12 Unifikation unter Gleichheitstheorien

Wie bereits in dem Kapitel über die Gleichheitsbehandlung ausgeführt wurde, bereiten Formeln, in denen die Gleichheit vorkommt, ziemliche Schwierigkeiten beim automatischen Beweisen, trotz Paramodulationsregel. Besonders unangenehm verhalten sich dabei zum Beispiel Formeln, die die Kommutativität gewisser Funktionssymbole definieren, d.h. wenn etwa $\forall x, y : g(x, y) = g(y, x)$ gilt. Diese Kommutativitätsformel führt unter Umständen zu einem ständigen Vertauschen der Argumente von Termen mit dem kommutativen Funktionssymbol. Daher wurde schon relativ früh versucht, solche Gleichungsformeln aus den Formelmengen zu entfernen und durch entsprechend veränderte Schlußregeln zu ersetzen. G. Plotkin schlug vor, die Resolutionsregel dahingehend zu ändern, daß die Unifikation durch eine die herausgenommenen Gleichungsformeln berücksichtigende Unifikation ersetzt wird. Er gab auch die wesentlichen Kriterien an, unter denen das geschehen kann, ohne daß die Vollständigkeit des Kalküls verloren geht. Vorausgesetzt, das Gleichheitsprädikat kommt nur in unären Klauseln vor, d.h. die Klauselmenge enthält endlich viele Klauseln $l_1 = r_1, \ldots, l_n = r_n$, und es tritt sonst in keiner weiteren Klausel auf, dann darf die Unifikation durch eine sogenannte *Theorieunifikation* ersetzt werden, die die Gleichungsaxiome $l_1 = r_1, \ldots, l_n = r_n$ berücksichtigt.

Der Resolutionskalkül RF wird dann folgendermaßen abgeändert:

statt auf $C \cup E$

wird auf C Resolution und Faktorisierung mit Unifikation bzgl. E verwendet.

Voraussetzung: C enthält keine Gleichheitssymbole, E besteht nur aus Unit-Klauseln, die Gleichungen sind. Der E-Unifikationsalgorithmus liefert i.a. eine Lösungsmenge von Unifikatoren, die alle auszuprobieren sind.

Es gilt: Wenn der E-Unifikationsalgorithmus eine vollständige Menge von Unifikatoren berechnet, dann ist der Kalkül widerlegungsvollständig.

Hierbei muß die Lösungsmenge (endlich oder unendlich) und rekursiv aufzählbar sein.

Theorieunifikationsalgorithmen

Beispiel 12.1. Die beiden Terme $g(x, y)$ und $g(a, b)$ haben den üblichen Unifikator $\theta_1 = \{x \mapsto a, y \mapsto b\}$. Unter der Annahme, daß g kommutativ ist, ergibt sich aber noch ein zweiter Unifikator, nämlich $\theta_1 = \{x \mapsto b, y \mapsto a\}$. Offensichtlich sind beide Unifikatoren unabhängig voneinander, d.h. es gibt in diesem Fall mehr als einen allgemeinsten Unifikator. Die Gleichung $k(f(x, g(a, y)), g(x, h(y))) \stackrel{?}{=} k(f(h(y), g(y, a)), g(z, z))$ hat eine Lösung $\sigma_1 = \{x \mapsto h(a), y \mapsto a, z \mapsto h(a)\}$, die z.B. mit dem Unifikationsalgorithmus U1 (Definition 5.19) berechnet werden kann. Unter der Annahme, daß g kommutativ ist, ist $\sigma_2 = \{x \mapsto h(y), z \mapsto h(y)\}$ ebenfalls ein Unifikator, der in diesem Fall offensichtlich sogar allgemeiner ist als der oben berechnete. Man erhält den obigen Unifikator durch weitere Instantiierung, indem man y durch a substituiert. Benutzt man σ_2 als Unifikator (und zusätzlich $y' \mapsto y$ dann kann man zum Beispiel folgende (totale) Theorieresolution durchführen:

$$\frac{\begin{array}{l} P(y, k(f(x, g(a, y)), g(x, h(y)))), Q(x, y) \\ \neg P(y', k(f(h(y'), g(y', a)), g(z, z)))R(y', z) \end{array}}{Q(h(y), y), R(y, h(y))}$$

Die Theorie, die dabei benutzt wurde ist die Kommutativität von g, d.h. $\forall x, y : g(x, y) = g(y, x)$.

Wir definieren jetzt einen Unfikationsalgorithmus für kommutativ deklarierte Funktionssymbole. Er ist eine leichte Abwandlung von U1.

Definition 12.2. *Unifikationsalgorithmus U_C: Eingabe:*

1. *zwei Terme oder Atome s und t,*
2. *sowie eine Liste F_C von zweistelligen Funktionssymbolen, die als kommutativ deklariert sind.*

Ausgabe: "nicht unifizierbar" oder eine vollständige Menge von Unifikatoren.
Zustände, auf denen der Algorithmus operiert: Eine Menge Γ von Gleichungen.
Initialzustand: $\Gamma_0 = \{s \stackrel{?}{=} t\}$.
Unifikationsregeln:

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

$$\frac{g(s_1, s_2) \stackrel{?}{=} g(t_1, t_2), \Gamma}{s_1 \stackrel{?}{=} t_2 \;,\; s_2 \stackrel{?}{=} t_1, \Gamma} \qquad \textit{(Vertauschung)}$$

falls g in F_C enthalten ist, d.h. g ist kommutativ.

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

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

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

Abbruchbedingungen:

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

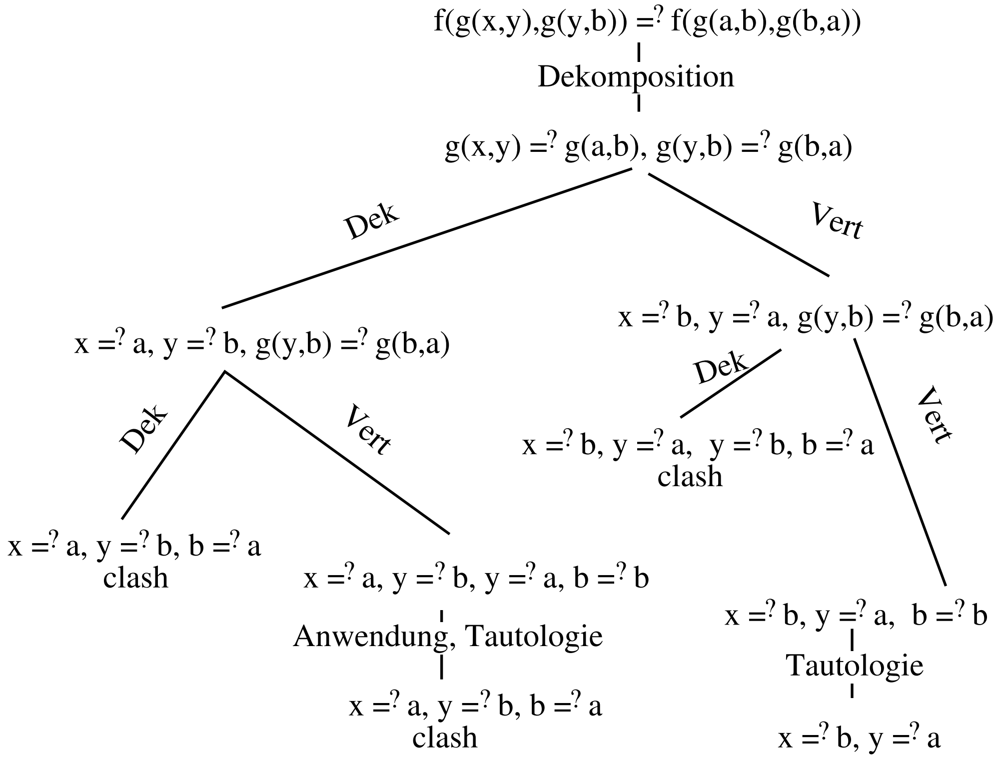
$$\frac{x \stackrel{?}{=} t, \Gamma}{Fail} \qquad \text{wenn } x \in FV(t) \text{ vorkommt} \qquad \textit{(occurs check Fehler)}$$

Steuerung: *Erzeuge einen Baum folgendermaßen: Der Wurzelknoten ist $\Gamma := \Gamma_0$. Für jeden Knoten, der mit einem Gleichungssystem Γ markiert ist: Wähle eine Gleichung $s \stackrel{?}{=} t$ in Γ aus, auf die mindestens eine Regel anwendbar ist und erzeuge die Nachfolgeknoten (maximal zwei) dadurch, daß alle anwendbaren Regeln auf $s \stackrel{?}{=} t$ angewendet werden. Auf diese Weise werden so lange neue Äste im Baum erzeugt, bis entweder eine Abbruchbedingung erfüllt ist oder keine Regel mehr anwendbar ist. Die Gleichungsmenge in den Blattknoten, auf die keine Abbruchbedingung zutrifft hat die Form $\{x_1 \stackrel{?}{=} t_1, \ldots, x_k \stackrel{?}{=} t_k\}$. Die daraus generierten Substitutionen $\{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ werden aufgesammelt und als Ergebnis zurückgeliefert.*

Beispiel 12.3. $g(x,y) \stackrel{?}{=} g(a,b)$ kann auf zwei Weisen dekomponiert werden, wenn $g \in F_C$ angenommen wird. Dies ergibt sofort zwei Systeme von Gleichungen:

- $x \stackrel{?}{=} a, y \stackrel{?}{=} b$.
- $x \stackrel{?}{=} b, y \stackrel{?}{=} b$.

Ein weiteres Beispiel ist folgender Berechnungsbaum:



Es gibt also nur eine Lösung in diesem Fall: $\{x \mapsto b, y \mapsto a\}$.

Wie man jetzt leicht nachvollziehen kann, würde der Algorithmus U_C für das Beispiel aus 12.1 $k(f(x, g(a, y)), g(x, h(y))) \stackrel{?}{=} k(f(h(y), g(y, a)), g(z, z))$ beide Lösungen $\sigma_1 = \{x \mapsto h(a), y \mapsto a, z \mapsto h(a)\}$ und $\sigma_2 = \{x \mapsto h(y), z \mapsto h(y)\}$ ausrechnen, wobei σ_2 allgemeiner ist als σ_1. Das heißt der Algorithmus ist *nicht minimal*, er berechnet eventuell mehr Lösungen als notwendig. Da nur endlich viele Lösungen berechnet werden, kann man aber die überflüssigen im nachhinein ermitteln (mittels Unifikation der Termpaare $\sigma_1(x_i)$ und $\sigma_2(x_i)$ unter Kommutativität, wobei man auf einer Seite die Variablen als konstant ansieht. Danach kann man die nicht allgemeinsten Unifikatoren eliminieren. Für Kommutativität gibt es also im allgemeinen mehr als einen allgemeinsten Unifikator, aber immer noch endlich viele, wie man sich leicht überlegen kann und wie wir auch unten zeigen werden.

Es kann aber noch unangenehmer sein, wenn etwa eine Funktion f assoziativ ist, wenn also $\forall x, y, z : f(x, f(y, z)) = f(f(x, y), z)$ gilt. In diesem Fall hat etwa das Gleichungssystem $\{f(x, a) \stackrel{?}{=} f(a, x)\}$ unendlich viele unabhängige Lösungen:

$$\{x \mapsto f(a, a)\}, \{x \mapsto f(a, f(a, a))\}, \{x \mapsto f(a, f(a, f(a, a)))\}, \ldots$$

Alle Lösungen sind im Sinne der Assoziativität von f gleich zu einer dieser Lösungen, d.h. der für x substituierte Term ist bis auf Umklammerung einer der hier angegebenen. Allerdings erspart man sich auch in diesem unangenehmen Fall gegenüber der Resolution (bzw. Paramodulation) ohne Theorieunifikation etwa das ständige Umklammern von Termen mit assoziativem Funktionssymbol.

Das folgende Beispiel zeigt, daß es neben den Fällen, daß es maximal einen, endlich viele oder unendlich viele allgemeinste Unifikatoren gibt, auch noch den Fall gibt, daß es für eine Theorie gar keine allgemeinsten Unifikatoren mehr gibt:

Beispiel 12.4. Es ist möglich, daß es in speziellen Fällen keine allgemeinsten Unifikatoren gibt.

Sei $L := \{\mathtt{append}(x, \mathtt{nil}) = x, \mathtt{first}(\mathtt{append}(x, y)) = \mathtt{first}(x), \mathtt{first}(\mathtt{nil}) = \mathtt{nil}\}$ die Menge der Axiome einer Theorie (eine Theorie für), wobei es eine Konstante `nil`, ein einstelliges Symbol `first` und ein zweistelliges Funktionssymbol `append` gibt. Dann hat das L-Unifikationsproblem $\{\mathtt{first}(x) \stackrel{?}{=} \mathtt{nil}\}$ die vollständige Menge von Unifikatoren:

$$cU_L(\mathtt{first}(x) \stackrel{?}{=} \mathtt{nil}) = \{\sigma_n \mid n \geq 0\}$$

mit

$$\sigma_0 = \{x \mapsto \texttt{nil}\}$$

$$\sigma_1 = \{x \mapsto \texttt{append}(\texttt{nil}, x1)\}$$

$$\sigma_2 = \{x \mapsto \texttt{append}(\texttt{append}(\texttt{nil}, x_1), x_2)\}$$

$$\ldots$$

$$\sigma_n = \{x \mapsto \texttt{append}(\texttt{append}(\ldots(\texttt{append}(\texttt{nil}, x_1), x_2), \ldots, x_n)\}$$

$$\ldots$$

Aber jeder Unifikator σ_n ist allgemeiner als der Vorgänger σ_{n-1}: Man substituiere x_n durch `nil` und wende das erste Axiom aus L an, dann erhält man den Vorgänger. Man hat hier also eine Kette von immer allgemeiner werdenden Unifikatoren. Diese ist auch vollständig, aber kein Unifikator alleine ist bereits vollständig, also gibt es keine allgemeinsten.

Da die Verhältnisse offensichtlich ziemlich kompliziert sind, ist es nötig, Theorie-unifikation etwas formaler zu untersuchen und zu definieren. Wir haben jetzt den Fall, daß zwei Terme, z.B. $g(a, b)$ und $g(b, a)$ zwar syntaktisch verschieden sind, aber unter der Annahme der Kommutativität von g das gleiche bedeuten. D.h. es gibt eine Relation zwischen $g(a, b)$ und $g(b, a)$, die wir mit $g(a, b) =_C g(b, a)$ (C für Commutativity) bezeichnen wollen. (Für die beiden Terme $f(g(a, b))$ und $f(g(b, a))$ soll dann auch gelten $f(g(a, b)) =_C f(g(b, a))$, d.h. die Relation $=_C$ zieht sich hoch auf zusammengesetzte Terme.) Ähnliches gilt für Substitutionen: $\{x \mapsto g(a, b)\}$ und $\{x \mapsto g(b, a)\}$ sehen zwar auch verschieden aus, durch die Kommutativität von g bedeuten sie aber das gleiche, d.h. $\{x \mapsto g(a, b)\} =_C \{x \mapsto g(b, a)\}$.

Die formalen Definitionen dafür sind:

Definition 12.5. *Eine Menge von Axiomen $E := \{l_1 = r_1, \ldots, l_n = r_n\}$ von unären Klauseln mit dem Gleichheitsprädikat (Gleichheitsaxiome) induziert eine Gleichheitstheorie auf der Termalgebra $T(\Sigma, V)$ über einer Signatur Σ, die mindestens die Symbole der Gleichheitsaxiome in E enthält. Diese Gleichheitstheorie ist die kleinste Äquivalenzrelation $=_E$ auf $T(\Sigma, V)$ die alle Termpaare (l_i, r_i) für $l_i = r_i$ aus E enthält, und abgeschlossen ist gegenüber der Termbildung und der Substitution von Variablen:*

1. *Wenn $s_1 =_E t_1, \ldots, s_n =_E t_n$ und f ist ein n-stelliges Funktionssymbol, dann $f(s_1, \ldots, s_n) =_E f(t_1, \ldots, t_n)$*
2. *Wenn $s =_E t$ und σ ist eine Substitution, dann $\sigma s =_E \sigma t$.*

Eine solche Relation heißt auch Kongruenzrelation.

Zwei Terme sind genau dann in der Relation $=_E$, wenn man ihre Gleichheit aus den Axiomen E ableiten kann mit den folgenden Regeln:

1. $\vdash t = t$
2. $s = t \vdash t = s$
3. $s = r, r = t \vdash s = t$
4. $s_1 = t_1, \ldots, s_n = t_n \vdash f(s_1, \ldots, s_n) =_E f(t_1, \ldots, t_n)$, wobei f ein n-stelliges Funktionssymbol.
5. $s = t \vdash \sigma t = \sigma s$ für eine beliebige Substitution σ.

Die Regeln sind so zu verstehen: Man startet mit E, den Axiomen, und erweitert diese Menge von Gleichungen unter Benutzung der Regeln, d.h. $E \subseteq A_1 \subseteq A_2 \subseteq \ldots$ wobei jeweils eine Gleichung hinzugefügt wurde.

Der Satz von Birkhoff macht eine Aussage über die Vollständigkeit des Ableitungssystems. Hierbei ist $\models$ als $\models_E$ zu lesen.

Satz 12.6. Satz von Birkhoff: *Für jede endliche Menge E von Gleichungen und zwei Terme s, t gilt:*

$$E \models s = t \Leftrightarrow E \vdash s = t$$

Beweis. (siehe Literatur)

Beispiel 12.7. Sei $C := \{g(x, y) = g(y, x)\}$ die Theorie der Kommutativität. C-gleiche Terme sind dann:
$g(a, b) =_C g(b, a)$ oder $f(x, g(a, b), z) =_C f(x, g(b, a), z)$.

Die Gleichheit von zwei Termen kann mit effizienteren Methoden überprüft werden. Die Idee ist, daß zwei Terme s und t gleich sind unter einer Theorie $TH(E)$, wenn sich s mittels Termersetzung in t überführen läßt, wobei die Axiome in E mitverwendet werden dürfen. Ein Darstellung der Termersetzung findet sich in [BN98,Ave95].

Definition 12.8. (Termersetzungskalkül, Rewrite-Kalkül)
Gegeben eine Menge E von Gleichungen (Axiomatisierung einer Theorie). Dann sei für einen Term s:

$$s \to_E s[\pi \to \sigma(r)]$$

genau dann wenn eine Gleichung $l = r$ oder $r = l$ in E existiert, eine Position π und eine Substitution σ mit $\sigma(l) = s|_\pi$. Die Termersetzungsrelation $\longrightarrow_E$ ist der reflexive, transitive Abschluß von $\to_E$.

Als Beispiel betrachten wir den Term $f(x, g(a, b), z)$, wobei g ein kommutatives Symbol ist. Das Axiom der Kommutativität ist $g(x, y) = g(y, x)$. Mit $\sigma = \{x \mapsto a, y \mapsto b\}$ gilt: $\sigma g(x, y) = g(a, b)$. Mit einem Ersetzungsschritt kann der Term $f(x, g(a, b), z)$ in den Term $f(x, g(b, a), z)$ überführt werden. Die Position könnte man hier als [2] notieren. Also gilt: $f(x, g(a, b), z) \longrightarrow_E f(x, g(b, a), z)$. Der Termersetzungskalkül ist vollständig, d.h., er kann (unter einer Menge von Axiomen E) alle gleichen Term ineinander überführen:

Satz 12.9. Korrektheit und Vollständigkeit des Termersetzungskalküls:

1. *Der Termersetzungskalkül ist korrekt.*
2. *Ist E eine Menge von Axiomen und sind s und t zwei Terme mit $s =_E t$, dann gilt $s \longrightarrow_E t$.*

Beweis. **Korrektheit** (Übungsaufgabe).

Vollständigkeit: Zunächst muß man zeigen, daß eine leichte Abwandlung des Birkhoff-kalküls noch vollständig ist: Die Regel 5 wird ersetzt durch

$$s = t \vdash \sigma(s) = \sigma(t)$$

für eine beliebige Substitution σ und falls $s = t$ ein Axiom ist.

Induktion nach der Anzahl der Schritte einer Herleitung von $s =_E t$ mittels des obigen Kalküls. Hierbei betrachten wir nur die Anzahl der Schritte 1) bis 4).

Induktionsbasis: Länge $= 0$. Dann ist $s = t$ ein Axiom oder Instanz eines Axioms. Aus der Definition 12.8 folgt dann $s \to_E t$, insbesondere $s \longrightarrow_E t$.

Induktionsschritt: Hier muß man alle Fälle des Kalküls durchgehen und die Ersetzungsschritte jeweils zu einem ganzen Reduktionsbeweis zusammensetzen.

□

Definition 12.10. (**E-Unifikationsprobleme**) *Ein Unifikationsproblem unter der Gleichheitstheorie E – ein E-*Unifikationsproblem *– ist ein Gleichungssystem $\Gamma = \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}_E$. Eine Lösung von Γ ist eine Substitution σ mit $\sigma s_i =_E \sigma t_i$ (für alle i mit $1 \leq i \leq n$). Sie heißt E-*Unifikator *von Γ. Die Menge der E-Unifikatoren bezeichnen wir mit $U_E(\Gamma)$ oder auch $U_E(s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n)$.*

Im allgemeinen sind die Mengen $U_E(\Gamma)$ unendlich und man möchte sie daher durch eine möglichst kleine repräsentative Teilmenge ersetzen. Bei der üblichen syntaktischen Unifikation - sie entspricht der Unifikation bezüglich der Theorie mit der leeren Axiomatisierung - gelingt dies sogar mit einelementigen Teilmengen, wie Robinson gezeigt hat (siehe auch 5.16).

Definition 12.11. Minimale *und* vollständige *Mengen von E-Unifikatoren - auch Mengen allgemeinster E-Unifikatoren (auch Lösungsbasis genannt) - $\mu U_E(\Gamma)$, sind Mengen von Unifikatoren, die den folgenden Bedingungen genügen:*

(1) $\mu U_E(\Gamma) \subseteq U_E(\Gamma)$ *(Korrektheit)*

(2) Für alle $\delta \in U_E(\Gamma)$ *existiert ein* $\sigma \in \mu U_E(\Gamma)$

mit $\delta x =_E \lambda(\sigma x)$ *(für alle* $x \in V(\Gamma)$*)* *(Vollständigkeit)*

(3) Für alle $\sigma, \tau \in U_E(\Gamma)$ *mit* $\tau x =_E \lambda(\sigma x)$ *(für alle* $x \in V(\Gamma)$*)*

ist $\sigma = \tau$ *(Minimalität)*

Das heißt: (1) alle allgemeinsten E-Unifikatoren lösen Γ*, (2) jeder beliebige E-Unifikator ist Instanz eines allgemeinsten E-Unifikators, und (3) verschiedene allgemeinste E-Unifikatoren sind keine Instanzen voneinander. Unifikatormengen* $cU_E(\Gamma)$ *mit (1) und (2) heißen auch vollständige Mengen von E-Unifikatoren.* $U_E(\Gamma)$ *selbst ist trivialerweise eine vollständige Lösungsmenge.*

Für das Beispiel aus 12.1, $\Gamma = \{k(f(x, g(a, y)), g(x, h(y))) \stackrel{?}{=} k(f(h(y), g(y, a)), g(z, z))\}$ berechnet unser Unifikationsalgorithmus U_C die zwei Lösungen $\{\{x \mapsto h(a), y \mapsto a, z \mapsto h(a)\}, \{x \mapsto h(y), z \mapsto h(y)\}\}$, die zwar eine vollständige Menge bilden, aber nicht minimal ist. Die minimale Menge ist $\{\{x \mapsto h(y), z \mapsto h(y)\}\}$.

Mit den neuen Begriffen können wir jetzt Korrektheit, Terminierung und Vollständigkeit des Unifikationsalgorithmus U_C für Kommutativität beweisen.

Aussage 12.12. *Der Algorithmus* U_C *(Definition 12.2) ist korrekt, terminiert und ist vollständig.*

Beweis. Der Beweis geht jetzt ganz analog zum Beweis für den Algorithmus U1 (Theorem 5.21). Es muß anstelle der syntaktischen Gleichheit die Kongruenzrelation $=_C$ benutzt werden. Wie im Beweis zur Vollständigkeit von U1 zeigt man, daß für die Regeln Tautologie, Anwendung, Orientierung und Dekomposition für nicht C-Symbole keine Lösungen verlorengehen. Im Falle der Dekomposition und Vertauschung für C-Symbole gilt: $U_C(g(s_1, s_2) \stackrel{?}{=} g(t_1, t_2), \Gamma)$ und $U_C(s_1 \stackrel{?}{=} t_1, s_2 \stackrel{?}{=} t_2, \Gamma) \cup U_C(s_1 \stackrel{?}{=} t_2, s_2 \stackrel{?}{=} t_1, \Gamma)$ sind dieselben Mengen. Die Terminierung zeigt man analog zu der Terminierung des Algorithmus U1. Daß durch den Abbruch bei occurs-check keine Lösungen verlorengehen, folgt daraus, daß für Terme s und t mit $s =_C t$ gilt, daß s und t die gleiche Tiefe haben müssen, was mittels des Ersetzungskalküls in 12.8 bewiesen werden kann. □

Da der Algorithmus vollständig ist, terminiert und der aufgebaute Baum nur eine endliche Verzweigungsrate hat, bedeutet das, daß es im Fall der Kommutativität immer maximal endlich viele allgemeinste Unifikatoren gibt.

Jetzt können wir allgemeine Theorieresolution unter Gleichheit definieren.

Definition 12.13. (Theorieresolution unter Gleichheit
Vorausgesetzt man hat ein Verfahren, das für jedes Gleichungssystem die Lösungsmenge $U_E(\Gamma)$ *– oder besser noch eine möglichst kleine repräsentative*

Teilmenge $\mu U_E(\Gamma)$ - berechnet, dann kann man die Resolutionsregel wie folgt abwandeln:

$$\begin{array}{lll} \textit{Klausel1:} & P(s_1,\ldots,s_n), K_1,\ldots,K_m & \\ \textit{Klausel2:} & \underline{\neg P(t_1,\ldots,t_n), L_1,\ldots,L_k} & \sigma \in U_E(s_1 \stackrel{?}{=} t_1,\ldots,s_n \stackrel{?}{=} t_n) \\ \textit{E-Resolvente:} & \sigma K_1,\ldots,\sigma K_m, \sigma L_1,\ldots,\sigma L_k & \end{array}$$

12.1 Eigenschaften von Lösungsmengen

Die folgenden Theoreme zeigen, daß wir hier die "richtige" Form der Repräsentation gewählt haben. Die Eindeutigkeit garantiert, daß verschiedene minimale und vollständige Mengen von E-Unifikatoren eines Gleichungssystems Γ äquivalent im Sinne der folgenden durch wechselseitige Instanzenbildung induzierten *E-Äquivalenz* von Substitutionen sind: $\sigma =_E \tau \Leftrightarrow \exists \lambda, \mu : \sigma x =_E \lambda(\tau x)$ und $\tau x =_E \mu(\sigma x)$ für alle $x \in V(\Gamma)$, d.h. wenn jede eine Instanz der anderen ist. Die Aussage zur Repräsentation zeigt, daß die allgemeinsten Unifikatoren genau die Menge der Unifikatoren repräsentieren: Eine Substitution löst Γ genau dann, wenn sie Instanz eines allgemeinsten Unifikators ist. Die untenstehende Aussage zur Vererbung schließlich besagt, daß man die Lösungsbasen schrittweise berechnen kann: Um ein Gleichungssystem zu lösen, kann man die allgemeinsten Unifikatoren einer der Gleichungen berechnen, diese auf die anderen Gleichungen anwenden und die resultierenden Restsysteme analog lösen.

Aussage 12.14. Eindeutigkeit

1. *Die Lösungsbasen eines Gleichungssystems Γ haben gleiche Kardinalität und unterscheiden sich höchstens um E-Äquivalenz ihrer Elemente.*
2. *Ersetzt man Elemente einer Lösungsbasis von Γ durch E-äquivalente Substitutionen, so erhält man wieder eine Lösungsbasis.*

Aussage 12.15. Repräsentation:

1. *Für jede vollständige Menge $cU_E(\Gamma)$ von E-Unifikatoren gilt: δ löst Γ genau dann, wenn $\delta x =_E \lambda(\sigma x)$ für alle $x \in V(\Gamma)$ mit $\sigma \in cU_E(\Gamma)$ und einer Substitution λ.*
2. *Die Lösungsbasen sind die kleinsten Mengen mit dieser Eigenschaft, d.h. wenn man eine Substitution aus einer Lösungsbasis entfernt, ist sie nicht mehr vollständig.*

Aussage 12.16. Vererbung *Sei $cU_E(\Gamma_1)$ eine vollständige Menge von E-Unifikatoren von Γ_1 und seien $cU_E(\sigma\Gamma_2)$ vollständige Mengen von E-Unifikatoren für Γ_2 für alle Lösungen $\sigma \in cU_E(\Gamma_1)$.*

Dann ist die Menge $\{\tau\sigma \mid \tau \in cU_E(\sigma\Gamma_2), \sigma \in cU_E(\Gamma_1)\}$ eine vollständige Lösungsmenge für das System aus beiden Gleichungen $\Gamma_1 \cup \Gamma_2$

Wie wir bereits gesehen haben, ist es möglich, daß die Menge der allgemeinsten Unifikatoren unendlich groß ist (siehe das Assoziativitätsbeispiel von oben). Es kann sogar sein, daß überhaupt keine Menge allgemeinster E-Unifikatoren existiert (Beispiel 12.4). D.h. die Forderung nach gleichzeitiger Vollständigkeit und Minimalität können kann unverträglich sein.

12.2 Die Unifikationshierarchie

Eine der wichtigsten Aufgaben in der Unifikationstheorie ist es daher, für spezielle Theorien oder gar ganze Klassen von Theorien Existenztheoreme zu finden: Für die Theorie E oder die Klasse $\mathcal{K}$ von Theorien existiert für jedes Unifikationsproblem stets eine Lösungsbasis. Theorien E, für die gewisse Gleichungssysteme keine Lösungsbasis besitzen, heißen auch vom *Unifikationstyp 0* . Man ist darüberhinaus daran interessiert, die Theorien, die nicht Typ 0 sind, zu klassifizieren: Wenn einige Unifikationsprobleme unendliche Lösungsbasen haben, ist die Theorie E vom Typ ∞ oder infinitär. Wenn die Lösungsbasen aller Gleichungssysteme endlich sind, dann ist E vom Typ ω oder *finitär,*. Der beste Fall ist, wenn alle Lösungsbasen einelementig sind; dann ist E vom Typ 1 oder *unitär*. Die letzteren sind insbesondere in der Praxis von Bedeutung, da auch bei einer finitären Theorie die Anzahl der allgemeinsten Unifikatoren beliebig groß werden kann. Eine weitere Klassifikation kann man nach der Komplexität des Entscheidungsproblems durchführen: D.h. in welche Klasse fällt die Frage "Ist Γ bzgl E unifizierbar?" Die Hauptaufgabe in der "praktischen" Unifikationstheorie ist es nun, für spezielle Theorien Unifikationsalgorithmen zu entwickeln. Das sind Algorithmen (oder Verfahren) die als Eingabe ein E-Unifikationsproblem erhalten und eine - möglichst minimale - Menge von E-Unifikatoren bezüglich der speziellen Theorie E zurückliefern.

Als Mindestvoraussetzung für eine Theorie stellt sich damit folgendes

Problem 1: (Unifizierbarkeitsproblem)
Ist die Unifizierbarkeit eines E-Unifikationsproblems in der gegebenen Theorie entscheidbar?

Daß dies im allgemeinen nicht der Fall ist, zeigt die bekannte Unentscheidbarkeit von Hilbert's zehntem Problem: Gibt es ein Verfahren das die Lösbarkeit einer Polynomgleichung über den ganzen Zahlen (Diophantische Gleichung) entscheidet? Antwort: nein. Für Implementierungen ist natürlich dann auch die Frage nach der Anzahl der zurückgelieferten Unifikatoren relevant. Ob überhaupt ein Algorithmus existieren kann oder nur ein Aufzählverfahren, folgt unmittelbar aus der Lage der zu untersuchenden Theorie in der Unifikationshierarchie.

Problem 2: (Hierarchieproblem)
Welchen Unifikationstyp hat die gegebene Theorie?

Schließlich stellt sich die letzte Frage fast von allein: Existiert überhaupt ein Algorithmus, der minimale Mengen von Unifikatoren berechnet bzw. aufzählt?

Hierbei wollen wir noch eine kleine Unterscheidung zwischen finitären und infinitären Theorien treffen. Ein E-Unifikationsalgorithmus heißt *typkonform*, wenn er eine Menge Ψ von E-Unifikatoren berechnet - oder aufzählt - mit:

– Ψ ist immer eine vollständige Menge von E-Unifikatoren

– der Algorithmus terminiert (mit endlichem Ψ), falls eine endliche, vollständige Lösungsmenge existiert

– Ψ ist eine Lösungsbasis, falls keine endliche, vollständige Menge existiert.

Mit anderen Worten: Ein typkonformer Algorithmus berechnet immer eine endliche, vollständige Lösungsmenge, falls eine solche existiert, oder er zählt eine unendliche, aber minimale auf. Natürlich darf die Theorie dann nicht vom Typ 0 sein. Insbesondere berechnen typkonforme Algorithmen für finitäre Theorien immer endliche, vollständige Mengen. Die Aufzählung einer unendlichen, vollständigen Menge wäre nicht sehr sinnvoll, da die gesamte Lösungsmenge bereits vollständig und aufzählbar ist. Ideal sind natürlich minimale Algorithmen, die immer minimale, vollständige Lösungsmengen berechnen oder aufzählen. Damit ergibt sich

> Problem 3: (typkonformer Algorithmus)
> Gibt es für eine gegebene Theorie –die nicht vom Typ 0 ist – einen typkonformen beziehungsweise einen minimalen Algorithmus?

12.3 Einige Resultate für spezielle Theorien

Wir geben eine tabellarische Übersicht über einige Theorien, die bereits näher untersucht wurden. Es sind dies im wesentlichen die durch folgende Axiome definierten Theorien:

$$
\begin{array}{lll}
\emptyset(f) & := \emptyset & \text{(leere Theorie; freie Funktion)} \\
A(f) & := \{f(x, f(y, z)) = f(f(x, y), z)\} & \text{(Assoziativität)} \\
C(f) & := \{f(x, y) = f(y, x)\} & \text{(Kommutativität)} \\
I(f) & := \{f(x, x) = x\} & \text{(Idempotenz)} \\
Dl(f, g) & := \{f(g(x, y), z) = g(f(x, z), f(y, z))\} & \text{(Links-Distributivität)} \\
Dr(f, g) & := \{f(x, g(y, z)) = g(f(x, y), f(x, z))\} & \text{(Rechts-Distributivität)} \\
D(f, g) & := D_l(f, g) \cup D_r(f, g) & \text{(Distributivität)} \\
Invl(f, i, e) & := \{f(i(x), x) = x\} & \text{(Links-Inverse)} \\
Invr(f, i, e) & := \{f(x, i(x)) = x\} & \text{(Rechts-Inverse)} \\
Inv(f, i, e) & := Inv_l(f, i, e) \cup Inv_r(f, i, e) & \text{(Inverse)} \\
Nl(f, e) & := \{f(e, x) = x\} & \text{(Links-Neutrales)} \\
Nr(f, e) & := \{f(x, e) = x\} & \text{(Rechts-Neutrales)} \\
N(f, e) & := N_l(f, e) \cup N_r(f, e) & \text{(Neutrales)} \\
Iv(f) & := \{f(f(x)) = x\} & \text{(Involution)} \\
E(h, f) & := \{h(f(x_1, \ldots, x_n)) = f(h(x_1), \ldots, h(x_n))\} & \text{(Endomorphismus)} \\
AE(h, f) & := \{h(f(x_1, \ldots, x_n)) = f(h(x_n), \ldots, h(x_1))\} & \text{(Anti-Endomorphismus)}
\end{array}
$$

Durch Kombinationen aus diesen Axiomen lassen sich bereits viele wichtige mathematische Theorien aufbauen: Monoide ($A \cup N$), Gruppen ($A \cup N \cup Inv$) und andere.

Spezielle Kombinationen, die in der Unifikationstheorie untersucht wurden, sind zum Beispiel

$$
\begin{array}{ll}
AC(f) & := A(f) \cup C(f) \\
AI(f) & := A(f) \cup I(f) \\
ACI(f) & := A(f) \cup C(f) \cup I(f)
\end{array}
$$

Folgende Tabelle spiegelt einige Eigenschaften dieser Theorien wider:

Theorie	entscheidbar	Typ	Algorithmus
$\emptyset$	ja	unitär	minimal
$A(f)$	ja	infinitär	typkonform
$C(f)$	ja	finitär	typkonform
$I(f)$	ja	finitär	typkonform
$AC(f)$	ja	finitär	minimal
$AI(f)$	ja	0	?
$D(f, g)$	ja	infinitär	typkonform*
$D(f, g) \cup A(f)$	nein	infinitär	typkonform*
$D(f, g) \cup AC(f)$	nein	infinitär	typkonform*
$E(f, g)$	ja	unitär	minimal
$H(f, g_1, g_2)$	ja	unitär	minimal
Minus-Theorien	ja	(in)finitär**	typkonform
Abelsche Gruppen	ja	finitär	minimal
Boolesche Ringe	ja	unitär**	minimal

12.4 Theorieresolution mit "compilierten Theorien"

Die Idee der compilierten Theorien ist, bestimmte Klauseln aus der Klauselmenge herauszunehmen und durch eine spezielle Theorieresolutionsregel zu ersetzen.

Beispiel 12.17. Die Symmetrieklausel: $\neg P(x,y), P(y,x)$ erlaubt, die beiden unären Klauseln $P(a,b)$ und $\neg P(b,a)$ durch zwei aufeinanderfolgende Resolutionsschritte zu widerlegen. Viel einfacher ist es jedoch, einen Unifikationsalgorithmus zu benutzen, der die Symmetrie direkt ausnutzt und dann mit einem Schritt die Widerlegung findet.

Definition 12.18. *Compilierte Theorieresolution. Sei $C = \{L_1, \ldots, L_n\}$ eine Klausel mit zwei Literalen, die nicht selbstresolvierend ist, d.h. nicht mit einer Kopie von sich selbst resolviert. Dann läßt sich daraus folgende Theorieresolutionsregel erzeugen:*

$K_1 \vee R_1$	*L_i und K_i haben unterschiedliches Vorzeichen.*
$\ldots$	*σ ist simultaner Unifikator für $\lvert L_i\rvert$[1] $\stackrel{?}{=} \lvert K_i\rvert, i = 1, \ldots, n$, d.h*
$\ldots$	$\lvert\sigma L_1\rvert = \lvert\sigma K_1\rvert$
$\ldots$	$\lvert\sigma L_n\rvert = \lvert\sigma K_n\rvert$
$K_n \vee R_n$	*σ' ist σ eingeschränkt auf die Variablen in $K_1, \ldots, K_n$*
$\sigma'(R_1 \vee \ldots \vee R_n)$	*(Theorieresolvente in der Theorie von C)*

Beispiel 12.19.

1. Sei $C = \{\neg P(x,y), P(y,x)\}$ (Die Selbstresolvente[2] ist eine Tautologie und kann daher ignoriert werden.)

$P(a,z), Q(z)$	$\sigma = \{x \mapsto a, y \mapsto b, z \mapsto b\}$
$\neg P(b,a)$	$\sigma' = \{z \mapsto b\}$
$Q(b)$	

2. Sei $C = \{\neg links(x,y), rechts(y,x)\}$

[1] $|L|$ bedeutet L ohne Vorzeichen, d.h. den atomaren Anteil von L.
[2] Das Ergebnis der Resolution einer Klausel mit sich selbst (bzw. einer Kopie)

$$\begin{array}{ll} links(a,z), Q(z) & \sigma = \{x \mapsto a, y \mapsto b, z \mapsto b\} \\ \neg rechts(b,a) & \sigma' = \{z \mapsto b\} \\ \hline Q(b) & \end{array}$$

3. Sei $C = \{\neg\texttt{Vater}(x,y), \neg\texttt{Vater}(y,z), \texttt{Grossvater}(x,z)\}$

$$\begin{array}{ll} \texttt{Vater}(Uli, Karl), S & \sigma = \{x \mapsto Uli, y \mapsto Karl, z \mapsto Willi, u \mapsto Willi\} \\ \texttt{Vater}(Karl, Willi), R & \sigma' = \{u \mapsto Willi\} \\ \neg\texttt{Grossvater}(Uli, u), Q(u) & \\ \hline S, R, Q(Willi) & \end{array}$$

Die Korrektheit des Verfahrens ergibt sich unmittelbar aus der Tatsache, daß ein solcher Theorieresolutionsschritt sich in eine Folge von n einzelnen Resolutionsschritten zerlegen läßt, die wiederum korrekt sind.

Vorteile des Verfahrens: Größere Schritte, zielgerichteteres Vorgehen.
Nachteile: Eventuell größere Verzweigungsrate im Suchraum und damit eventuell mehr Suche.

Das folgende Beispiel zeigt, was passieren kann wenn man die Forderung, daß die Klausel C nicht selbstresolvierend ist, fallen läßt.

Beispiel 12.20. **Gegenbeispiel:**
$C = \{\neg P(x), P(f(x))\}$ Die Klauseln $P(a)$ und $\neg P(f(f(a)))$ sind zusammen mit C unerfüllbar und auch widerlegbar mit normaler Resolution, jedoch nicht mit der compilierten Theorieresolution: (Die Selbstresolventen sind: $\{\neg P(x), P(f(f(x)))\}$, $\{\neg P(x), P(f(f(f(x))))\}$, ...
Mit der ersten Selbstresolvente würde es funktionieren. Dann funktionieren aber andere Beispiele nicht.)

Es bleibt noch das Problem, wie man mehrere Klauseln gleichzeitig compiliert. Dafür gilt:

Satz 12.21. *Gegeben eine endliche konsistente Klauselmenge D, die alle ihre Resolventen und Faktoren enthält. Dann kann man alle diese Klauseln einzeln in eine Theorieresolutionsregel übersetzen und erhält damit einen korrekten und vollständigen Kalkül.*

Beweis. Der Beweis geht ganz analog zum Vollständigkeitsbeweis für Resolution: Man zeigt zuerst die Vollständigkeit auf der (variablenfreien) Grundebene und "liftet" dann die Schritte. Der einzige wesentliche Unterschied der sich ergibt, ist der Basisfall in der "literal excess number" Induktion (vgl. 5.5). In diesem Fall ist die Menge der unären Grund Klauseln C_{gr} nicht mehr notwendigerweise allein dadurch widersprüchlich weil sie zwei komplementäre Klauseln enthält, sondern auch weil sie mit der Menge D zusammen widersprüchlich ist. Da D vervollständigt ist bzgl. Resolventen und Faktorenbildung, genügt jedoch eine Klausel A aus D, um mit der passenden Anzahl von unären Klauseln aus C_{gr} zur leeren Klausel zu resolvieren. Diese Folge von Resolutionen läßt sich jetzt als ein Theorieresolutionsschritt modulo A interpretieren. Der Rest des Beweises geht ganz analog zur normalen Resolution. □

Beispiel 12.22. Gegeben seien die Klauseln $\{P(x), Q(x)\}$ und $\{\neg Q(y), R(y)\}$. Eine daraus vervollständigte Menge ist: $\{\{P(x), Q(x)\}, \{\neg Q(y), R(y)\}, \{\neg P(z), R(z)\}\}$.

Jetzt kann man damit die folgende Klauselmenge widerlegen:

$$\neg P(a), Q(b), P(c)$$

$$\neg Q(a)$$

$$\neg R(b)$$

$$\neg R(c)$$

12.5 Resolution mit bedingten Klauseln (Constraints)

Die Idee der Theorieresolution kann man verallgemeinern zur Resolution mit eingeschränkten Quantoren oder auch Constraint-Resolution (siehe [Bür91]). Wenn die Hintergrundtheorie eine Gleichungstheorie ist, dann ist es damit möglich trotz einer großen oder unendlichen Anzahl von Unifikatoren die Verzweigungsrate des Suchraumes klein zu halten, indem man andere Darstellungen für die Menge der Unifikatoren wählt. Eine solche ist zum Beispiel das Gleichungssystem selbst, das ursprünglich zu lösen war.

Statt Klauseln nimmt man bedingte Klauseln bestehend aus einem Paar $(\mathcal{C}, C)$, wobei $\mathcal{C}$ eine *Bedingung (constraint)* ist und C eine normale Klausel. Wir wollen nur den einfachen Fall betrachten, daß die Prädikate der Theorie in dem Literalteil C der Klauseln nicht vorkommen. Den Constraintteil kann man sich als eine PL_1-Formel vorstellen. Im Fall der Gleichungstheorien wäre zum Beispiel ein Gleichungssystem $\{x \stackrel{?}{=} f(y), g(x) \stackrel{?}{=} g(y)\}$ ein solche Bedingung. Die logische Bedeutung einer verallgemeinerten Klausel $(\mathcal{C}, C)$ ist einfach $\forall(V)(\mathcal{C} \Rightarrow C)$[3], wobei V die Menge aller in $(\mathcal{C}, C)$ freien Variablen ist. Klauseln, deren Constraint-Anteil unerfüllbar ist, d.h. $\forall(V)\neg\mathcal{C}$ gilt, sind Tautologien. Um

[3] $\forall(V)$ soll die Quantifikation über alle Variablen in V bezeichnen

diese Klauseln sicher zu erkennen, ist ein Entscheidungsverfahren für die Konsistenz von Bedingungen erforderlich. Die leere Klausel ist $(\mathcal{C}, \emptyset)$, wobei $\mathcal{C}$ ein konsistenter Constraint sein muß.

Ein (einfacher) *bedingter Resolutionsschritt* sieht dann wie folgt aus: $(\mathcal{C}_1, \{L_1\} \cup C_1)$ und $(\mathcal{C}_2, \{\neg L_2\} \cup C_2)$ haben als Resolvente: $(\mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \{L1 \stackrel{?}{=} L_2\}, C_1 \cup C_2)$. Ein verallgemeinerter Resolutionsschritt schließt die Faktorisierung mit ein, indem statt L_1 und L_2 mehrere Literale gleichzeitig an der Resolution teilnehmen können.

Es sieht so aus, als sei Unifikation und Variableneinsetzung unnötig geworden. Dies ist nicht der Fall. Man hat aber etwas gewonnen: um eine Resolutionsschritt auszuführen, braucht man nur zu testen, ob die zugehörige Bedingung konsistent ist.

Die bedingte Resolutionsmethode ist am einfachsten und anschaulichsten für sogenannte *definite Theorien erster Ordnung*, das sind Theorien, die mit einer endlichen Hornklauselmenge (siehe Kapitel. 8) definiert werden können. In diesen gilt, daß die Bedingungen Lösungen haben, die als (evtl. unendliche) Menge von Substitutionen beschrieben werden können. Eine etwas allgemeinere Formulierung ist, daß die Theorie ein *generisches Modell* $\mathcal{G}$ hat, d.h. für jede Bedingung $\mathcal{C}$ gilt $\mathcal{G} \models \exists\mathcal{C}$ gdw. für alle Modelle $\mathcal{M} : \mathcal{M} \models \exists\mathcal{C}$. Für solche Theorien ist die normale bedingte Resolution vollständig:

Satz 12.23. *Für Theorien erster Ordnung mit generischem Modell kann allgemeine bedingte Resolution aus jeder unerfüllbaren bedingten Klauselmenge die leere Klausel ableiten.*

Eine Theorie, die nicht definit ist, ist z.B. die durch $Pa \vee Pb$ definierte Theorie. Den Begriff einer Lösung für die Bedingung $\{P(x)\}$ läßt sich nicht mehr so formulieren wie man das für Unifikatoren gewöhnt ist. Was man noch sagen kann ist: $\{P(x)\}$ ist äquivalent zu $\{x \stackrel{?}{=} a \vee x \stackrel{?}{=} b\}$. Dies kann man als "indefinite Lösung" ansehen. Die Konsistenzbedingung für Bedingungen in indefiniten Theorien ist nur, daß die Bedingung in einem Modell konsistent ist. Für solche Theorien gilt der folgende Vollständigkeitssatz:

Satz 12.24. *Für eine unerfüllbare Menge von bedingten Klauseln lassen sich mittels bedingter Resolution endlich viele leere Klauseln* $(\mathcal{C}_1, \emptyset), \ldots, (\mathcal{C}_n, \emptyset\}$ *ableiten, so daß die Konjunktion* $(\mathcal{C}_1 \wedge \ldots \wedge (\mathcal{C}_n$ *unerfüllbar ist, d.h. man hat einen Widerspruch.*

Für ein Deduktionssystem für bedingte Resolution bedeutet das, daß nicht nur eine leere Klausel zu suchen ist, sondern daß viele leere Klauseln abzuleiten und zu sammeln sind und die Konjunktion der Constraints auf Erfüllbarkeit zu testen. Bisher hat der allgemeine Fall der bedingten Resolution für indefinite Theorien keine praktische Bedeutung erlangt. Implementierungen gibt es jedenfalls noch keine.

Beispiel 12.25. Sei f ein assoziatives Funktionssymbol mit dem Axiom $f(x, f(y, z)) = f(f(x, y), z)$ und sei folgende Klauselmenge gegeben:

$C1\ (\emptyset, Qf(a,b))$

$C2\ (\emptyset, \neg Qx, Pf(x,y))$

$C3\ (\emptyset, \neg P(f(a, f(b,z))))$

Bedingte Resolution auf $C2, 2$ und $C3$ ergibt:

$R1$	$(\{f(x,y) \stackrel{?}{=} f(a, f(b,z))\}, \neg Q(x))$	Bedingung ist erfüllbar
$R1 + C1$	$(\{f(x,y) \stackrel{?}{=} f(a, f(b,z)), x \stackrel{?}{=} f(a,b)\}, \emptyset)$	Bedingung ist erfüllbar mit
		$\{x \mapsto f(a,b), y \mapsto z\}$

Hätte man assoziative Unifikation und Resolution verwendet, dann hätte man eine Menge der allgemeinsten Unifikatoren von $f(x,y) \stackrel{?}{=} f(a, f(b,z))$ betrachten müssen. Dies sind unter anderem: $\{x \mapsto a, y \mapsto f(b,z)\}, \{x \mapsto f(a,b), y \mapsto z\}, \ldots$

Mittels bedingter Resolution lassen sich auch ganzzahlige Arithmetik oder ähnlich bekannte Theorien in einem Deduktionssystem verwenden. Dies sind dann evtl. noch definite Theorien, aber sie sind nicht unbedingt erster Ordnung, denn z.B. die Menge der natürlichen Zahlen läßt sich nicht mit einer Theorie erster Ordnung charakterisieren. Bei der bedingten Resolution tritt dann das Problem auf, daß die Konsistenz von Bedingungen möglicherweise nicht mehr entscheidbar ist.