Kürzel $C, n + D, m \vdash R$ steht für eine Anwendung der Resolutionsregel mit den entsprechenden Resolutionsliteralen der Elternklauseln C und D und der Resolvente R. Die Variablen jeder Resolvente werden systematisch umbenannt, um Namenskonflikte zu vermeiden.Wir führen folgende Resolutionsableitung durch:

$C1 + C3, 1\{x_3 \mapsto imp(x_1, imp(y_1, x_1))\}$

$\vdash \; R1' = \neg ableitbar(imp(imp(x_1, imp(y1_, x_1)), y_3)), ableitbar(y3)$

$\rightarrow R1 \; = \neg ableitbar(imp(imp(x_4, imp(y_4, x_4)), z_4)), ableitbar(z4)$

$C2 + R1, 1\{z_4 \mapsto imp(imp(x_4, y_4), imp(x_4, x_4)), x_2 \mapsto x_4, y_2 \mapsto y_4, z_2 \mapsto x_4\}$

$\vdash \; R2' = ableitbar(imp(imp(x_4, y_4), imp(x_4, x_4)))$

$\rightarrow R2 \; = ableitbar(imp(imp(x_5, y_5), imp(x_5, x_5)))$

$R2 + C3, 2\{x_3 \mapsto imp(x_5, y_5), y_3 \mapsto imp(x_5, x_5)\}$

$\vdash \; R3' = \neg ableitbar(imp(x_5, y_5)), ableitbar(imp(x_5, x_5))$

$\rightarrow R3 \; = \neg ableitbar(imp(x_6, y_6)), ableitbar(imp(x_6, x_6))$

$C1 + R3, 1\{x_6 \mapsto x_1, y_6 \mapsto imp(y_1, x_1)\}$

$\vdash \; R4' = ableitbar(imp(x_1, x_1))$

$\rightarrow R4 \; = ableitbar(imp(x_7, x_7))$

$B + R4 \quad \vdash \quad R5 = \Box$