Allgemeine AC-Unifikation durch Variablenabstraktion mit Fremdtermbedingungen

Jörn Richts

Diplomarbeit

Fachbereich Informatik, Universität Kaiserslautern

Betreuung:
Prof. Dr. Klaus Madlener
Dr. Franz Baader
Dr. Hans–Jürgen Bürckert
Dipl.–Inform. Axel Präcklein

Abstract

This work investigates general AC-unification, i. e. unification in the combination of free function symbols and free Abelian semigroups, whose function symbols fulfill associativity and commutativity.

The three necessary parts of general AC-unification are presented: a combination algorithm, a procedure for elementary AC-unification, and methods for solving systems of diophantine equations. Starting with A. Boudet's unification algorithm for the combination of regular and collapse-free theories, an efficient algorithm for general AC-unification is developed, setting out conditions that must be fulfilled by the solutions of elementary AC-unification. These conditions are used by the procedure for elementary AC-unification whereby further conditions are set out that must be fulfilled by the solutions of the diophantine equations. Then three methods (those of E. Contejean and H. Devie, E. Domenjoud, L. Pottier) for solving systems of diophantine equations are presented. The three methods are compared and evaluated, trying to use the conditions efficiently.

Finally some problems which arise from the reuse of AC-unifiers in unification, such as merging, are presented. It is shown that reusing AC-unifiers for partial problems is often worse than solving the entire problem from the beginning.

Zusammenfassung

Diese Arbeit befaßt sich mit allgemeiner AC-Unifikation, d. h. mit der Unifikation in der Kombination von freien Funktionssymbolen mit freien Abelschen Halbgruppen, also mit Funktionssymbolen, für die die Assoziativität und die Kommutativität gilt.

Dabei werden die drei zur allgemeinen AC-Unifikation notwendigen Teile beschrieben, d. h. ein Kombinationsverfahren, ein Verfahren zur elementaren AC-Unifikation und Verfahren zum Lösen diophantischer Gleichungssysteme. Ausgehend von dem Unifikationsalgorithmus zur Kombination von regulären, kollapsfreien Theorien von A. Boudet wird ein effizientes Verfahren zur allgemeinen AC-Unifikation entwickelt, indem Bedingungen aufgestellt werden, die die Lösungen der elementaren AC-Unifikation erfüllen müssen. Diese Bedingungen werden dann vom Verfahren zur elementaren AC-Unifikation benutzt, wobei weitere Bedingungen aufgestellt werden, die von den Lösungen der diophantischen Gleichungen erfüllt werden müssen. Danach werden drei Verfahren (von E. Contejean und H. Devie, E. Domenjoud, L. Pottier) zum Lösen diophantischer Gleichungssysteme beschrieben, für die versucht wird, diese Bedingungen möglichst effizient auszunutzen. Anschließlich werden die drei Verfahren verglichen und bewertet.

Abschließend werden noch einige Probleme vorgestellt, die beim Wiederverwenden von AC-Unifikatoren in der Unifikation, wie z. B. beim Merging, auftreten. Dabei zeigt sich, daß das Wiederverwenden von AC-Unifikatoren für Teilprobleme häufig schlechtere Ergebnisse liefert als es das erneute Lösen des Gesamtproblems tun würde.

Inhaltsverzeichnis

Kapitel 1

Einführung und Definitionen

1.1 Motivation und Einleitung

Unifikation spielt in der Künstlichen Intelligenz seit der Beschreibung der Resolution durch Robinson [20] 1965 eine tragende Rolle. Eine weitere wichtige Anwendung besteht in der Termersetzung und der Vervollständigung von Knuth und Bendix [15] von 1967. Die Aufgabe der Unifikation in beiden Verfahren ist es, die Variablen zweier Terme durch eine Substitution so zu belegen, daß die beiden Terme syntaktisch gleich werden und dadurch eine Inferenzregel anwendbar wird. Da mit beiden Kalkülen Sätze automatisch bewiesen werden können, ist die Mathematik ein wichtiges Anwendungsgebiet.

In der Mathematik spielen Gleichungen eine Schlüsselrolle. Sie dienen dazu, algebraische Strukturen zu definieren, wie z. B. Monoide, Gruppen, Abelsche Gruppen usw. Da die Behandlung der Gleichungen in der Resolution, z. B. durch Paramodulation, zu Komplexitätsschwierigkeiten führt, schlug Plotkin [18] 1972 vor, die Behandlung der Gleichungen, oder eines Teils davon, in der Unifikation vorzunehmen. Diese Theorie- oder E-Unifikation macht Terme nicht mehr syntaktisch gleich, sondern nur gleich bezüglich der Gleichheitstheorie E, einer Menge von Gleichungen. Ein häufig vorkommendes Beispiel dafür ist die Assoziativität und Kommutativität für ein Symbol $+$, d. h. $E = \{+(x,+(y,z)) = +(+(x,y),z), +(x,y) = +(y,x)\}$. Auch für die Vervollständigung ist Theorieunifikation sinnvoll, da bestimmte Gleichungen, die nicht richtbar sind, sonst nur durch eine komplexitätssteigernde Variante des Verfahrens behandelt werden können.

Ein Nachteil der Theorieunifikation ist, daß das Ergebnis gegenüber der syntaktischen Unifikation komplizierter wird. Bei der syntaktischen Unifikation reicht eine Substitution, der allgemeinste Unifikator, aus, um alle, meist unendlich viele, Lösungen zu repräsentieren, weil es immer genau eine Lösungssubstitution gibt, die bezüglich der Instanzenordnung (Definition Seite 7) minimal ist. In der Theorieunifikation hingegen gibt es im allgemeinen mehrere minimale Lösungen, so daß das Ergebnis der Unifikation gewöhnlich eine Menge von Substitutionen ist. Durch die Unifikationshierarchie nach Siekmann [22] (Definition auf Seite 13) werden die Theorien danach klassifiziert, welche Kardinalität diese Menge haben kann und ob sie überhaupt existiert. Dieser Unifikationstyp ist im allgemeinen unentscheidbar [17] und hängt außer von der Theorie, unter der unifiziert wird, auch

noch davon ab, ob nur ein Termpaar oder mehrere Paare unifiziert werden sollen und ob die Terme neben den Funktionssymbolen der Theorie auch noch freie Funktionssymbole enthalten.

Um Terme unifizieren zu können, die Symbole aus verschiedenen Theorien oder freie Symbole enthalten, benötigt man ein Kombinationsverfahren, das die Algorithmen für die verschiedenen Theorien und die syntaktische Unifikation benutzt. Die ersten impliziten Kombinationsverfahren wurden für die Kombination von freien Symbolen mit Symbolen, für die die Assoziativität und Kommutativität gelten, in [24] und [10] beschrieben. Für Theorien mit bestimmten Eigenschaften haben Kirchner [14], Herold [11], Tidén [25] und Yelick [26] Kombinationsverfahren vorgestellt. Die Verfahren von Schmidt-Schauß [21] und Boudet [2] erlauben die Kombination von beliebigen Theorien mit disjunkten Mengen von Funktionssymbolen. Da der Algorithmus von Boudet regelbasiert ist, erlaubt er eine besonders flexible Kontrolle und ist daher eine Grundlage für diese Arbeit.

Die Theorie der freien Abelschen Halbgruppe, kurz AC-Theorie, ist die Theorie, in der für ein Funktionssymbol die Assozitivität $(x + (y + z) = (x + y) + z)$ und die Kommutativität $(x + y = y + x)$ gilt. Sie repräsentiert die Datenstruktur der Multimenge und wird in der englischen Literatur häufig als „bag“ bezeichnet. Sie ist wie kaum eine andere für die Theorieunifikation untersucht worden [23, 24, 8, 16, 10]. Dies liegt einerseits daran, daß sie in der Mathematik eine bedeutende Rolle spielt, weil sie für die wichtigsten Funktionen gilt (Addition und Multiplikation von Zahlen, Vereinigung und Durchschnitt von Mengen, Disjunktion und Konjunktion in der Boolschen Algebra) und die Grundlage für viele algebraische Strukturen ist (Abelsche Gruppen, Ringe, ...). Andererseits ist sie für die Vervollständigung von großer Bedeutung, weil die Kommutativität nicht richtbar ist und auch die Assoziativität nicht mehr richtbar ist, wenn die Kommutativität in der Unifikation behandelt wird. Erweiterte Theorien, die noch zusätzliche Gleichungen enthalten, werden seltener benutzt, weil die Unifikationsalgorithmen meist komplizierter werden und für die Vervollständigung immer weniger Gleichungen richtbar sind.

Der erste Algorithmus zur Unifikation von Termen, die ein AC-Symbol und sonst nur Konstanten und Variablen enthalten, wurde 1975 von Stickel [23] angegeben. Von Stickel [24] und Herold und Siekmann [10] wurden später zwei verschiedene Verfahren vorgestellt, die Terme mit mehreren AC-Symbolen und auch mehrstelligen freien Funktionssymbolen unifizieren können. Diese beiden Verfahren enthalten implizit auch ein Kombinationsverfahren für AC-Theorien und die freie Theorie. Während Stickels Verfahren dabei Variablenabstraktion benutzt, arbeitet das Verfahren von Herold und Siekmann mit Konstantenabstraktion. Für die allgemeinen Kombinationsverfahren wurde später meist eine Mischung von Konstantenabstraktion und Variablenabstraktion benutzt, d. h. die zur Abstraktion eingeführten Symbole werden in einigen Schritten als Konstanten und in einigen Schritten als Variablen behandelt oder sie werden als Variablen mit eingeschränktem Wertebereich betrachtet.

In allen Verfahren zur AC-Unifikation werden diophantische Gleichungen, d. h. lineare Gleichungen mit ganzzahligen Koeffizienten und natürlichzahligen Lösungen, aufgestellt. Mit der minimalen Lösungsmenge dieser Gleichungen werden dann die Lösungssubstitutionen berechnet. Wegen dieses engen Zusammenhangs mit der AC-Unifikation ist auch

das Lösen diophantischer Gleichungen häufig untersucht worden [5, 9, 6, 7, 19]. Von den recht einfachen Verfahren am Anfang, die nur die Lösungen einer einzelnen Gleichung berechnen können, kam man zu komplizierteren und effizienteren Algorithmen und zu Algorithmen, die auch für Gleichungssysteme geeignet sind. Auch die Grenzen, die die Größe der minimalen Lösungen beschränken und die in den meisten Algorithmen eine wichtige Rolle spielen, sind immer mehr verfeinert worden. In Kapitel 4 werden die Verfahren von Contejean und Devie [6], Domenjoud [7] und Pottier [19] vorgestellt und analysiert.

Bei der Anwendung eines allgemeinen Kombinationsalgorithmus auf spezielle Theorien treten häufig Situationen auf, in denen sich schon berechnete Teillösungen als unbrauchbar herausstellen, weil sie für das gesamte Problem nicht zu einer Lösung führen. In dieser Arbeit sollen nun, wie schon bei Stickel [24] angesprochen, für die Kombination von AC-Theorien mit der freien Theorie Bedingungen aufgestellt werden, die diese Situation frühzeitig erkennen und die Berechnung einer unbrauchbaren Teillösung verhindern. Diese Bedingungen sollen dann in den Kombinationsalgorithmus von Boudet [2] eingearbeitet werden. Dabei wird sich herausstellen, daß ein wesentlicher Bestandteil des Verfahrens, die Variablenabstraktion und die dadurch erzeugten Abstraktionsvariablen, explizit nicht mehr benötigt werden. Vom Kombinationsalgorithmus werden die Bedingungen an den Algorithmus zur Unifikation von Termen einer AC-Theorie weitergegeben und anschließend von diesem weiter an den Algorithmus zur Lösung diophantischer Gleichungen, so daß schon von diesem überflüssige Lösungen nicht erzeugt werden.

1.2 Terme

Als erstes werden an [3] angelehnt einige Standardbegriffe definiert. Dabei ist in $\mathbb{N}$, der Menge der natürlichen Zahlen, die Null stets mit enthalten.

Eine **Signatur** ist eine Menge von Funktionssymbolen mit fester Stelligkeit. Im folgenden sei Σ eine Signatur und $\mathcal{X}$ eine (unendliche) abzählbare Menge von Variablen. Dann ist $\mathcal{T}(\Sigma, \mathcal{X})$ die Menge der **Terme** über den Funktionssymbolen aus Σ und den Variablen aus $\mathcal{X}$ mit $t \in \mathcal{T}(\Sigma, \mathcal{X})$ genau dann, wenn $t \in \mathcal{X}$ oder $t = f(t_1, \ldots, t_n)$, wobei $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$, $f \in \Sigma$ und n die Stelligkeit von f ist. Die Menge der Variablen eines Terms t wird mit $Var(t)$ bezeichnet.

Die Teilterme eines Terms sind durch ihre **Stellen** festgelegt. Stellen sind Folgen von natürlichen Zahlen, wobei ϵ die leere Folge ist. Die Stellen eines Terms t sind $O(t)$ und $t|_p$ ist der Teilterm an der Stelle p:

Ist $t \in \mathcal{X}$, so ist $O(t) = \{\epsilon\}$ und $t|_\epsilon = t$.

Sonst ist $t = f(t_1, \ldots, t_n)$ mit $f \in \Sigma$ und $n \in \mathbb{N}$. Dann ist

$$O(t) = \{\epsilon\} \cup \{ip \mid i \in \{1, \ldots, n\}, p \in O(t_i)\},\ t|_{ip} = t_i|_p \text{ und } t|_\epsilon = t.$$

Auf den Stellen ist die Präfixordnung $<_\mathrm{p}$ definiert und sie lassen sich als Folgen einfach konkatenieren, wobei $t|_{pq} = (t|_p)|_q$ gilt. Das **Kopfsymbol**, d. h. das oberste Symbol oder

Topsymbol, eines Terms t ist $t(\epsilon)$ und $t(p) = t|_p(\epsilon)$ ist das Symbol an der Stelle $p \in O(t)$. Das Vorkommen eines Teilterms s in t wird mit $t[s]$ bezeichnet oder genauer mit $t[s]_p$, wenn $t|_p = s$. Die Termersetzung an einer Position p in t durch s wird mit $t[p \leftarrow s]$ notiert.

Um Terme zu instanzieren, d. h. ihre Variablen durch andere Terme zu ersetzen, benutzt man Substitutionen. Eine **Substitution** weist einer endlichen Menge von Variablen aus $\mathcal{X}$ Terme aus $\mathcal{T}(\Sigma, \mathcal{X})$ zu und kann durch $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$ zu einer Abbildung von $\mathcal{T}(\Sigma, \mathcal{X})$ nach $\mathcal{T}(\Sigma, \mathcal{X})$ erweitert werden. Notiert werden Substitutionen als $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. Die Konkatenation einer Substitution σ mit einer anderen Substitution τ ist durch $\sigma \circ \tau(x) = \sigma(\tau(x))$ für alle $x \in \mathcal{X}$ definiert.

Eine Menge von Gleichungen (Axiomen) $E \subseteq \mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$ definiert eine **Gleichheitstheorie** $=_E$, d. h. eine Kongruenzrelation auf Termen, die operational definiert ist durch

$$
\begin{array}{lll}
s \vdash\!\dashv t & \text{genau dann,} & \text{wenn es } p \in O(s),\ (l, r) \in E \text{ und eine Substitution } \sigma \text{ gibt mit} \\
 & & s|_p = \sigma(l) \text{ und } t = s[p \leftarrow \sigma(r)] \text{ oder} \\
 & & s|_p = \sigma(r) \text{ und } t = s[p \leftarrow \sigma(l)].
\end{array}
$$

Die E-Gleichheit $=_E$ ist dann $\overset{*}{\vdash\!\dashv}$, d. h. die transitive, reflexive Hülle von $\vdash\!\dashv$.

Für das Arbeiten mit $=_E$ ist es sinnvoll, E in Teilmengen zu unterteilen, wenn die Mengen der Funktionssymbole der Teiltheorien disjunkt bleiben. Im folgenden sei E also in die Teiltheorien $E_1 \mathbin{\mathring{\cup}} \ldots \mathbin{\mathring{\cup}} E_n = E$ unterteilt, wobei $\mathcal{F}_1, \ldots, \mathcal{F}_n \subseteq \Sigma$ mit $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ für $i \neq j$ die Funktionssymbole dieser Theorien sind. Die Funktionssymbole, die nicht in E vorkommen, sind die freien Funktionssymbole $\mathcal{F}_0$ und werden der leeren Theorie $E_0 = \emptyset$ zugeordnet. Es gilt also $\Sigma = \mathring{\bigcup}_{h=0}^{n} \mathcal{F}_h$. Da die freien Konstanten eine besondere Rolle spielen, wird $\mathcal{F}_0$ in $\mathcal{F} \cup \mathcal{C} \subseteq \Sigma$ unterteilt, wobei $\mathcal{C}$ die Menge der freien Konstanten und $\mathcal{F}$ die Menge der mehrstelligen, freien Funktionssymbole ist. Die freien Konstanten eines Terms t sind $Const(t) \subseteq \mathcal{C}$. Ein Teilterm s von t mit $s = t|_p$ ist ein **Fremdterm** in t, falls $t(\epsilon) \in \mathcal{F}_i$, $s(\epsilon) \in \mathcal{F}_j$ mit $i \neq j$ und für alle Stellen $q <_{\mathrm{p}} p$ gilt $t(q) \in \mathcal{F}_i$.

Eine Theorie $=_E$ oder kurz E heißt **regulär**, wenn aus $s =_E t$ folgt $Var(s) = Var(t)$. Sie heißt **kollapsfrei**, wenn aus $s =_E t$ mit $s \neq t$ folgt $s \notin \mathcal{X}$ und $t \notin \mathcal{X}$. Gibt es keine Gleichung $s =_E t$, bei der t ein echter Teilterm von s ist, heißt sie **simpel**. Eine Gleichheitstheorie $=_E$ ist genau dann regulär oder kollapsfrei, wenn die Axiome in E dies sind.

Die Menge $\mathcal{AC} \subseteq \Sigma$ enthält zweistellige Funktionssymbole $+$, für die die Theorie der **Abelschen Halbgruppe** $AC^+ = \{+(x, +(y, z)) = +(+(x, y), z), +(x, y) = +(y, x)\}$, d. h. die Assoziativität und Kommutativität gilt. Die Symbole in $\mathcal{AC}$ werden zur besseren Lesbarkeit häufig infix geschrieben und wegen der Assoziativität können die Klammern weggelassen werden, also $x + y + z$ statt $+(x, +(y, z))$. Die **AC^+-Argumente** eines Terms t für ein $+ \in \mathcal{AC}$ sind

$$
Arg^+(t) = \begin{cases} Arg^+(t_1) \cup Arg^+(t_2), & \text{falls} \quad t = +(t_1, t_2) \\ \{t\}, & \text{sonst.} \end{cases}
$$

Arg^+ und Var können auf natürliche Weise auf alle Objekte, die Terme enthalten, erweitert werden.

Beispiel 1.1 Ist $t = +(a, +(x, f(b, y)))$ mit $x, y \in \mathcal{X}$, $a, b \in \mathcal{C}$, $f \in \mathcal{F}_0$ und $+ \in \mathcal{AC}$, so ist $t(\epsilon) = +$, $Var(t) = \{x, y\}$, $Const(t) = \{a, b\}$ und $Arg^+(t) = \{a, x, f(b, y)\}$. $Arg^+(f(b, y))$ ist $\{f(b, y)\}$ und $x(\epsilon) = x$. ■

Lemma 1.1 *AC ist eine reguläre und kollapsfreie Theorie.*

Beweis: Dies folgt unmittelbar aus der Definition von regulär und kollapsfrei und aus den Axiomen in AC. □

Um Substitutionen danach zu vergleichen, wie stark sie einen Term instanzieren, wird die Instanzenordnung definiert. Anstatt der üblichen Instanzenordnung in der freien Theorie benutzt man aber die **eingeschränkte Instanzenordnung**, die zwei Substitutionen nur für eine endliche Menge von Variablen vergleicht. Sie ist für zwei Substitutionen σ und ρ und eine Variablenmenge $\mathcal{V} \subseteq \mathcal{X}$ definiert durch

$$\begin{array}{lll} \sigma \leq_E \rho[\mathcal{V}] & \text{genau dann, wenn} & \text{eine Substitution } \tau \text{ existiert mit} \\ & & \tau(\sigma(x)) =_E \rho(x) \text{ für alle } x \in \mathcal{V} \\ \sigma <_E \rho[\mathcal{V}] & \text{genau dann, wenn} & \sigma \leq_E \rho[\mathcal{V}] \text{ und nicht } \rho \leq_E \sigma[\mathcal{V}]. \end{array}$$

Bei der Unifikation wird die Instanzenordnung benutzt, die auf die Variablenmenge der ursprünglich zu unifizierenden Terme eingeschränkt ist. Durch diese Einschränkung der Instanzenordnung wird die zu berechnende Lösungsmenge der Unifikation, die minimal bezüglich dieser Ordnung ist, kleiner, und der Unifikationstyp ändert sich unter Umständen. Dies liegt daran, daß die durch die Unifikation neu eingeführten Variablen in den Lösungssubstitutionen wegen der Einschränkung auf die ursprünglichen Variablen nicht verglichen werden. Ihre Belegung durch τ in der Instanzenordnung ist für das ursprüngliche Unifikationsproblem irrelevant.

1.3 Vektoren

Für einen Vektor $\vec{x} \in \mathbb{R}^n$ ist $\vec{x}(i)$ die i-te Komponente, $l(\vec{x}) = \sum_{i=1}^{n} \vec{x}(i)$ die **Länge** und $h(\vec{x}) = \max |\vec{x}(i)|$ die **Höhe** des Vektors. Die **Trägermenge** von $\vec{x}$ ist $\{i \mid \vec{x}(i) \neq 0\}$, die Menge der Indizes der Komponenten ungleich Null. Zur Vereinfachung der Schreibweise werden Vektoren, die normalerweise senkrecht ausgerichtet sind, manchmal entgegen der üblichen Notation auch waagerecht geschrieben. Die korrekte Orientierung geht meist aus dem Zusammenhang hervor und wird in mißverständlichen Situationen explizit angegeben. Außerdem wird die i-te Komponente eines Vektors manchmal als i-te Spalte bezeichnet, weil dies dem Bild von Vektoren entspricht, die zur besseren Übersicht waagerecht untereinander angeordnet sind.

Für zwei Vektoren $\vec{x}, \vec{y} \in \mathbb{R}^n$ gilt

$$\begin{array}{rll} \vec{x} \leq^n \vec{y} & \text{genau dann, wenn} & \vec{x}(i) \leq \vec{y}(i) \text{ für alle } i \in \{1, \ldots, n\}, \\ \vec{x} <^n \vec{y} & \text{genau dann, wenn} & \vec{x} \leq \vec{y} \text{ und } \vec{x} \neq \vec{y}, \\ \vec{x} \ll \vec{y} & \text{genau dann, wenn} & \vec{x}(i) < \vec{y}(i) \text{ für alle } i \in \{1, \ldots, n\}, \\ \vec{x} <_{\text{lex}} \vec{y} & \text{genau dann, wenn} & \text{ein } i \in \{1, \ldots, n\} \text{ existiert mit} \\ & & \vec{x}(i) < \vec{y}(i) \text{ und } \vec{x}(j) = \vec{y}(j) \text{ für alle } j < i. \end{array}$$

Der Null-Vektor aus $\mathbb{R}^n$ ist 0^n und entsprechend wird $x^n = (x \ldots x) \in \mathbb{R}^n$ abgekürzt. Die Einheitsvektoren des $\mathbb{N}^n$ werden mit $\vec{e}_1, \ldots \vec{e}_n$ bezeichnet.

Für eine $m \times n$-Matrix A ist A^i die i-te Spalte, d. h. ein Vektor aus $\mathbb{R}^m$. Eine $n \times n$-Matrix A ist **unimodular**, wenn $|A| = \pm 1$, wobei $|A|$ die Determinante von A ist. Für eine $m \times n$-Matrix A mit ganzzahligen Koeffizienten und einen Vektor $\vec{b} \in \mathbb{Z}^m$ ist $A\vec{x} = \vec{b}$ mit den Unbekannten $\vec{x}$ mit Lösungen aus $\mathbb{N}^n$ ein **diophantisches Gleichungssystem**. Es ist **homogen**, wenn $\vec{b} = \vec{0}^m$ und **inhomogen** sonst.

1.4 Unifikationsprobleme

Die im folgenden definierte Struktur, in der Unifikationsprobleme repräsentiert werden, wurde von A. Boudet [3] übernommen. Solche Strukturen werden zur Darstellung der Daten in Unifikationsalgorithmen verwendet. Sie repräsentieren die für das Ergebnis wesentlichen Variablen und können eine Menge von Gleichungen beinhalten, wenn eine einzelne Gleichungen zur Datendarstellung in einem Algorithmus nicht ausreicht. Man beachte aber, daß sich der in Definition 1.8 definierte Unifikationstyp für bestimmte Theorien bei der Unifikation von mehreren Gleichungen im Vergleich zur Unifikation einer einzelnen Gleichung ändern kann. Für die AC-Theorie ist dies nicht der Fall.

Definition 1.1 Ein **Unifikationsproblem** ist eine Formel der Prädikatenlogik erster Ordnung, die nur ein binäres Prädikat $\stackrel{?}{=}$ enthält und nur aus Konjunktionen und Existenzquantoren aufgebaut ist.

- T ist das triviale Unifikationsproblem.
- F ist das Unifikationsproblem ohne Lösungen.
- Eine Gleichung $s \stackrel{?}{=} t$ ist ein atomares Unifikationsproblem, wenn $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$.
- Sind P_1 und P_2 Unifikationsprobleme, so ist $P_1 \wedge P_2$ ein Unifikationsproblem.
- Ist P ein Unifikationsproblem und $y \in \mathcal{X}$ eine Variable, so ist $(\exists y)P$ ein Unifikationsproblem. ■

Für eine Gleichheitstheorie E sind die E-Lösungen eines Unifikationsproblems P die Substitutionen, für die P als prädikatenlogische Formel wahr ist, wenn $\stackrel{?}{=}$ als $=_E$ interpretiert wird. Die Unifikationsprobleme beinhalten also über die reine Datenrepräsentation hinaus auch logisch die Aufgabenstellung.

Definition 1.2 E**-Lösungen**

- Für das Unifikationsproblem T sind alle Substitutionen σ E-Lösungen.
- Das Unifikationsproblem F hat keine E-Lösung.
- Die E-Lösungen von $s \stackrel{?}{=} t$ sind alle Substitutionen σ mit $\sigma(s) =_E \sigma(t)$.
- Die E-Lösungen von $P_1 \wedge P_2$ sind alle Substitutionen, die für P_1 und P_2 E-Lösungen sind.
- Die E-Lösungen von $(\exists y)P$ sind die Substitutionen σ, für die ein Term t existiert, so daß σ eine E-Lösung von $\{y \mapsto t\}(P)$ ist.

Zwei Unifikationsprobleme heißen **äquivalent**, wenn sie dieselben E-Lösungen haben. Zwei Terme s und t sind **unifizierbar**, wenn die Menge der E-Lösungen zu $s \stackrel{?}{=} t$ nicht leer ist. Eine Menge von Termen $\{t_1, \ldots, t_n\}$ heißt **unifizierbar**, wenn die Menge der E-Lösungen zu $t_1 \stackrel{?}{=} t_2 \wedge \ldots \wedge t_1 \stackrel{?}{=} t_n$ nicht leer ist. ■

Ein Unifikationsproblem der Form

$$(\exists y_1) \ldots (\exists y_q)P,$$

in der P keine Quantoren mehr enthält, ist in **Pränexform** und wird mit

$$(\exists y_1, \ldots, y_q)P$$

abgekürzt. Für Unifikationsprobleme P in Pränexform heißen $\{y_1, \ldots, y_q\}$ die **quantifizierten Variablen** von P. Die restlichen Variablen in P sind die **freien Variablen** $Var(P)$. Zu jedem Unifikationsproblem läßt sich eine äquivalente Pränexform durch Umbenennung der quantifizierten Variablen und Voranstellung der Quantoren erreichen. Daher sollen zur Vereinfachung im folgenden Unifikationsprobleme immer in Pränexform sein. Außerdem werden beim Vergleich und der Transformation von Unifikationsproblemen implizit die folgenden Eigenschaften benutzt.

- Die Symmetrie von $\stackrel{?}{=}$, d. h. $s \stackrel{?}{=} t$ ist gleichbedeutend zu $t \stackrel{?}{=} s$.
- Die Assoziativität, Kommutativität und Idempotenz von $\wedge$, d. h. $P_1 \wedge (P_2 \wedge P_3) = (P_1 \wedge P_2) \wedge P_3$, $P_1 \wedge P_2 = P_2 \wedge P_1$ und $P \wedge P = P$.
- Die Gleichheit von $t \stackrel{?}{=} t$ und T.

- Die logischen Gleichungen $P \wedge T = P$ und $P \wedge F = F$.
- Die Umbenennung von quantifizierten Variablen, d. h. $(\exists y)P = (\exists y')\{y \mapsto y'\}(P)$, wenn y' nicht in P vorkommt.

Durch die Existenzquantoren werden die Variablen quantifiziert, die nicht in den Termen des ursprünglichen Unifikationsproblems enthalten waren und denen die Lösungssubstitutionen keinen Term zuweisen sollen. Alle während der Unifikation neu eingeführten Variablen sind daher existenzquantifiziert.

Zur Berechnung der E-Lösungen eines Unifikationsproblems werden Regeln aufgestellt, mit denen das Unifikationsproblem bearbeitet werden kann. Eine Regel $P \longrightarrow P'$ heißt **korrekt**, wenn alle E-Lösungen von P' auch E-Lösungen von P sind; sie heißt **vollständig**, wenn alle E-Lösungen von P auch E-Lösungen von P' sind. Eine Regel $P \longrightarrow P'$ ist genau dann vollständig und korrekt, wenn P und P' äquivalent sind.

Da es im allgemeinen unendlich viele E-Lösungen zu einem Unifikationsproblem geben kann, beschränkt man sich auf Teilmengen von E-Lösungen.

Definition 1.3 Eine Menge von E-Lösungen $\mathcal{U}$ zu einem Unifikationsproblem P heißt **vollständig**, wenn für jede E-Lösung τ von P ein $\sigma \in \mathcal{U}$ existiert mit $\sigma \leq_E \tau[Var(P)]$. Sie heißt **minimal**, wenn sie vollständig ist und wenn für alle $\sigma, \rho \in \mathcal{U}$ mit $\sigma \neq \rho$ weder $\sigma \leq_E \rho[Var(P)]$ noch $\rho \leq_E \sigma[Var(P)]$ gilt. ■

Gesucht werden bei der Unifikation also vollständige oder besser noch minimale Mengen von E-Lösungen. Die Elemente einer minimalen Menge sind entsprechend der folgenden Definition allgemeinste Unifikatoren.

Definition 1.4 Eine E-Lösung σ von P ist **allgemeinster E-Unifikator** von P, wenn es keine E-Lösung τ für P gibt mit $\tau <_E \sigma[Var(P)]$. ■

Unifikationsprobleme dienen als Datenstruktur des Unifikationsalgorithmus, der aber E-Lösungen, d. h. Substitutionen, als Ergebnis liefern soll. Deshalb wird die Form eines Unifikationsproblems definiert, die einer E-Lösung entspricht und die den Endzustand der Regeltransformation auf Unifikationsproblemen im Algorithmus darstellt.

Definition 1.5 Ein Unifikationsproblem P ist in **gelöster Form**, wenn $P \equiv T$ oder $P \equiv F$ oder $P \equiv (\exists y_1, \ldots, y_q) x_1 \stackrel{?}{=} t_1 \wedge \ldots \wedge x_n \stackrel{?}{=} t_n$, wobei $\{x_1, \ldots, x_n\} \cap \{y_1, \ldots, y_q\} = \emptyset$ und jedes x_i genau einmal in P vorkommt. ■

Lemma 1.2 *Sei $P \equiv (\exists y_1, \ldots, y_q) x_1 \stackrel{?}{=} t_1 \wedge \ldots \wedge x_n \stackrel{?}{=} t_n$ ein Unifikationsproblem in gelöster Form. Dann ist $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ ein allgemeinster E-Unifikator von P und $\{\sigma\}$ ist eine minimale Menge von E-Lösungen von P.*

Beweis: Für jede E-Lösung σ' gibt es nach der Definition von E-Lösungen eine Substitution $\tau = \{y_1 \mapsto s_1, \ldots, y_q \mapsto s_q\}$, die den quantifizierten Variablen Terme zuweist, so daß $\sigma'(\tau(x_i)) =_E \sigma'(\tau(t_i))$. Da nach Definition $x_i \notin \{y_1, \ldots, y_q\}$, gilt außerdem $\tau(x_i) = x_i$. Daraus folgt nun

$$\begin{aligned} \sigma'(x_i) &=_E \sigma'(\tau(x_i)) =_E \sigma'(\tau(t_i)) =_E \sigma' \circ \tau(t_i) \\ &=_E \sigma' \circ \tau(\sigma(x_i)) \quad \text{für alle } x_i \in \{x_1, \ldots, x_n\}. \end{aligned}$$

Für jede andere Variable $z \in Var(P) \setminus \{x_1, \ldots, x_n\}$ gilt $\sigma(z) = z$ und $\tau(z) = z$, weil $Var(P) \cap \{y_1, \ldots, y_q\} = \emptyset$. Daher gilt

$$\sigma'(z) =_E \sigma'(\tau(z)) =_E \sigma' \circ \tau(\sigma(z)) \text{ für alle } z \in Var(P) \setminus \{x_1, \ldots, x_n\}.$$

Aus beiden Gleichungen folgt $\sigma \leq_E \sigma'[Var(P)]$ und damit die Behauptung. □

Um die Komplexität zu verringern, wird im Algorithmus hauptsächlich mit der im folgenden definierten, sequentiell gelösten Form gearbeitet.

Definition 1.6 Ein Unifikationsproblem P ist in **sequentiell gelöster Form**, wenn $P \equiv T$ oder $P \equiv F$ oder $P \equiv (\exists y_1, \ldots, y_q) x_1 \stackrel{?}{=} t_1 \wedge \ldots \wedge x_n \stackrel{?}{=} t_n$, wobei für alle $i, j \in \{1, \ldots, n\}$ gilt:

1. Ist $x_i = x_j$, so ist $i = j$
2. Ist $x_i \in Var(t_j)$, so ist $i > j$
3. Ist $x_i \in \{y_1, \ldots, y_q\}$, so existiert ein $j < i$ mit $x_i \in Var(t_j)$. ■

Die erste Bedingung fordert, daß die Unifikation wirklich gelöst wurde. Die zweite garantiert, daß kein Zyklus vorkommt und die Dritte verhindert das Auftreten von unnötigen neuen Variablen.

Es ist einfach, aus einer sequentiell gelösten Form eine gelöste Form zu machen. Man wendet so lange wie möglich die beiden folgenden Regeln an.

Regel Rep (Termersetzung)
$(\exists y_1, \ldots, y_q) x \stackrel{?}{=} s \wedge P \longrightarrow_{Rep} (\exists y_1, \ldots, y_q) x \stackrel{?}{=} s \wedge \{x \mapsto s\}(P)$,
wenn $x \in Var(P)$ *und* $s \notin \mathcal{X}$ *und* $x \notin Var(s)$ *oder wenn* $x \in Var(P)$ *und* $s \in \mathcal{X}$ *und dabei* $x \in \{y_1, \ldots, y_q\}$ *oder* $s \notin \{y_1, \ldots, y_q\}$.

Der Zusatz, daß x quantifiziert oder s frei ist, verhindert, daß eine freie Variable durch eine quantifizierte ersetzt wird. Die Regel *Rep* ist korrekt und vollständig, weil $=_E$ eine Kongruenzrelation ist.

Da ein Existenzquantor mit seiner Variablen überflüssig ist, wenn die quantifizierte Variable y nur einmal im Unifikationsproblem in einer Gleichung $y \stackrel{?}{=} s$ vorkommt, braucht man die folgende Regel.

Regel EQE (Elimination von Existenzquantoren)
$(\exists y, y_1, \ldots, y_q) y \stackrel{?}{=} s \wedge P \longrightarrow_{EQE} (\exists y_1, \ldots, y_q) P$,
wenn $y \notin Var(P)$ *und* $y \notin Var(s)$.

Die Regel *EQE* ist vollständig wegen der Definition von E-Lösungen für $\wedge$. Sie ist korrekt, weil jede Substitution eine E-Lösung von $(\exists x) x \stackrel{?}{=} s$ mit $x \notin Var(s)$ ist. Beide Regeln erhalten also die Menge der E-Lösungen für das Unifikationsproblem. Die Anwendung beider Regeln auf ein Unifikationsproblem terminiert, weil *EQE* die Anzahl der Gleichungen reduziert und weil *Rep* die Anzahl der Gleichungen nicht erhöht und die Anzahl der Variablen, die mehr als zweimal in den Gleichungen vorkommen, reduziert. Das Resultat der Anwendung ist ein Unifikationsproblem in gelöster Form, weil sonst noch eine der beiden Regeln anwendbar wäre. Diese gelöste Form ist eindeutig, da jede Variable nur durch einen Term ersetzt werden kann und sich die Reihenfolge der Ersetzung auf das Ergebnis nicht auswirkt.

Jeder sequentiell gelösten Form ist also genau eine gelöste Form und ein allgemeinster E-Unifikator zugeordnet.

Beispiel 1.2 Dieses Beispiel zeigt, wie aus einer sequentiell gelösten Form mit den Regeln *Rep* und *EQE* eine gelöste Form hergeleitet wird. Die erste Form der Ableitung ist keine gelöste Form, weil y_1 und y_2 nicht frei sind und z_2 mehr als einmal darin vorkommt; es ist aber eine sequentiell gelöste Form.

$$
\begin{array}{lrlll}
 & (\exists y_1, y_2) & z_1 \stackrel{?}{=} y_1 + z_2 + z_3 & \wedge\, z_2 \stackrel{?}{=} f(y_1) & \wedge\, y_1 \stackrel{?}{=} f(y_2) \wedge y_2 \stackrel{?}{=} z_3 \\
\longrightarrow_{Rep} & (\exists y_1, y_2) & z_1 \stackrel{?}{=} y_1 + z_2 + z_3 & \wedge\, z_2 \stackrel{?}{=} f(y_1) & \wedge\, y_1 \stackrel{?}{=} f(z_3) \wedge y_2 \stackrel{?}{=} z_3 \\
\longrightarrow_{EQE} & (\exists y_1) & z_1 \stackrel{?}{=} y_1 + z_2 + z_3 & \wedge\, z_2 \stackrel{?}{=} f(y_1) & \wedge\, y_1 \stackrel{?}{=} f(z_3) \\
\longrightarrow_{Rep} & (\exists y_1) & z_1 \stackrel{?}{=} f(z_3) + z_2 + z_3 & \wedge\, z_2 \stackrel{?}{=} f(f(z_3)) & \wedge\, y_1 \stackrel{?}{=} f(z_3) \\
\longrightarrow_{EQE} & & z_1 \stackrel{?}{=} f(z_3) + z_2 + z_3 & \wedge\, z_2 \stackrel{?}{=} f(f(z_3)) & \\
\longrightarrow_{Rep} & & z_1 \stackrel{?}{=} f(z_3) + f(f(z_3)) + z_3 & \wedge\, z_2 \stackrel{?}{=} f(f(z_3)) &
\end{array}
$$

Auf diese letzte Form der Ableitung lassen sich die Regeln *Rep* und *EQE* nicht mehr anwenden und es handelt sich um eine gelöste Form. Die Substitution dazu ist $\{z_1 \mapsto f(z_3) + f(f(z_3)) + z_3, z_2 \mapsto f(f(z_3))\}$. Man beachte, daß man im ersten Schritt der Ableitung die Regel *Rep* nur so anwenden darf, daß y_2 durch z_3 ersetzt wird. Umgekehrt, z_3 durch y_2 zu ersetzen, ist nicht möglich, da z_3 frei ist und y_2 nicht. ■

Definition 1.7 Eine Menge von (sequentiell) gelösten Formen für ein Unifikationsproblem P heißt **vollständige (bzw. minimale) Menge von gelösten Formen** oder **sequentiell gelösten Formen**, wenn die zugehörigen allgemeinsten E-Unifikatoren eine vollständige (bzw. minimale) Menge von E-Lösungen für P ist.

Ein Unifikationsproblem P' ist eine (sequentiell) gelöste Form von P, wenn es ein Element einer vollständigen Menge von (sequentiell) gelösten Formen von P ist. ■

In der Unifikationshierarchie nach Siekmann [22] klassifiziert man Theorien danach, wie groß eine minimale Menge von (sequentiell) gelösten Formen bzw. E-Lösungen werden kann und ob so eine Menge überhaupt existiert.

Definition 1.8 Der **Unifikationstyp** einer Theorie ist

unitär, wenn zu jedem Unifikationsproblem dieser Theorie eine minimale Menge von E-Lösungen existiert, die höchstens ein Element hat,

finitär, wenn immer eine solche Menge existiert, die endlich ist,

infinitär, wenn immer eine solche Menge existiert und

null, wenn es ein Unifikationsproblem dieser Theorie gibt, für das keine minimale Menge von E-Lösungen existiert. ■

Im allgemeinen ist der Unifikationstyp einer Theorie unentscheidbar (siehe [17]), aber in vielen Fällen kann man ihn bestimmen. Die Theorien $E_0 \cup AC_1 \cup \ldots \cup AC_n$ und AC sind finitär und nicht unitär. Dies wird in Kapitel 2 und 3 deutlich. Für die Unifikationsprobleme mit Termen aus $\mathcal{T}(\mathcal{AC} \cup \mathcal{F}_0, \mathcal{X})$ oder aus $\mathcal{T}(\{+\}, \mathcal{X})$ mit $+ \in \mathcal{AC}$ gibt es also immer eine endliche minimale Menge von sequentiell gelösten Formen und diese Menge kann mehr als ein Element enthalten.

Kapitel 2

Allgemeine AC-Unifikation

2.1 Einführung

Unter **allgemeiner E-Unifikation** versteht man das Lösen von Unifikationsproblemen, die aus den freien Funktionssymbolen und den Funktionssymbolen einer oder mehrerer Instanzen von E aufgebaut sind. **Elementare E-Unifikation** hingegen ist das Lösen von Unifikationsproblemen, die nur Funktionssymbole der Theorie E enthalten. Von Bedeutung ist auch noch die **elementare E-Unifikation mit Konstanten**, bei der in den Unifikationsproblemen auch noch freie Konstanten erlaubt sind.

In diesem Kapitel soll ein Algorithmus zur allgemeinen AC-Unifikation, d. h. der Unifikation von Termen aus $\mathcal{T}(\mathcal{AC} \cup \mathcal{F}_0, \mathcal{X})$ vorgestellt werden. Dabei treten, wie bei jeder Unifikation modulo einer Theorie $E = E_0 \mathbin{\mathring{\cup}} \ldots \mathbin{\mathring{\cup}} E_n$, zwei Hauptprobleme auf.

1. Die elementare E_h-Unifikation, d. h. die Unifikation von Termen aus $\mathcal{T}(\mathcal{F}_h, \mathcal{X})$ modulo E_h.
2. Die Kombination der verschiedenen elementaren Unifikationsalgorithmen für die Teiltheorien $E_0, \ldots, E_n$.

Der erste Punkt wird für die AC-Theorie in Kapitel 3 behandelt. Zum zweiten Punkt wird in Abschnitt 2.2 das Kombinationsverfahren von A. Boudet [3] vorgestellt, das für die Kombination regulärer, kollapsfreier Theorien geeignet ist. Anschließend werden dann für die allgemeine AC-Unifikation spezielle Bedingungen aufgestellt, aus denen sich ergibt, daß ein wesentlicher Bestandteil des Kombinationsverfahrens, die Variablenabstraktion, nicht mehr explizit benötigt wird. Sei $E = E_0 \mathbin{\mathring{\cup}} \ldots \mathbin{\mathring{\cup}} E_n$ also die Vereinigung regulärer, kollapsfreier Theorien (im hier betrachteten Fall die freie Theorie und mehrere AC-Theorien).

Definition 2.1 Eine Gleichung $s \stackrel{?}{=} t$ heißt **variabel**, wenn $s, t \in \mathcal{X}$; sie heißt **echt**, wenn $s, t \notin \mathcal{X}$. Ein Term t ist **pur** in einer Theorie E_h, wenn $t \in \mathcal{T}(\mathcal{F}_h, \mathcal{X})$ und eine Gleichung $s \stackrel{?}{=} t$ ist **pur** in E_h, wenn s und t pur in E_h sind. Gibt es für einen Term t (oder eine Gleichung $s \stackrel{?}{=} t$) kein h mit t (bzw. $s \stackrel{?}{=} t$) pur in E_h, so heißt t (bzw. $s \stackrel{?}{=} t$) **heterogen**. ■

Ein Unifikationsproblem wird für den Kombinationsalgorithmus als

$$P \equiv (\exists y_1, \ldots, y_q) P_V \ \wedge \ P_H \ \wedge \ P_0 \ \wedge \ P_1 \ \wedge \ \ldots \ \wedge \ P_n$$

dargestellt, wobei

P_V die Konjunktion der variablen Gleichungen,

P_H die Konjunktion der heterogenen Gleichungen und

P_h für $h \in \{0, \ldots, n\}$ die Konjunktion der nicht-variablen Gleichungen ist, die pur in der Theorie E_h sind.

Die P_h werden zusätzlich noch in $P_h = P_h^u \wedge P_h^e$, die unechten und echten, nicht-variablen Gleichungen aufgeteilt.

2.2 Der Kombinationsalgorithmus für reguläre, kollapsfreie Theorien

Der Kombinationsalgorithmus startet mit einem beliebigen Unifikationsproblem und liefert eine vollständige Menge von sequentiell gelösten Formen. Er besteht aus Regeln, die in beliebiger Reihenfolge angewendet werden können. Lediglich bei der Anwendung der Regel *E-Res* ist darauf zu achten, daß alle Möglichkeiten dieser Regel angewendet werden.

Die erste Regel sorgt dafür, daß die heterogenen Gleichungen in pure überführt werden; dies geschieht durch Variablenabstraktion.

Regel VA (Variablenabstraktion)
$s[s_1]_p \stackrel{?}{=} t \longrightarrow_{VA} (\exists y) s[p \leftarrow y] \stackrel{?}{=} t \ \wedge \ y \stackrel{?}{=} s_1$,
wenn s_1 *ein Fremdterm in* $s[s_1]_p$ *und* y *eine neue Variable ist.*

Die durch diese Regel neu eingeführten Variablen y heißen **Abstraktionsvariablen**.

Durch sukzessive Variablenabstraktion werden pure Terme erzeugt. Die Gleichungen, in denen beide Terme pur in derselben Theorie E_i geworden sind, werden in P_i gesammelt. Ein pures Teilproblem P_i kann dann mit der folgenden Regel unter Benutzung eines Algorithmus zur elementaren E_i-Unifikation gelöst werden.

Regel *E*-Res
$P_h \longrightarrow_{E\text{-}Res} P_h'$,
wenn P_h *nicht in sequentiell gelöster Form vorliegt und* P_h' *eine sequentiell gelöste Form von* P_h *ist.*

Diese Regel ist offensichtlich korrekt, d. h. das neu erzeugte Unifikationsproblem hat keine E-Lösungen, die das ursprüngliche nicht hatte; es können aber Lösungen fehlen, was nicht überrascht, da E_h nicht unitär sein muß (AC ist finitär) und eine sequentiell gelöste Form genau einer E-Lösung entspricht. Damit diese Regel vollständig ist, ist sie daher immer mit allen Elementen einer vollständigen Menge von sequentiell gelösten Formen für P_h parallel durchzuführen, d. h. es findet eine Aufspaltung der Ableitung statt.

In einer kollapsfreien Theorie kann eine Gleichung $s \stackrel{?}{=} t$ mit nicht-variablen Termen s und t keine Lösung haben, wenn s und t Kopfsymbole aus verschiedenen Theorien haben. Daher kann man die folgenden Regeln anwenden.

Regel Conflict 1
$s \stackrel{?}{=} t \longrightarrow_{Conflict1} F$,
wenn $s(\epsilon) \in \mathcal{F}_i$ *und* $t(\epsilon) \in \mathcal{F}_j$ *und* $i \neq j$.

Regel Conflict 2
$x \stackrel{?}{=} s \ \wedge \ x \stackrel{?}{=} t \longrightarrow_{Conflict2} F$,
wenn $s(\epsilon) \in \mathcal{F}_i$ *und* $t(\epsilon) \in \mathcal{F}_j$ *und* $i \neq j$.

Der normale Occur-Check der freien Theorie gilt für reguläre, kollapsfreie Theorien im Allgemeinen nicht, da z. B. das Unifikationsproblem $x \stackrel{?}{=} f(x)$ für die Theorie $E = \{a = f(a)\}$ die Lösung $\{x \mapsto a\}$ hat. Es gilt aber eine schwächere Fassung, die zusammengesetzte Zyklen mit Operatoren aus verschiedenen Theorien verbietet:

Regel Check*
$P \longrightarrow_{Check^*} F$,
wenn P *einen zusammengesetzten Zyklus* $x_1 \stackrel{?}{=} s_1[x_2] \ \wedge \ x_2 \stackrel{?}{=} s_2[x_3] \ \wedge \ \ldots \ \wedge \ x_n \stackrel{?}{=} s_n[x_1]$ *besitzt, für den* $i, j \in \{1, \ldots, n\}$ *existieren mit* $s_i(\epsilon) \in \mathcal{F}_k$ *und* $s_j(\epsilon) \in \mathcal{F}_l$ *und* $k \neq l$.

Da es sich im hier behandelten Fall, der Kombination von AC-Theorien und der freien Theorie, nur um simple Theorien handelt, kann der Occur-Check wieder verschärft werden und eine Gleichung $x \stackrel{?}{=} s$ mit $x \in Var(s)$ sofort als unlösbar erkannt werden, weil es nach der Definition einer simplen Theorie keinen Term t geben kann, für den $\{x \mapsto t\}x =_E \{x \mapsto t\}s$, also $t =_E s[t]$ gilt. Die Berechnung kann also abgebrochen werden, wenn ein Zyklus $x_1 \stackrel{?}{=} s_1[x_2] \ \wedge \ \ldots \ \wedge \ x_n \stackrel{?}{=} s_n[x_1]$ mit einem $s_i \notin \mathcal{X}$ existiert.

Da der Algorithmus nur eine sequentiell gelöste Form berechnen soll, kann auf eine Ersetzung von Variablen durch Terme (wie in *Rep*) verzichtet werden. Dieser Verzicht ist für die Terminierung des Algorithmus notwendig. Benötigt wird lediglich eine Ersetzung durch Variablen.

Regel Var-Rep

$(\exists y_1, \ldots, y_q) x \stackrel{?}{=} y \ \wedge \ P \longrightarrow_{\mathit{Var\text{-}Rep}} (\exists y_1, \ldots, y_q) x \stackrel{?}{=} y \ \wedge \ \{x \mapsto y\}(P)$,
wenn $x, y \in Var(P)$ *und* x *ist existenzquantifiziert oder* y *ist frei.*

Auch hier darf wie bei der Regel *Rep* eine freie Variable nicht durch eine quantifizierte ersetzt werden.

Zu den in diesem Kapitel vorgestellten Regeln kommt noch die Regel *EQE* aus Kapitel 1. Die Menge der Regeln $S = \{\mathit{VA}, \mathit{E\text{-}Res}, \mathit{Conflict1}, \mathit{Conflict2}, \mathit{Check}^*, \mathit{Var\text{-}Rep}, \mathit{EQE}\}$ erhält die Menge der Lösungen für ein Unifikationsproblem, terminiert für jede Eingabe und liefert eine Menge von sequentiell gelösten Formen. Für Beweise siehe [2, Vollständigkeit: S. 45 ff.; Terminierung: S. 47 ff.].

Beispiel 2.1 In diesem Beispiel soll gezeigt werden, wie der Kombinationsalgorithmus die Terme $z_1 + f(z_3)$ und $z_2 + f(z_1)$ unifiziert. Dabei ist $+ \in \mathcal{F}_1$ ein AC-Symbol, $a, f \in \mathcal{F}_0$ sind freie Symbole und $z_i, v_i, u_i \in \mathcal{X}$ Variablen.

$$\begin{array}{lll} P \equiv & & z_1 + f(z_3) \stackrel{?}{=} z_2 + f(z_1) \\ \stackrel{2}{\longrightarrow}_{\mathit{VA}} & (\exists v_1, v_2) & z_1 + v_1 \stackrel{?}{=} z_2 + v_2 \quad \wedge v_1 \stackrel{?}{=} f(z_3) \ \wedge \ v_2 \stackrel{?}{=} f(z_1) \end{array}$$

Nun kann auf $z_1 + v_1 \stackrel{?}{=} z_2 + v_2$ die Regel *AC-Res* angewendet werden. Das Ergebnis sind sieben gelöste Formen, die alle bearbeitet werden müssen. Hier soll dies für drei von diesen Formen gezeigt werden.

1. $\longrightarrow_{\mathit{AC\text{-}Res}} \quad (\exists v_1, v_2, u_1, u_2, u_3) \quad z_1 \stackrel{?}{=} u_1 \quad \wedge v_1 \stackrel{?}{=} u_2 + u_3 \wedge z_2 \stackrel{?}{=} u_1 + u_2 \wedge v_2 \stackrel{?}{=} u_3$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge \quad v_1 \stackrel{?}{=} f(z_3) \wedge v_2 \stackrel{?}{=} f(z_1)$
$\longrightarrow_{\mathit{Conflict2}} \quad F$

Die Regel *Conflict2* konnte auf die beiden Gleichungen $v_1 \stackrel{?}{=} u_2 + u_3$ und $v_1 \stackrel{?}{=} f(z_3)$ angewendet werden.

2. $\longrightarrow_{\mathit{AC\text{-}Res}} \quad (\exists v_1, v_2, u_1, u_2, u_3) \quad z_1 \stackrel{?}{=} u_1 + u_2 \wedge v_1 \stackrel{?}{=} u_3 \quad \wedge z_2 \stackrel{?}{=} u_1 + u_3 \wedge v_2 \stackrel{?}{=} u_2$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge \quad v_1 \stackrel{?}{=} f(z_3) \quad \wedge v_2 \stackrel{?}{=} f(z_1)$
$\longrightarrow_{\mathit{Check}^*} \quad F$

In diesem Fall entsteht der zusammengesetzte Zyklus $z_1 \stackrel{?}{=} u_1 + u_2 \wedge u_2 \stackrel{?}{=} v_2 \wedge v_2 \stackrel{?}{=} f(z_1)$, so daß die Regel *Check** angewendet werden kann.

3. $\begin{array}{lll} \longrightarrow_{\mathit{AC\text{-}Res}} & (\exists v_1, v_2, u_1, u_2) & z_1 \stackrel{?}{=} u_1 \wedge v_1 \stackrel{?}{=} u_2 \quad \wedge v_1 \stackrel{?}{=} f(z_3) \wedge v_2 \stackrel{?}{=} f(z_1) \\ \stackrel{3}{\longrightarrow}_{\mathit{Var\text{-}Rep}} & (\exists v_1, v_2, u_1, u_2) & z_1 \stackrel{?}{=} u_1 \wedge v_1 \stackrel{?}{=} u_2 \quad \wedge z_2 \stackrel{?}{=} z_1 \quad \wedge v_2 \stackrel{?}{=} u_2 \\ & & \qquad\qquad\qquad \wedge u_2 \stackrel{?}{=} f(z_3) \qquad\qquad \wedge u_2 \stackrel{?}{=} f(z_1) \\ \stackrel{3}{\longrightarrow}_{\mathit{EQE}} & (\exists u_2) & z_2 \stackrel{?}{=} z_1 \wedge u_2 \stackrel{?}{=} f(z_3) \qquad\qquad \wedge u_2 \stackrel{?}{=} f(z_1) \\ \longrightarrow_{\mathit{E_0\text{-}Res}} & (\exists u_2) & z_2 \stackrel{?}{=} z_1 \wedge z_3 \stackrel{?}{=} z_1 \qquad\qquad \wedge u_2 \stackrel{?}{=} f(z_1) \\ \longrightarrow_{\mathit{EQE}} & & z_2 \stackrel{?}{=} z_1 \wedge z_3 \stackrel{?}{=} z_1 \end{array}$

Dies ist der einzige Fall aus den sieben gelösten Formen der AC-Unifikation, der nicht abbricht und zu einer Lösung führt. Im ersten Schritt werden durch dreimaliges Ausführen der Regel *Var-Rep* die Substitutionen $\{u_1 \mapsto z_1\}$, $\{v_1 \mapsto u_2\}$ und $\{v_2 \mapsto u_2\}$ auf den jeweiligen Rest des Unifikationsproblems angewendet. Anschließend werden die so überflüssig gewordenen drei Variablen u_1, v_1 und v_2 mit der Regel *EQE* entfernt. Danach wird das

in der freien Theorie neu entstandene Teilproblem mit der Regel E_0-*Res* gelöst und zum Schluß die danach überflüssige Variable u_2 gelöscht. Da das Ergebnis schon eine gelöste Form ist, kann die Substitution sofort ohne eine Transformation mit den Regeln *Rep* und *EQE* gebildet werden. Das Ergebnis, das der Kombinationsalgorithmus liefert, ist also $\{z_2 \mapsto z_1, z_3 \mapsto z_1\}$. ■

Das folgende Beispiel zeigt, daß der Algorithmus nur für reguläre, kollapsfreie Theorien geeignet ist.

Beispiel 2.2 Das Unifikationsproblem $P = x \stackrel{?}{=} +(y,z) \ \wedge \ y \stackrel{?}{=} f(x)$ mit $\mathcal{F}_0 = \{f\}$ und $\mathcal{F}_1 = \{+,0\}$ kann mit der Regel *Check** abgebrochen werden. Für die nicht reguläre Theorie $E_1 = \{+(x,x) = 0\}$ existiert aber die Lösung $\{x \mapsto 0, y \mapsto f(0), z \mapsto f(0)\}$.

Das Unifikationsproblem $P = f(x) \stackrel{?}{=} +(y,z)$ mit $\mathcal{F}_0 = \{f\}$ und $\mathcal{F}_1 = \{+,0\}$ kann mit der Regel *Conflict1* zum Abbruch geführt werden. Es hat für die nicht kollapsfreie Theorie $E_1 = \{+(x,0) = x\}$ aber die Lösung $\{y \mapsto f(x), z \mapsto 0\}$. ■

2.3 Bedingungen für elementare AC-Unifikation

Für die Variablen eines puren Teilproblems (insbesondere für die Abstraktionsvariablen) gelten im gesamten Unifikationsproblem bestimmte Bedingungen, die den Suchraum für die Lösungen beschränken. Deshalb ist es sinnvoll, diese Bedingungen bei der Lösung eines solchen Teilproblems durch elementare AC-Unifikation mit der Regel *E-Res* einzubeziehen. So führt es z. B. zu einem ergebnislosen Abbruch des Berechnungspfades, wenn einer Variablen, der im ursprünglichen Unifikationsproblem schon ein Term zugewiesen wurde, mit der Regel *E-Res* ein neuer Term zugewiesen wird, der mit dem ersten Term nicht unifizierbar ist. Solche Bedingungen gelten in ähnlicher Weise auch für andere Theorien, sind aber wegen der speziellen Form des Ergebnisses der elementaren AC-Unifikation für allgemeine AC-Unifikation besonders wirkungsvoll und sollen im folgenden für diese Theorie vorgestellt werden. Sei $E = AC_1 \cup \ldots \cup AC_n$ also eine Kombination von AC-Theorien mit den AC-Symbolen $\mathcal{F}_i = \{+_i\}$. Außerdem enthalte die Signatur weiterhin die freien Funktionssymbole $\mathcal{F}_0$.

Ein Unifikationsproblem habe die Form

$$P^0 \equiv (\exists y_1, \ldots, y_q) P_V \ \wedge \ P_0 \ \wedge \ P_1 \ \wedge \ \ldots \ \wedge \ P_n.$$

Dabei sind

P_h für $h \in \{0, \ldots, n\}$ die Gleichungen $s \stackrel{?}{=} t$ mit $s(\epsilon), t(\epsilon) \in \mathcal{F}_h \cup \mathcal{X}$ und $s \notin \mathcal{X}$ oder $t \notin \mathcal{X}$ und

P_V die Gleichungen $s \stackrel{?}{=} t$ mit zwei Variablen $s, t \in \mathcal{X}$.

Im Gegensatz zu vorher werden in den P_h nun nicht mehr die puren Gleichungen gesammelt, sondern die Gleichungen werden nach der Theorie des Kopfsymbols der beiden Terme sortiert. Ein Unifikationsproblem in einer anderen Form, d. h. mit einer Gleichung, bei der die Kopfsymbole der beiden Terme zu verschiedenen Theorien gehören, kann mit der Regel *Conflict1* sofort als unlösbar erkannt werden.

Als erstes sollten immer so weit wie möglich die Regeln *Var-Rep* und *EQE* auf die Gleichungen in P_V angewandt werden. Ist dies geschehen, sollte als nächstes das Unifikationsproblem P_0 gelöst werden, weil dies von geringerer Komplexität ist, als das Lösen eines AC-Problems. Sind diese beiden Schritte bereits durchgeführt, kann ein Teilproblem P_h mit $h > 0$ ausgesucht werden, das als nächstes mit elementarer AC-Unifikation gelöst werden soll.

Zuerst wird für die Fremdterme in den Termen von P_h mit der Regel *VA* Variablenabstraktion durchgeführt, so daß ein Unifikationsproblem

$$\begin{aligned} P^1 &\equiv (\exists y_1, \ldots, y_q, v_1, \ldots, v_l) P_h^p \ \wedge \ P_h^a \ \wedge \ P' \quad \text{mit} \\ P' &\equiv P_V \ \wedge \ P_0 \ \wedge \ \ldots \ \wedge \ P_{h-1} \ \wedge \ P_{h+1} \ \wedge \ \ldots \ \wedge \ P_n \end{aligned}$$

entsteht. Dabei sind P_h^a die neu entstanden Gleichungen der Form $v_i \stackrel{?}{=} r_i$ mit den Abstraktionsvariablen $\mathcal{V} = \{v_1, \ldots, v_l\}$, und P_h^p ist ein pures Unifikationsproblem in AC_h, das mit der Regel *E-Res* gelöst werden kann. Seien $\mathcal{Z} = \{z_1, \ldots, z_k\} = Var(P_h^p) \setminus \mathcal{V} \subseteq Arg^{+_h}(P_h^p)$ die restlichen Variablen in P_h^p. Eine gelöste Form zu P_h^p, wie sie der Algorithmus in Kapitel 3 liefert, hat die Gestalt

$$P_h^s \equiv (\exists u_1, \ldots, u_p) z_1 \stackrel{?}{=} s_1 \ \wedge \ \ldots \ \wedge \ z_k \stackrel{?}{=} s_k \ \wedge \ v_1 \stackrel{?}{=} t_1 \ \wedge \ \ldots \ \wedge \ v_l \stackrel{?}{=} t_l$$

mit $s_i, t_i \in \mathcal{T}(\{+_h\}, \mathcal{U})$ und den neuen Variablen $\mathcal{U} = \{u_1, \ldots, u_p\}$, d. h. jeder Variablen des Unifikationsproblems wird eine neue Variable oder ein AC-Term mit neuen Variablen aus $\mathcal{U}$ als Argumente zugewiesen. Das gesamte Unifikationsproblem hat dann die Form

$$P^2 \equiv (\exists y_1, \ldots, y_q, v_1, \ldots, v_l, u_1, \ldots, u_p) P_h^s \ \wedge \ P_h^a \ \wedge \ P'.$$

Dadurch, daß P_h^s nun im Kontext von P_h^a und P' weiter behandelt werden muß, ergeben sich für die gelösten Formen P_h^s Bedingungen, damit sie zu einer Lösung führen können. In den nächsten Unterabschnitten werden diese Bedingungen aufgezählt und am folgenden Beispiel verdeutlicht.

Beispiel 2.3 Das Unifikationsproblem, das im Rest des Kapitels als Beispiel dienen soll, ist $P^0 \equiv z_1 + f(a) \stackrel{?}{=} z_2 + z_3 + g(b) \ \wedge \ z_3 \stackrel{?}{=} f(z_4)$. Dabei ist $+ \in \mathcal{F}_1$ ein AC-Symbol, $a, b, f, g \in \mathcal{F}_0$ sind freie Symbole und $z_i, v_i, u_i \in \mathcal{X}$ Variablen. Durch zweimaliges Anwenden der Regel *VA* erhält man

$$P^1 \equiv (\exists v_1, v_2, v_3) z_1 + v_1 \stackrel{?}{=} z_2 + z_3 + v_2 \ \wedge \ z_3 \stackrel{?}{=} f(z_4) \ \wedge \ v_1 \stackrel{?}{=} f(a) \ \wedge \ v_2 \stackrel{?}{=} g(b).$$

Die vollständige Menge von sequentiell gelösten Formen für das in AC^+ pure Unifikationsproblem $P_+^p \equiv z_1 + v_1 \stackrel{?}{=} z_2 + z_3 + v_2$ hat 25 Elemente. Eins dieser Elemente ist

$$P_+^{s_1} \equiv (\exists u_1, u_2, u_3) z_1 \stackrel{?}{=} u_1 \ \wedge \ v_1 \stackrel{?}{=} u_2 + u_3 \ \wedge \ z_2 \stackrel{?}{=} u_1 \ \wedge \ z_3 \stackrel{?}{=} u_2 \ \wedge \ v_3 \stackrel{?}{=} u_3. \quad \blacksquare$$

2.3.1 Bedingungen für Abstraktionsvariablen

Die Abstraktionsvariablen v_i sind in P^2 existenzquantifiziert und kommen genau zweimal darin vor; einmal in der Gleichung $v_i = t_i$ und einmal in der Gleichung $v_i \stackrel{?}{=} r_i$. Da $v_i \stackrel{?}{=} r_i$ durch Variablenabstraktion entstanden ist, d. h. r_i in einem Term mit $+_h$ als Kopfsymbol ein Fremdterm war, gilt $r_i(\epsilon) \neq +_h$. Wird einer Abstraktionsvariablen v_i ein Term t_i mit dem AC-Kopfsymbol $+_h$ zugewiesen, hat dieser Lösungszweig keine E-Lösung, da die Regel *Conflict2* auf $v_i \stackrel{?}{=} t_i$ und $v_i \stackrel{?}{=} r_i$ anwendbar ist. Den Abstraktionsvariablen dürfen also nur Variablen zugewiesen werden.

Bedingung 2.1
$v \stackrel{?}{=} t \Rightarrow F$,
wenn v eine Abstraktionsvariable ist und $t \notin \mathcal{X}$.

Beispiel 2.4 Im obigen Beispiel fällt die Lösung $P_+^{s_1}$ weg, weil der Abstraktionsvariablen v_1 ein AC-Term zugewiesen wird (auf die Gleichungen $v_1 \stackrel{?}{=} u_2 + u_3$ und $v_1 \stackrel{?}{=} f(a)$ kann die Regel *Conflict2* angewendet werden). ■

Aus dieser Bedingung, die für die gelösten Formen purer AC-Probleme nur eine bestimmte Gestalt zuläßt, ergeben sich Konsequenzen für das gesamte Unifikationsproblem. Eine gelöste Form kann also nur die Gestalt

$$P_h^s \equiv (\exists u_1, \ldots, u_p) z_1 \stackrel{?}{=} s_1 \ \wedge \ \ldots \ \wedge \ z_k \stackrel{?}{=} s_k \ \wedge \ v_1 \stackrel{?}{=} u_{j_1} \ \wedge \ \ldots \ \wedge \ v_l \stackrel{?}{=} u_{j_l}$$

mit $s_i \in \mathcal{T}(\{+_h\}, \mathcal{U})$ und $u_{j_i} \in \mathcal{U} = \{u_1, \ldots, u_p\}$ haben. Da den Abstraktionsvariablen v_i nach Bedingung 2.1 nur Variablen zugewiesen werden dürfen, kann man nun im gesamten Unifikationsproblem für die v_i l-mal die Regeln *Var-Rep* anwenden und so jedes v_i in den durch Variablenabstraktion entstandenen Gleichungen $v_i \stackrel{?}{=} r_i$ durch u_{j_i} ersetzen. Danach kommt jedes v_i nur noch einmal in der Gleichung $v_i \stackrel{?}{=} u_{j_i}$ im Unifikationsproblem vor. Deshalb kann man durch l-maliges Anwenden von *EQE* alle Abstraktionsvariablen in $\mathcal{V}$ aus dem Unifikationsproblem entfernen. Das gesamte Unifikationsproblem hat dann die Form

$$P^3 \equiv (\exists u_1, \ldots, u_p, y_1, \ldots, y_q) z_1 \stackrel{?}{=} s_1 \ \wedge \ \ldots \ \wedge \ z_k \stackrel{?}{=} s_k \ \wedge \ u_{i_1} \stackrel{?}{=} r_1 \ \wedge \ \ldots \ \wedge \ u_{i_l} \stackrel{?}{=} r_l \ \wedge \ P'.$$

Nun noch einmal die Schritte im Überblick.

$$
\begin{array}{lllll}
 & P_h^0 & \equiv & & P_h \\
\xrightarrow{l}_{VA} & P_h^1 & \equiv & (\exists v_1,\ldots,v_l) & P_h^p \wedge v_1 \stackrel{?}{=} r_1 \wedge \ldots \wedge v_l \stackrel{?}{=} r_l \\
\longrightarrow_{AC_h\text{-}Res} & P_h^2 & \equiv & (\exists v_1,\ldots,v_l,u_1,\ldots,u_p) & z_1 \stackrel{?}{=} s_1 \wedge \ldots \wedge z_k \stackrel{?}{=} s_k \wedge \\
 & & & & v_1 \stackrel{?}{=} u_{i_1} \wedge \ldots \wedge v_l \stackrel{?}{=} u_{i_l} \wedge \\
 & & & & v_1 \stackrel{?}{=} r_1 \wedge \ldots \wedge v_l \stackrel{?}{=} r_l \\
\xrightarrow{l}_{Var\text{-}Rep} & P_h^{2'} & \equiv & (\exists v_1,\ldots,v_l,u_1,\ldots,u_p) & z_1 \stackrel{?}{=} s_1 \wedge \ldots \wedge z_k \stackrel{?}{=} s_k \wedge \\
 & & & & v_1 \stackrel{?}{=} u_{i_1} \wedge \ldots \wedge v_l \stackrel{?}{=} u_{i_l} \wedge \\
 & & & & u_{i_1} \stackrel{?}{=} r_1 \wedge \ldots \wedge u_{i_l} \stackrel{?}{=} r_l \\
\xrightarrow{l}_{EQE} & P_h^3 & \equiv & (\exists u_1,\ldots,u_p) & z_1 \stackrel{?}{=} s_1 \wedge \ldots \wedge z_k \stackrel{?}{=} s_k \wedge \\
 & & & & u_{i_1} \stackrel{?}{=} r_1 \wedge \ldots \wedge u_{i_l} \stackrel{?}{=} r_l
\end{array}
$$

Dieses Schema zeigt, wie durch Ausnutzung von Bedingung 2.1 alle Abstraktionsvariablen, die zur Anwendung der elementaren AC-Unifikation benötigt wurden, wieder verschwinden. Die so aus P_h entstandene Form P_h^3 heißt *AC_{+_h}-gelöste Form* zu P_h. Die dabei durchgeführten Schritte werden in einer neuen Regel zusammengefaßt.

Regel C-AC-Res (Constraint AC-Res)
$P_h \Rightarrow P_h^s$,
wenn $s(\epsilon), t(\epsilon) \in \mathcal{F}_h \cup \mathcal{X}$ *für alle* $s \stackrel{?}{=} t$ *in* P_h *und* P_h^s *eine* AC^{+_h}*-gelöste Form zu* P_h *ist.*

Da diese Regel vollständig aus Regeln aus S zusammengesetzt ist,

$$\longrightarrow_{C\text{-}AC\text{-}Res} = \xrightarrow{l}_{VA} \longrightarrow_{AC\text{-}Res} \xrightarrow{l}_{Var\text{-}Rep} \xrightarrow{l}_{EQE},$$

ist auch die Regelmenge $S \cup \{C\text{-}AC\text{-}Res\}$ korrekt, terminierend und vollständig. Da außerdem *C-AC-Res* immer anwendbar ist, wenn *E-Res* auf eine AC-Theorie anwendbar ist ($l = 0$), kann *E-Res* auf die Regel E_0*-Res* zur Unifikation freier Terme reduziert werden. Die Regelmenge $S' = \{VA, E_0\text{-}Res, C\text{-}AC\text{-}Res, Conflict1, Conflict2, Check^*, Var\text{-}Rep, EQE\}$ erhält also auch die Menge der Lösungen für ein Unifikationsproblem, terminiert für jede Eingabe und liefert eine vollständige Menge von sequentiell gelösten Formen.

Statt eines normalen Algorithmus zur elementaren AC-Unifikation, wird für die Regel *C-AC-Res* nun ein Algorithmus benötigt, der AC-gelöste Formen liefert. Dieser Algorithmus ist in gewisser Weise weiterhin elementar, da die Unifikation nur für ein AC-Symbol durchgeführt wird. Das zu lösende Unifikationsproblem kann aber auch Fremdterme, d. h. Symbole aus anderen Theorien enthalten. Diese Fremdterme werden dann aber nicht untereinander unifiziert, sondern nur über Variablen gleichgesetzt. Die Unifikation der Fremdterme wird dann vom Kombinationsalgorithmus weiter bearbeitet. In Kapitel 3 wird gezeigt, wie sich ein Algorithmus zur elementaren AC-Unifikation einfach zu einem

Algorithmus zur elementaren AC-Unifikation mit Fremdtermen modifizieren läßt. In diesem modifizierten Algorithmus, der durch die Regel *C-AC-Res* aufgerufen wird, werden die Abstraktionsvariablen nicht mehr benötigt, so daß auf ihre Erzeugung ganz verzichtet werden kann. Eine Variablenabstraktion findet implizit aber immer noch statt, was sich darin bemerkbar macht, daß zwei Fremdterme nicht direkt gleichgesetzt werden, sondern nur über eine durch die elementare AC-Unifikation eingeführte Variable.

Beispiel 2.5 Im obigen Beispiel kann die Regel *C-AC-Res* nun direkt auf das Unifikationsproblem $z_1 + f(a) \stackrel{?}{=} z_2 + z_3 + g(b)$ angewendet werden, ohne daß eine Variablenabstraktion nötig ist. Es gibt dann noch 5 Lösungen. Drei dieser Lösungen sind

$$\begin{array}{lcllll}
P_+^{s_2} & \equiv & (\exists u_1,u_2,u_3,u_4) & z_1 \stackrel{?}{=} u_1+u_2+u_3 \wedge f(a) \stackrel{?}{=} u_4 & \wedge & \\
 & & & z_2 \stackrel{?}{=} u_1 \wedge z_3 \stackrel{?}{=} u_2+u_4 & \wedge & g(b) \stackrel{?}{=} u_3 \\
P_+^{s_3} & \equiv & (\exists u_1,u_2,u_3) & z_1 \stackrel{?}{=} u_1+u_2 \wedge f(a) \stackrel{?}{=} u_3 & \wedge & \\
 & & & z_2 \stackrel{?}{=} u_1 \wedge z_3 \stackrel{?}{=} u_2 & \wedge & g(b) \stackrel{?}{=} u_3 \\
P_+^{s_4} & \equiv & (\exists u_1,u_2,u_3) & z_1 \stackrel{?}{=} u_1+u_2 \wedge f(a) \stackrel{?}{=} u_3 & \wedge & \\
 & & & z_2 \stackrel{?}{=} u_1 \wedge z_3 \stackrel{?}{=} u_3 & \wedge & g(b) \stackrel{?}{=} u_2.
\end{array}$$

Alle diese Lösungen müssen in P^0 für $z_1 + f(a) \stackrel{?}{=} z_2 + z_3 + g(b)$ eingesetzt werden. ■

2.3.2 Bedingungen für freie Variablen

Auch für die übrigen Variablen läßt sich eine ähnliche Bedingung aufstellen wie für die Abstraktionsvariablen, wenn ihnen im übrigen Unifikationsproblem P' ein Term zugewiesen wird.

Gibt es für eine Variable $z_i \in \mathcal{Z}$ eine Gleichung $z_i \stackrel{?}{=} t$ in P', so können während der elementaren AC-Unifikation alle Lösungspfade abgebrochen werden, die z_i einen Term s_i zuweisen, der mit t nicht unifizierbar ist. Über die Gestalt von t lassen sich aber noch weitere Aussagen machen. Es gilt $t(\epsilon) \notin \mathcal{F}_h$, da sonst die Gleichung $z_i \stackrel{?}{=} t$ in P_h und nicht in P' seien müßte. Ist $t \in \mathcal{V}$, hat diese Bedingung überhaupt keine Wirkung, da eine Variable mit jedem Term unifizierbar ist. Der Test auf Unifizierbarkeit zwischen t und s_i reduziert sich also lediglich auf die Überprüfung der Existenz einer Gleichung $z_i \stackrel{?}{=} t$ in P' mit nicht-variablem t. Existiert so ein $z_i \stackrel{?}{=} t$, dann gilt $t(\epsilon) \in \mathcal{F}_j$ mit $j \neq h$, woraus sofort folgt, daß t und s_i nicht unifizierbar sind.

Bedingung 2.2
$z \stackrel{?}{=} s \Rightarrow F$,
wenn $s \notin \mathcal{X}$ *und ein* $z \stackrel{?}{=} t$ *in* P' *existiert mit* $t \notin \mathcal{X}$.

Beispiel 2.6 Im obigen Beispiel fällt die Lösung $P_+^{s_2}$ weg, weil der Variablen z_3 ein AC-Term zugewiesen wird, obwohl es in P^0 die Gleichung $z_3 \stackrel{?}{=} f(z_4)$ gibt (auf die Gleichungen $z_3 \stackrel{?}{=} u_2 + u_4$ und $z_3 \stackrel{?}{=} f(z_4)$ kann die Regel *Conflict2* angewendet werden). ■

Zur Überprüfung dieser Bedingung wird dem elementaren Algorithmus die Existenz solcher Terme für die Variablen in P_h übergeben. Den Variablen, für die solch ein Term existiert, dürfen dann nur Variablen und kein AC-Term zugewiesen werden.

2.3.3 Bedingungen für Gleichsetzungen durch neue Variablen

Eine weitere früh zu erkennende Situation, in der das gerade bearbeitete Unifikationsproblem keine Lösungen mehr haben kann, tritt auf, wenn Terme aus $\{r_1, \ldots, r_l\}$ oder Variablen in P_h untereinander gleichgesetzt werden, indem ihnen das gleiche u_i zugewiesen wird. Zur Motivation soll dies erst an einem Beispiel verdeutlicht werden.

Beispiel 2.7 Die Lösung $P_+^{s_3}$ im obigen Beispiel führt zu keiner E-Lösung, da den Fremdtermen $f(a)$ und $g(b)$ die gleiche Variable u_3 zugewiesen wird, obwohl sie nicht unifizierbar sind (auf die Gleichungen $u_3 \stackrel{?}{=} f(a)$ und $u_3 \stackrel{?}{=} g(b)$ kann die Regel *Conflict2* angewendet werden). Eine ähnliche Situation tritt in Lösung $P_+^{s_4}$ auf, da $f(a)$ und der Variablen z_3, der im übrigen Unifikationsproblem noch der Term $f(z_4)$ zugeordnet ist, die gleiche Variable u_3 zugewiesen wird. Nach Anwendung von *Var-Rep* mit $\{u_3 \mapsto z_3\}$ erhält man die Gleichungen $z_3 \stackrel{?}{=} f(a)$ und $z_3 \stackrel{?}{=} f(z_4)$. Da die beiden Terme $f(a)$ und $f(z_4)$ unifizierbar sind, kann man diese Ableitung jedoch nicht abbrechen. ■

Zu jedem $u_i \in \mathcal{U}$ wird die Termmenge

$$Q_i = \{r_j \mid u_i \stackrel{?}{=} r_j \text{ in } P^3\} \cup \{t \mid z_j \stackrel{?}{=} u_i \text{ in } P^3 \text{ und } z_j \stackrel{?}{=} t \text{ in } P'\}$$

betrachtet. P^3 hat keine Lösungen, wenn sich die Terme in einem dieser Q_i nicht miteinander unifizieren lassen. Man kann also Lösungen P_h^s außer acht lassen, die zwei Variablen oder Fremdtermen dieselbe neue Variable zuweisen, obwohl die Fremdterme selbst oder die den Variablen in P' zugewiesenen Terme nicht miteinander unifizierbar sind.

Bedingung 2.3
$P_h^s \Rightarrow F$,
wenn ein $Q_i = \{r_j \mid u_i \stackrel{?}{=} r_j \text{ in } P_h^s\} \cup \{t \mid z_j \stackrel{?}{=} u_i \text{ in } P_h^s \text{ und } z_j \stackrel{?}{=} t \text{ in } P'\}$
existiert, das nicht unifizierbar ist.

Zu jeder Variablen z_i wird dem elementaren AC-Unifikationsalgorithmus zur Überprüfung dieser Bedingung die Menge $R'_{z_i} = \{t \mid z_i = t \text{ in } P' \text{ und } t \notin \mathcal{X}\}$ übergeben. Damit ist auch gleichzeitig die Bedingung 2.2, die Existenz eines nicht-variablen Terms t überprüfbar.

Da der Test auf Unifizierbarkeit sehr aufwendig sein kann, versucht man, ihn durch einen einfacheren zu ersetzen. Dies ist möglich, da durch die negative Formulierung der Bedingung nicht getestet werden muß, ob Q_i wirklich unifizierbar ist, sondern nur, ob es nicht unifizierbar ist. Anstatt die Unifikation wirklich durchzuführen, reicht es also, ein viel einfacheres Kriterium zu benutzen. In Kapitel 3 wird diese Bedingung daher einfach

zu einem Test der Kopfsymbole der Terme in Q_i reduziert. Durch diese Vereinfachung können dann zwar weniger Lösungspfade frühzeitig abgebrochen werden, aber man vermeidet die aufwendige Durchführung der Unifikation. Wird ein aufwendigerer Test, wie z. B. die Unifikation selbst durchgeführt, sollte auch bei einem positiven Ergebnis versucht werden, die berechnete Information, wie z. B. die Unifikatoren, zu speichern, da sie bei der später noch durchzuführenden Unifikation der Terme in Q_i benutzt werden kann. In Kapitel 5 wird hierauf näher eingegangen.

Beispiel 2.8 Im obigen Beispiel führt nur die Lösung P_+^{s4} zu einem Ergebnis. Nach Durchführung der Variablenersetzung und Quantorenelimination erhält man mit ihr die Form

$$(\exists u_1, u_2, u_3) z_1 \stackrel{?}{=} u_1 + u_2 \ \wedge \ z_2 \stackrel{?}{=} u_1 \ \wedge \ u_3 \stackrel{?}{=} f(a) \ \wedge \ u_3 \stackrel{?}{=} f(z_4) \ \wedge \ u_2 \stackrel{?}{=} g(b),$$

die nach der Anwendung von E_0-*Res* auf die Gleichungen mit freien Termen zu der sequentiell gelösten Form

$$(\exists u_2) z_1 \stackrel{?}{=} z_2 + u_2 \ \wedge \ z_3 \stackrel{?}{=} f(a) \ \wedge \ z_4 \stackrel{?}{=} a \ \wedge \ u_2 \stackrel{?}{=} g(b)$$

wird. Nach Anwendung von *Rep* und *EQE* erhält man die gelöste Form

$$z_1 \stackrel{?}{=} z_2 + g(b) \ \wedge \ z_3 \stackrel{?}{=} f(a) \ \wedge \ z_4 \stackrel{?}{=} a. \quad \blacksquare$$

2.3.4 Zusammenfassung

Die in diesem Abschnitt gezeigten drei Bedingungen führen dazu, daß auf die Abstraktionsvariablen für die elementare AC-Unifikation im Kombinationsalgorithmus ganz verzichtet werden kann. Außerdem führen sie dazu, daß viele AC-gelöste Formen schon in der elementaren AC-Unifikation für das gesamte Unifikationsproblem als ergebnislos erkannt werden können. Dies bedeutet nicht nur, daß diese Formen nach ihrer Berechnung nicht an den Kombinationsalgorithmus übergeben werden, sondern daß während der elementaren AC-Unifikation ihre Berechnung frühzeitig abgebrochen werden kann.

Kapitel 3

Elementare AC-Unifikation

3.1 Elementare AC-Unifikation

Elementare AC-Unifikation ist die Unifikation von Termen aus $\mathcal{T}(\{+\}, \mathcal{X})$ mit $+ \in \mathcal{AC}$. Gegeben ist also ein Unifikationsproblem $P \equiv s_1 \stackrel{?}{=} t_1, \ldots, s_m \stackrel{?}{=} t_m$ mit $s_i, t_i \in \mathcal{T}(\{+\}, \mathcal{X})$ und gesucht wird eine vollständige Menge von sequentiell gelösten Unifikationsproblemen. Der hier vorgestellte Algorithmus liefert die Unifikationsprobleme sogar in gelöster Form.

Als erstes werden die zu P gehörenden diophantischen Gleichungen aufgestellt. Seien $\{z_1, \ldots, z_n\} = Arg^+(P) = Var(P)$ die AC-Argumente von $+$ in P und $d(z_i, \{s_j \stackrel{?}{=} t_j\})$ die Anzahl der Auftreten von z_i in s_j minus der Anzahl der Auftreten von z_i in t_j. Dann sind die diophantischen Gleichungen von P

$$\begin{bmatrix} d(z_1, \{s_1 = t_1\}) & \ldots & d(z_n, \{s_1 = t_1\}) \\ \vdots & & \vdots \\ d(z_1, \{s_m = t_m\}) & \ldots & d(z_n, \{s_m = t_m\}) \end{bmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \vec{0}^m,$$

die als $A\vec{x} = \vec{0}$ mit den Unbekannten $\vec{x} = (x_1, \ldots, x_n)$ bezeichnet werden.

Beispiel 3.1 Die zu $z_1 + z_1 \stackrel{?}{=} z_3 + z_4 \ \wedge \ z_1 + z_2 \stackrel{?}{=} z_3 + z_4 + z_4$ mit $z_1, z_2, z_3, z_4 \in \mathcal{X}$ gehörenden diophantischen Gleichungen sind

$$\begin{bmatrix} 2 & 0 & -1 & -1 \\ 1 & 1 & -1 & -2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ mit } \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \vec{x}. \blacksquare$$

Die natürlichzahligen Lösungen eines solchen Gleichungssystems sind

$$S(A) = \{\vec{s} \in \mathbb{N}^n \mid A\vec{s} = \vec{0}\}.$$

Für die AC-Unifikation interessieren nur die **minimalen Lösungen**, d. h. die Lösungen ungleich null, für die keine kleinere Lösung ungleich null existiert:

$$\mu S(A) = \{\vec{s} \in S(A) \mid \vec{s} \neq \vec{0} \text{ und es existiert kein } \vec{t} \in S(A) \text{ mit } \vec{0} < \vec{t} < \vec{s}\}.$$

Diese Menge ist immer endlich. Algorithmen zu ihrer Berechnung werden in Kapitel 4 vorgestellt.

Beispiel 3.2 Die Menge der minimalen Lösungen wird in den Beispielen zur besseren Lesbarkeit in Tabellenform untereinander und zusammen mit den AC-Argumenten und der Matrix der diophantischen Gleichung dargestellt. Im obigen Beispiel also

$$\begin{array}{rcl} Arg^+(P) & = & \{ \quad z_1 \quad z_2 \quad z_3 \quad z_4 \quad \} \\ A & = & \left[\begin{array}{cccc} 2 & 0 & -1 & -1 \\ 1 & 1 & -1 & -2 \end{array} \right] \\ \mu S(A) & = & \{(\quad 1 \quad 3 \quad 0 \quad 2 \quad) \qquad (\vec{s}_1) \\ & & \quad (\quad 1 \quad 1 \quad 2 \quad 0 \quad) \qquad (\vec{s}_2) \\ & & \quad (\quad 1 \quad 2 \quad 1 \quad 1 \quad)\}. \qquad (\vec{s}_3) \end{array}$$

Auch (2 3 3 1) ist eine Lösung, ist aber nicht minimal, da $\vec{s}_2 < (2\ 3\ 3\ 1)$. ■

Im nächsten Schritt werden bestimmte Teilmengen der Menge der minimalen Lösungen $\mu S(A)$ bestimmt. Eine Teilmenge $\mathcal{S} = \{\vec{s}_1, \ldots, \vec{s}_p\} \subseteq \mu S(A)$ ist **groß genug**, wenn jede „Spalte“ in der Tabellennotation wie im obigen Beispiel in mindestens einer Lösung in $\mathcal{S}$ größer null ist, d. h. wenn

$$\sum_{j=1}^{p} \vec{s}_j(i) > 0 \text{ für alle } i \in \{1, \ldots, n\}. \tag{3.1}$$

Im letzten Schritt werden dann die gelösten Unifikationsprobleme zu $\mathcal{S}$ aufgestellt. Dabei entspricht jeder Teilmenge, die groß genug ist, genau ein gelöstes Unifikationsproblem. Zuerst wird jeder Lösung $\vec{s}_i$ eine neue Variable u_i zugeordnet. Die gelöste Form zu einer Teilmenge $\mathcal{S} = \{\vec{s}_1, \ldots, \vec{s}_p)$ ist dann

$$P^s \equiv (\exists u_1, \ldots, u_p) z_1 \stackrel{?}{=} u_1^{\vec{s}_1(1)} + \ldots + u_p^{\vec{s}_p(1)} \ \wedge \ \ldots \ \wedge \ z_n \stackrel{?}{=} u_1^{\vec{s}_1(n)} + \ldots + u_p^{\vec{s}_p(n)}.$$

Dabei ist $u_i^0 + t = t$ und $u_i^{n+1} = u_i + u_i^n$. Die Bedingung, daß $\mathcal{S}$ groß genug ist, sorgt dafür, daß dies sinnvoll ist und kein „leerer“ Term auftritt. Die gelösten Formen aller Teilmengen, die groß genug sind, sind eine vollständige Menge von gelösten Formen für P.

Beispiel 3.3 Die Teilmengen, die im Beispiel groß genug sind, sind $\{\vec{s}_3\}, \{\vec{s}_1, \vec{s}_3\}$, $\{\vec{s}_2, \vec{s}_3\}$, $\{\vec{s}_1, \vec{s}_2\}$ und $\{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$. Werden $\vec{s}_1, \vec{s}_2, \vec{s}_3$ die Variablen u_1, u_2, u_3 zugeordnet, so ist die gelöste Form zu $\{\vec{s}_2, \vec{s}_3\}$

$$(\exists u_2, u_3) z_1 \stackrel{?}{=} u_2 + u_3 \ \wedge \ z_2 \stackrel{?}{=} u_2 + u_3 + u_3 \ \wedge \ z_3 \stackrel{?}{=} u_2 + u_2 + u_3 \ \wedge \ z_4 \stackrel{?}{=} u_3. \quad ■$$

3.2 Elementare AC-Unifikation mit Fremdtermen

Sei nun $P \equiv s_1 \stackrel{?}{=} t_1 \wedge \ldots \wedge s_m \stackrel{?}{=} t_m$ mit $s_i, t_i \in \mathcal{T}(\Sigma, \mathcal{X})$, aber weiterhin $s_i(\epsilon), t_i(\epsilon) \in \mathcal{X} \cup \mathcal{F}_h$. Die Fremdterme in P seien $\{r_1, \ldots, r_l\} = Arg^+(P) \setminus \mathcal{X}$. Ersetzt man diese Terme durch die Abstraktionsvariablen $\{v_1, \ldots, v_l\}$ so kann man den obigen Algorithmus zur elementaren AC-Unifikation benutzen, wenn man anschließend die Gleichungen $v_1 \stackrel{?}{=} r_1, \ldots, v_l \stackrel{?}{=} r_l$ zum Unifikationsproblem hinzufügt. In diesem Abschnitt soll nun aber gezeigt werden, wie man die in Kapitel 2 gezeigten Bedingungen einhält und dadurch auf die Abstraktionsvariablen verzichten kann. Ähnliche Bedingungen sind zum Teil auch schon in [23], [8] und [16] vorgeschlagen worden. Aber die Bedingungen in [16], die den hier vorgestellten am nächsten kommen, beziehen sich auf den (häufig auftretenden) Spezialfall, daß jedes AC-Argument nur einmal vorkommt. Bei allen drei Verfahren werden außerdem die Bedingungen nicht an den Algorithmus zum Lösen der diophantischen Gleichungen weitergegeben, und sie beziehen sich nur auf ein Termpaar, das unifiziert werden soll. Für die AC-Theorie ist es aber wichtig, auch Mengen von Termpaaren mit der elementaren Unifikation lösen zu können, wie in Kapitel 5 gezeigt wird.

Seien $\{a_1, \ldots, a_n\} = Arg^+(P)$ die AC-Argumente der Terme in P. Die diophantischen Gleichungen werden nun wie oben aufgestellt:

$$\begin{bmatrix} d(a_1, \{s_1 = t_1\}) & \ldots & d(a_n, \{s_1 = t_1\}) \\ \vdots & & \vdots \\ d(a_1, \{s_m = t_m\}) & \ldots & d(a_n, \{s_m = t_m\}) \end{bmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = 0^m.$$

Bedingung 2.1, die besagt, daß einer Abstraktionsvariablen nur eine Variable zugewiesen werden darf, kann man einhalten, indem man eine zusätzliche Anforderung an die Teilmengen stellt. Eine Teilmenge von Lösungen $\mathcal{S} \subseteq \mu S(A)$ ist **für Fremdterme klein genug**, wenn jede zu einem Fremdterm gehörende Spalte genau eine 1 und sonst nur 0 enthält:

$$\sum_{\vec{s}_i \in \mathcal{S}} \vec{s}_i(j) = 1 \text{ für alle } j \text{ mit } a_j \notin \mathcal{X}. \tag{3.2}$$

Nun werden nur noch zu den Teilmengen, die groß genug und für Fremdterme klein genug sind, gelöste Formen aufgestellt. Da nun in einer gelösten Form zu einer Teilmenge den Fremdtermen $\{r_1, \ldots, r_l\}$ durch die Unifikation nur Variablen zugewiesen werden, kann auf die Variablenabstraktion verzichtet werden und man erhält die von der Regel *C-AC-Res* in Kapitel 2 geforderte AC-gelöste Form des Unifikationsproblems.

Beispiel 3.4 Für $P \equiv z_1 + z_2 \stackrel{?}{=} a + f(z_3)$ mit $z_1, z_2, z_3 \in \mathcal{X}, a, f \in \mathcal{F}_0$ sind die AC^+-Argumente, die Matrix der diophantischen Gleichungen und die Menge der minimalen

Lösungen aus der folgenden Tabelle zu entnehmen.

$$\begin{array}{rcl} Arg^+(P) & = & \{\ z_1 \quad z_2 \quad a \quad f(z_3)\} \\ A & = & [\ 1 \quad 1 \quad -1 \quad -1\ ] \\ \mu S(A) & = & \{(\ 1 \quad 0 \quad 1 \quad 0\ ) \qquad (\vec{s}_1) \\ & & (\ 0 \quad 1 \quad 0 \quad 1\ ) \qquad (\vec{s}_2) \\ & & (\ 1 \quad 0 \quad 0 \quad 1\ ) \qquad (\vec{s}_3) \\ & & (\ 0 \quad 1 \quad 1 \quad 0\ )\} \qquad (\vec{s}_4) \end{array}$$

Die einzigen Teilmengen, die groß genug und für Fremdterme klein genug sind, sind $\{\vec{s}_1, \vec{s}_2\}$ und $\{\vec{s}_3, \vec{s}_4\}$ mit den AC^+-gelösten Formen

$$(\exists u_1, u_2) z_1 \stackrel{?}{=} u_1 \ \wedge\ z_2 \stackrel{?}{=} u_2 \ \wedge\ u_1 \stackrel{?}{=} a \ \wedge\ u_2 \stackrel{?}{=} f(z_3) \text{ und}$$

$$(\exists u_3, u_4) z_1 \stackrel{?}{=} u_3 \ \wedge\ z_2 \stackrel{?}{=} u_4 \ \wedge\ u_4 \stackrel{?}{=} a \ \wedge\ u_3 \stackrel{?}{=} f(z_3).$$

Es gibt sieben Teilmengen, die groß genug sind. Die zusätzlichen fünf Teilmengen sind aber nicht nach Gleichung 3.2 für Fremdterme klein genug und für sie braucht daher keine gelöste Form aufgestellt zu werden. So enthält z. B. $\{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$ in der vierten Spalte, der der Fremdterm $f(z_3)$ zugeordnet ist, in $\vec{s}_2$ und $\vec{s}_3$ einen Wert ungleich null, d. h. $\vec{s}_1(4) + \vec{s}_2(4) + \vec{s}_3(4) > 1$. Das mit dieser Teilmenge erzeugte Unifikationsproblem hätte die Gleichung $u_2 + u_3 \stackrel{?}{=} f(z_3)$ enthalten, die keine E-Lösung hat. ■

Nun sollen auch noch die restlichen Bedingungen aus Kapitel 2 berücksichtigt werden. Dazu wird dem Algorithmus zu jeder Variablen $z \in Arg^+(P) \cap \mathcal{X}$ eine Termmenge R'_z übergeben. Diese enthält die nicht-variablen Terme, mit denen z im übrigen Unifikationsproblem vor Aufruf der elementaren AC-Unifikation gleichgesetzt wurde. Jedem AC-Argument a_j (und damit jeder Spalte in der Matrix der diophantischen Gleichungen) wird nun eine Termmenge R_j zugeordnet. Ist $a_j = z$ eine Variable, so ist $R_j = R'_z$; sonst ist a_j ein Fremdterm und $R_j = \{a_j\}$.

Für die Terme $t \in R_j$ gilt $t(\epsilon) \in \mathcal{F}_i$ mit $i \neq h$. Für alle Variablen z_j, denen eine nichtleere Termmenge zugeordnet ist, kann nach Bedingung 2.2 dieselbe Anforderung an die passenden Teilmengen gestellt werden, wie für die Abstraktionsvariablen. Diesen Variablen darf durch die elementare AC-Unifikation nur eine Variable zugewiesen werden und daher muß eine Spalte, der eine solche Variable zugeordnet ist, genau eine 1 und sonst nur 0 enthalten. Eine Teilmenge $\mathcal{S}$ von Lösungen ist also **klein genug**, wenn

$$\sum_{\vec{s}_i \in \mathcal{S}} \vec{s}_i(j) \leq 1 \text{ für alle } j \text{ mit } R_j \neq \emptyset. \tag{3.3}$$

Man beachte, daß durch die Definition von R_j für Fremdterme a_j diese Gleichung die Gleichung 3.2 subsumiert und damit eine Menge von Lösungen, die klein genug ist, auch klein genug für Fremdterme ist.

Eine Teilmenge ist **passend**, wenn sie groß genug (Gleichung 3.1) und klein genug ist. Um eine vollständige Menge von gelösten Formen für das gesamte Unifikationsproblem zu erhalten, brauchen hier nur die passenden Teilmengen betrachtet zu werden.

Bedingung 2.3 besagt, daß die implizit über eine Variable u_i gleichgesetzten Terme unifizierbar sein müssen, um zu einer Lösung zu führen. Auf diese Art gleichgesetzt, werden alle Terme, die den Spalten zugeordnet sind, die in der Lösung $\vec{s_i}$ ungleich 0 ist. Die Menge dieser Terme ist $Q_i = \{t \mid t \in R_j, \vec{s_i}(j) > 0\}$. Sind die Terme in Q_i nicht miteinander unifizierbar, kann eine Teilmenge, die $\vec{s_i}$ enthält, nicht zu einer Lösung führen.

Beispiel 3.5 Für

$$P \equiv z_1 + z_1 + z_2 \stackrel{?}{=} z_3 + f(z_4) + g(z_5) \ \wedge \ z_2 \stackrel{?}{=} f(z_6)$$

mit $z_i \in \mathcal{X}$, $g, f \in \mathcal{F}_0$ wird die elementare AC-Unifikation mit

$$P_+ \equiv z_1 + z_1 + z_2 \stackrel{?}{=} z_3 + f(z_4) + g(z_5) \text{ und } R_{z_1} = R_{z_3} = \emptyset, R_{z_2} = \{f(z_6)\}$$

aufgerufen. Die AC^+-Argumente, die zugeordneten Termmengen, die Matrix der diophantischen Gleichungen und die Menge der minimalen Lösungen sind in der folgenden Tabelle zu sehen.

$Arg^+(P)$	=	$\{$ z_1	z_2	z_3	$f(z_4)$	$g(z_5)$ $\}$	
R_j		$\emptyset$	$\{f(z_6)\}$	$\emptyset$	$\{f(z_4)\}$	$\{g(z_5)\}$	
A	=	$[$ 2	1	-1	-1	-1 $]$	
$\mu S(A)$	=	$\{($ 1	0	2	0	0 $)$	$(\vec{s}_1)$
		$($ 1	0	0	2	0 $)$	$(\vec{s}_2)$
		$($ 1	0	0	0	2 $)$	$(\vec{s}_3)$
		$($ 1	0	1	1	0 $)$	$(\vec{s}_4)$
		$($ 1	0	1	0	1 $)$	$(\vec{s}_5)$
		$($ 1	0	0	1	1 $)$	$(\vec{s}_6)$
		$($ 0	1	1	0	0 $)$	$(\vec{s}_7)$
		$($ 0	1	0	1	0 $)$	$(\vec{s}_8)$
		$($ 0	1	0	0	1 $)\}$	$(\vec{s}_9)$

Die Menge $\{\vec{s}_5, \vec{s}_7, \vec{s}_8\}$ ist keine passende Teilmenge, weil sie nach 3.3 nicht klein genug ist. Die zweite Spalte, der eine nicht leere Termmenge zugeordnet ist, enthält in $\vec{s}_7$ und $\vec{s}_8$ einen Wert ungleich null, d. h. $\vec{s}_5(2) + \vec{s}_7(2) + \vec{s}_8(2) > 1$. Ein aus dieser Teilmenge berechnetes Unifikationsproblem hätte die Gleichungen $z_2 \stackrel{?}{=} u_7 + u_8$ enthalten. Zusammen mit der Gleichung $z_2 \stackrel{?}{=} f(z_6)$ aus dem ursprünglichen Unifikationsproblem hätte für diesen Berechnungspfad keine E-Lösung existiert.

Die den Lösungen $\vec{s}_6$ und $\vec{s}_9$ zugeordneten Termmengen $Q_6 = \{f(z_4), g(z_5)\}$ und $Q_9 = \{f(z_6), g(z_5)\}$ sind nicht unifizierbar. Diese beiden Lösungen können also in keiner passenden Teilmenge vorkommen. Käme z. B. $\vec{s}_6$ in einer Teilmenge vor, so enthielte das daraus entstandene Unifikationsproblem die Gleichungen $u_6 \stackrel{?}{=} f(z_4)$ und $u_6 \stackrel{?}{=} g(z_5)$, die keine E-Lösung haben.

Die Lösungen $\vec{s}_2$ und $\vec{s}_3$ können ebenfalls in keiner passenden Teilmenge vorkommen. Eine Teilmenge $\mathcal{S}$, die $\vec{s}_2$ oder $\vec{s}_3$ enthält, ist nicht klein genug, weil $\sum_{\vec{s}_i \in \mathcal{S}} \vec{s}_i(4) > 1$ bzw. $\sum_{\vec{s}_i \in \mathcal{S}} \vec{s}_i(5) > 1$, da $\vec{s}_2(4) = 2 > 1$ und $\vec{s}_3(5) = 2 > 1$.

Die passenden Teilmengen in diesem Beispiel sind $\{\vec{s}_5, \vec{s}_8\}, \{\vec{s}_4, \vec{s}_5, \vec{s}_7\}, \{\vec{s}_1, \vec{s}_5, \vec{s}_8\}$ und $\{\vec{s}_1, \vec{s}_4, \vec{s}_5, \vec{s}_7\}$. Die AC^+-gelöste Form zu $\{\vec{s}_5, \vec{s}_8\}$ ist

$$(\exists u_5, u_8) x_1 \stackrel{?}{=} u_5 \ \wedge \ x_2 \stackrel{?}{=} u_8 \ \wedge \ z_3 \stackrel{?}{=} u_5 \ \wedge \ u_8 \stackrel{?}{=} f(z_4) \ \wedge \ u_5 \stackrel{?}{=} g(z_5)$$

Ohne die Bedingungen 2.1, 2.2 und 2.3 hätte es 19 passende Teilmengen gegeben. ■

Eine Lösung $\vec{s}_i$, der eine nicht unifizierbare Termmenge Q_i zugeordnet ist oder die eine Spalte j mit einer nicht leeren Termmenge R_j hat, die einen Wert größer 1 enthält, kann in keiner passenden Teilmenge enthalten sein und braucht daher vom Algorithmus zum Lösen der diophantischen Gleichungen auch gar nicht erzeugt zu werden. An jede Lösung $\vec{s}_i$ können also die folgenden Bedingungen gestellt werden.

Bedingung 3.1
$\vec{s}_i(j) < 2$
für alle j *mit* $R_j \neq \emptyset$

Bedingung 3.2
$Q_i = \{t \mid t \in R_j, \vec{s}_i(j) > 0\}$
ist unifizierbar.

Die Termmengen R_j müssen also auch dem Algorithmus zum Lösen der diophantischen Gleichungen übergeben werden, damit dieser die Bedingungen überprüfen kann und nur die minimalen Lösungen liefert, die sie erfüllen. In der Praxis wird man den Test auf Unifizierbarkeit aus Komplexitätsgründen auf einen einfacheren Test reduzieren und z. B. nur testen, ob Q_i Terme mit verschiedenen Kopfsymbolen enthält. Dem Algorithmus zum Lösen der diophantischen Gleichungen wird dann für jede Spalte j ein Kopfsymbol $f_j \in \Sigma$ aus R_j übergeben (die Terme in R_j haben alle dasselbe Kopfsymbol, weil sonst die Regel *Conflict2* anwendbar war). Wenn $R_j = \emptyset$ ist, wird ein Platzhalter $f_j = \epsilon \notin \Sigma$ übergeben. Für jede Lösung $\vec{s}_i$ wird dann überprüft, ob allen Spalten mit einem Wert ungleich null dasselbe Funktionssymbol zugeordnet ist.

Bedingung 3.2'
$f_j = f_k$
für alle j, k *mit* $\vec{s}_i(j) > 0, \vec{s}_i(k) > 0$ *und* $f_j, f_k \in \Sigma$

Kapitel 4

Lösen diophantischer Gleichungssysteme

In diesem Kapitel geht es um das Lösen eines diophantischen Gleichungssystems $A\vec{x} = 0^n$ mit den Unbekannten $\vec{x}$ mit Lösungen aus $\mathbb{N}^n$. Gegeben ist die $m \times n$-Matrix A mit ganzzahligen Koeffizienten, und gesucht werden die Lösungen $S(A) = \{\vec{s} \in \mathbb{N}^n \mid A\vec{s} = 0^n\}$. Da man nicht alle diese unendlich vielen Lösungen aufstellen kann, berechnet man nur die minimalen Lösungen

$$\mu S(A) = \{\vec{s} \in S(A) \mid \vec{s} \neq 0^n \text{ und es gibt keine Lösung } \vec{t} \in S(A) \text{ mit } 0^n <^n \vec{t} <^n \vec{s}\}.$$

Die nicht minimalen Lösungen ergeben sich dann als Linearkombination der Vektoren aus $\mu S(A)$:

$$S(A) = \left\{ \sum_{\vec{s} \in \mu S(A)} a_{\vec{s}} \vec{s} \mid a_{\vec{s}} \in \mathbb{N} \right\}.$$

Das Verfahren von Contejean und Devie [6] arbeitet auf eine sehr einfache Art und Weise. Die Vektoren des $\mathbb{N}^n$ werden aufgezählt bis alle minimalen Lösungen gefunden sind. Die Verfahren von Domenjoud [7] und Pottier [19] arbeiten nach einem anderen Prinzip. Bei ihnen wird zuerst der Lösungsuntervektorraum der diophantischen Gleichungen in $\mathbb{Q}^n$ bzw. $\mathbb{Z}^n$ berechnet. Dann werden in diesen Untervektorräumen die minimalen natürlichzahligen Lösungen gesucht.

4.1 Verfahren von Contejean und Devie

Dem Verfahren von Contejean und Devie [6] liegt das Verfahren von Fortenbacher und Clausen [5] zugrunde. Während in [5] nur das Lösen einer diophantischen Gleichung beschrieben ist (A ist eine $1 \times n$-Matrix), wird in [6] dieses Verfahren auf Systeme von diophantischen Gleichungen erweitert. Die Idee, auf der diese Verfahren beruhen, besteht darin, die Vektoren aus $\mathbb{N}^n$ aufzuzählen und dabei zu testen, welche von ihnen minimale

Lösungen des diophantischen Gleichungssystems sind. Um sicherzustellen, daß der Algorithmus terminiert, werden dazu Schranken benötigt, die den Bereich, aus dem Vektoren aufgezählt werden, begrenzen.

Die erste bei diesem wie auch bei vielen anderen Verfahren verwendete Schranke entsteht aus der Beschränkung auf minimale Lösungen: Vektoren, die größer sind als eine minimale Lösung, brauchen nicht betrachtet zu werden, da diese Vektoren keine minimalen Lösungen sein können. Die zweite, entscheidende Schranke begrenzt den Suchraum auf Vektoren, die nicht „zu weit davon entfernt sind“, eine Lösung zu sein.

Um diesen „Abstand“ messen zu können, wird zu einer $m \times n$-Matrix A für die Vektoren $\vec{a} \in \mathbb{N}^n$, $\vec{a} = (a_1, \ldots, a_n)$ der **Defekt** $\vec{d}^A(\vec{a}) \in \mathbb{Z}^m$ betrachtet. Er ist definiert durch

$$\vec{d}^A(\vec{a}) = A\vec{a},$$

d. h. für die i-te Komponente mit $i \in \{1, \ldots, m\}$ gilt

$$\vec{d}^A(\vec{a})(i) = \sum_{j=1}^{n} A^j(i)\vec{a}(j).$$

Ein Vektor $\vec{a} \in \mathbb{N}^n$ ist genau dann eine Lösung, wenn der Defekt $\vec{d}^A(\vec{a})$ der Nullvektor 0^m ist. Da der Defekt eines Einheitsvektors gleich der jeweiligen Spalte in der Matrix der diophantischen Gleichungen ist, $\vec{d}^A(\vec{e}_j) = A^j$, sind die Defekte der Einheitsvektoren $D = \{\vec{d}^A(\vec{e}_1), \ldots, \vec{d}^A(\vec{e}_n)\}$ von besonderer Bedeutung. Außerdem gilt für den Defekt von zwei Vektoren $\vec{d}^A(\vec{a}) + \vec{d}^A(\vec{b}) = \vec{d}^A(\vec{a} + \vec{b})$.

4.1.1 Aufzählen der Vektoren

Das Aufzählen der Vektoren aus $\mathbb{N}^n$ geschieht, indem ausgehend von 0^n die Komponenten nacheinander um eins erhöht werden bis eine Lösung für die diophantischen Gleichungen erreicht ist. Das entspricht dem sukzessiven Addieren von Vektoren aus D bis das Ergebnis gleich 0^m ist. Die Reihenfolge des Aufzählens entspricht der Breitensuche in einem Baum, in dem jedem Knoten ein Vektor aus $\mathbb{N}^n$ und der zugehörige Defekt zugeordnet ist. Jeder Knoten hat n Nachfolger, deren Vektoren in jeweils einer Komponente gegenüber dem Vorgänger um eins erhöht sind. Der Wurzel ist $(0^n, 0^m)$ zugeordnet und den Nachfolgern eines Knoten, dem $(\vec{a}, \vec{d}^A(\vec{a}))$ zugeordnet ist, sind $(\vec{a} + \vec{e}_1, \vec{d}^A(\vec{a}) + \vec{d}^A(\vec{e}_1)), \ldots, (\vec{a} + \vec{e}_n, \vec{d}^A(\vec{a}) + \vec{d}^A(\vec{e}_n))$ zugeordnet. Für jeden Knoten mit $(\vec{a}, \vec{d}^A(\vec{a}))$ und jeden seiner Nachfolger mit $(\vec{a}', \vec{d}^A(\vec{a}'))$ gilt $\vec{a} <^n \vec{a}'$.

Der entscheidende Punkt ist nun, welche Knoten mit den darunter liegenden Teilbäumen wegen der Schranken nicht betrachtet werden müssen. Die erste Schranke ist trivial.

I. *Sind mehreren Knoten dieselben Vektoren zugeordnet, braucht nur einer dieser Knoten weiter betrachtet zu werden.*

Da für einen Knoten mit $(\vec{a}, \vec{d}^A(\vec{a}))$ die Tiefe im Suchbaum gleich der Länge $l(\vec{a})$ ist, kann ein Knoten mit denselben Vektoren nur in derselben Tiefe vorkommen. Zur Einhaltung dieser Schranke ist bei der Behandlung eines Knotens also nur zu testen, ob in der aktuellen Tiefe des Suchbaums bereits ein Knoten mit diesen Vektoren betrachtet wurde.

Die nächsten beiden Schranken ergeben sich aus der Tatsache, daß nur minimale Lösungen gesucht werden.

II. *Ein Knoten mit $(\vec{a}, \vec{d}^A(\vec{a}))$ und $\vec{d}^A(\vec{a}) = 0^m$ braucht nicht weiter betrachtet zu werden.*

Da ein Vektor mit Defekt $\vec{d}^A(\vec{a}) = 0^m$ eine Lösung ist, brauchen seine Nachfolger nicht betrachtet zu werden, weil sie keine minimalen Lösungen sein können. Der Vektor $\vec{a}$ muß nur in die Menge der bereits gefundenen minimalen Lösungen aufgenommen werden.

III. *Ein Knoten mit $(\vec{a}, \vec{d}^A(\vec{a}))$ braucht nicht weiter betrachtet zu werden, wenn eine minimale Lösung $\vec{s}$ mit $\vec{s} <^n \vec{a}$ existiert.*

Da Breitensuche benutzt wird, hat man, wenn man einen Vektor untersucht, bereits alle kleineren Vektoren bearbeitet und somit auch alle kleineren Lösungen bereits gefunden. Man muß zum Testen von Schranke III also nur überprüfen, ob der aktuelle Vektor größer ist als eine der bisher gefundenen Lösungen. Diese Schranke verhindert auch, daß Lösungen gefunden werden, die nicht minimal sind.

Während die ersten drei Schranken trivial sind und in ähnlicher Form in den meisten anderen Algorithmen vorkommen, enthält die vierte Schranke den eigentlichen Kern des Verfahrens. Mit ihr soll erreicht werden, daß der Defekt bei jedem Schritt auf den Nullpunkt zuwandert. Dazu wird der Vektorraum $\mathbb{Z}^m$, in dem sich der Defekt bewegt, durch eine Hyperebene, die senkrecht auf dem alten Defekt $\vec{d}^A(\vec{a})$ steht, in zwei Halbräume geteilt (siehe Abbildung 4.1). Der neue Defekt $\vec{d}^A(\vec{a} + \vec{e}_j)$ muß nun in dem Halbraum liegen, in dem auch der Nullpunkt liegt. Dies wird durch die Bedingung $\vec{d}^A(\vec{a}) \cdot \vec{d}^A(\vec{e}_j) < 0$ ausgedrückt, wobei $\cdot$ das Skalarprodukt zweier Vektoren ist.

IV. *Für einen Knoten mit $(\vec{a}, \vec{d}^A(\vec{a}))$ brauchen nur die Nachfolger betrachtet werden, denen $(\vec{a} + \vec{e}_j, \vec{d}^A(\vec{a} + \vec{e}_j))$ mit $\vec{d}^A(\vec{a}) \cdot \vec{d}^A(\vec{e}_j) < 0$ zugeordnet ist.*

In Abbildung 4.2 steht der Algorithmus zur Berechnung der minimalen Lösungen, in dem die vier Schranken benutzt werden. Der Algorithmus ist korrekt, vollständig und terminiert. Für Beweise siehe [6].

4.1.2 Zusätzliche Schranken

Nun sollen die Bedingungen 3.1 und 3.2 aus Kapitel 3 in den Algorithmus übernommen werden. Wie in Kapitel 3 beschrieben, wird dem Algorithmus dazu zu jeder Spalte ein

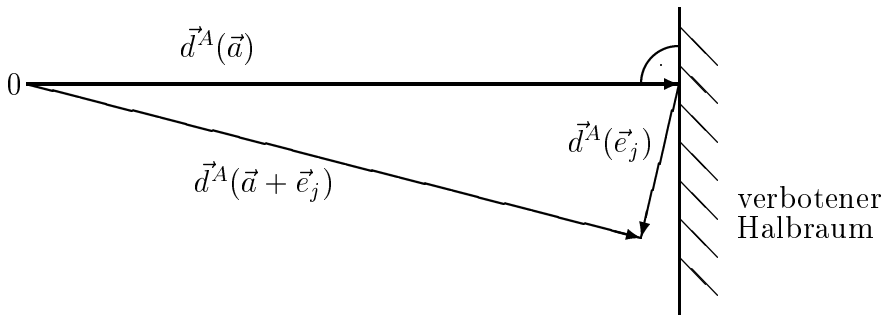

Abbildung 4.1: *Der durch* $\vec{d}^A(\vec{a}) \cdot \vec{d}^A(\vec{e}_j) < 0$ *für* $\vec{d}^A(\vec{a}+\vec{e}_j)$ *verbotene Halbraum.*

Wert f_j übergeben, der entweder ein Funktionssymbol aus Σ oder ein Platzhalter $\epsilon \notin \Sigma$ ist. Um die Bedingungen einzuhalten, werden Schranken aufgestellt, die zu den bisherigen Schranken hinzukommen.

Bedingung 3.1 ist einfach durch eine Schranke auszudrücken.

V. *Ein Knoten mit* $(\vec{a}, \vec{d}^A(\vec{a}))$ *braucht nicht weiter betrachtet zu werden, wenn er eine Komponente* j *mit* $\vec{a}(j) > 1$ *enthält, für die* $f_j \in \Sigma$ *gilt.*

Diese Schranke verhindert, daß eine Lösung erzeugt wird, die nie die Gleichung 3.3 erfüllen kann, weil sie in einer Spalte, der Fremdterme zugeordnet sind, einen Wert größer 1 enthält. Im Algorithmus muß getestet werden, ob der gerade erhöhte Wert einer Komponente größer als 1 ist, wenn dieser Komponente ein Funktionssymbol zugeordnet ist.

Bedingung 3.2' wird durch die folgende Schranke eingehalten.

VI. *Ein Knoten mit* $(\vec{a}, \vec{d}^A(\vec{a}))$ *braucht nicht weiter betrachtet zu werden, wenn er zwei Komponenten* j *und* k *mit* $\vec{a}(j) > 0$, $\vec{a}(k) > 0$ *enthält, für die* $f_j, f_k \in \Sigma$ *und* $f_j \neq f_k$ *gilt.*

Diese Schranke verhindert, daß Lösungen berechnet werden, die später dazu führen, daß einer Variablen zwei Terme zugewiesen werden, die verschiedene Kopfsymbole haben und damit nicht unifizierbar sind. Im Algorithmus wird dann überprüft, ob der Komponente, deren Wert gerade erhöht wurde, ein Funktionssymbol zugeordnet ist. Ist dies der Fall, so darf den anderen Komponenten mit einem Wert größer null nur dieses Funktionssymbol oder ϵ zugeordnet sein.

Der Algorithmus aus Abbildung 4.2 wird durch die Ergänzungen dieses Abschnitts geändert, indem hinter den ersten vier Schranken I.–IV. die folgenden beiden Zeilen eingefügt werden.

V. $\vec{a}(j) < 2$ oder $f_j = \epsilon \in \Sigma$ und
VI. $f_j = \epsilon$ oder für alle $k \in \{1, \ldots, n\}$ mit $\vec{a}(k) > 0$ gilt $f_k = \epsilon$ oder $f_k = f_j$

1. $V_0 := \{(\vec{e}_1, \vec{d}^A(\vec{e}_1)), \ldots, (\vec{e}_n, \vec{d}^A(\vec{e}_n))\}$; $M := \emptyset$; $i := 0$.
2. $i := i + 1$; $V_i := \emptyset$.
3. Für alle $(\vec{a}, \vec{d}^A(\vec{a})) \in V_{i-1}$
   Für alle $\vec{e}_j \in \{\vec{e}_1, \ldots, \vec{e}_n\}$
   Wenn I. $(\vec{a} + \vec{e}_j, \vec{d}^A(\vec{a} + \vec{e}_j)) \notin V_i$ und
   II. $\vec{d}^A(\vec{a} + \vec{e}_j) \neq 0^m$ und
   III. $\vec{s} \not<^n \vec{a} + \vec{e}_j$ für alle $\vec{s} \in M$ und
   IV. $\vec{d}^A(\vec{a}) \cdot \vec{d}^A(\vec{e}_j) < 0$ dann
   $V_i := V_i \cup \{(\vec{a} + \vec{e}_j, \vec{d}^A(\vec{a} + \vec{e}_j))\}$.
   Wenn $\vec{d}^A(\vec{a} + \vec{e}_j) = 0^m$ dann
   $M := M \cup \{\vec{a} + \vec{e}_j\}$.
4. Wenn $V_i \neq \emptyset$ dann
   weiter mit Schritt 2
   sonst
   Stopp. M ist die Menge der minimalen Lösungen $\mu S(A)$.

Abbildung 4.2: *Algorithmus von E. Contejean und H. Devie zur Berechnung der minimalen Lösungen* $\mu S(A)$

Die Schranke VI kann effizienter getestet werden, wenn den Knoten noch ein zusätzlicher Wert $f(\vec{a})$ zugeordnet wird, der für die Wurzel mit dem Null-Vektor der Platzhalter ϵ ist und der auf ein Funktionssymbol f_j gesetzt wird, wenn die j-te Komponente mit $f_j \in \Sigma$ größer null wird. Dann muß beim Erhöhen des Wertes einer Komponente k nur getestet werden, ob $f(\vec{a})$ und f_k gleich sind, wenn es Funktionssymbole sind.

4.2 Verfahren von Domenjoud

Im Verfahren von Domenjoud wird zuerst der Untervektorraum von $\mathbb{Q}^n$ bestimmt, der der Lösungsraum zu den diophantischen Gleichungen $A\vec{x} = 0^n$ ist, wobei A weiterhin eine $m \times n$-Matrix ist. Dies geschieht, indem die minimalen Lösungen aus $\mathbb{N}^n$ berechnet werden, deren Trägermenge[1] bezüglich der Teilmengenordnung minimal ist.

$$S_0(A) = \left\{ \vec{s} \in \mu S(A) \mid \text{es gibt kein } \vec{t} \in \mu S(A) \text{ mit } C(\vec{t}) \subset C(\vec{s}) \right\}$$

Die Vektoren aus $S_0(A)$ lassen sich im Gegensatz zu den restlichen Vektoren aus $\mu S(A)$ direkt aus der Matrix A berechnen. Dazu bildet man für alle Teilmengen $\{i_1, \ldots, i_{m+1}\} \subseteq \{1, \ldots, n\}$ der Kardinalität $m+1$ mit $\mathcal{A} = \{A^{i_1}, \ldots, A^{i_{m+1}}\}$ die Determinante

$$D(\mathcal{A}) = \left| \begin{bmatrix} \vec{e}_{i_1} & \ldots & \vec{e}_{i_{m+1}} \\ A^{i_1} & \ldots & A^{i_{m+1}} \end{bmatrix} \right| = \sum_{j=1}^{m+1} (-1)^{j+1} \vec{e}_{i_j} \left| \left[A^{i_1} \ldots A^{i_{j-1}} A^{i_{j+1}} \ldots A^{i_{m+1}} \right] \right|.$$

Das Ergebnis ist ein Vektor aus $\mathbb{Z}^n$, da in der Matrix neben den Zahlen aus A auch Einheitsvektoren des $\mathbb{N}^n$ stehen. Ist $D(\mathcal{A})$ ein Vektor ungleich Null, dessen Komponenten alle das gleiche Vorzeichen haben, teilt man ihn durch den größten gemeinsamen Teiler seiner Komponenten und erhält so einen Vektor aus $S_0(A)$. Tut man dies für alle Teilmengen $\mathcal{A}$, erhält man $S_0(A)$.

Beispiel 4.1 Für die Matrix

$$A = \begin{bmatrix} 1 & 2 & -1 & -1 & -1 \\ 2 & 1 & -1 & -1 & -1 \end{bmatrix}$$

gibt es $\binom{5}{3} = 10$ dreielementige Teilmengen ihrer Spalten. Für die ersten drei Spalten $\mathcal{A} = \{A^1, A^2, A^3\}$ erhält man

$$D(\mathcal{A}) = \begin{vmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \\ 1 & 2 & -1 \\ 2 & 1 & -1 \end{vmatrix} = -1\vec{e}_1 - 1\vec{e}_2 - 3\vec{e}_3 = (-1 \;\; -1 \;\; -3 \; 0 \; 0).$$

Da alle Komponenten das gleiche Vorzeichen haben, ist $(1 \;\; 1 \;\; 3 \;\; 0 \;\; 0)$ ein Vektor aus $S_0(A)$. Nur für $\mathcal{A} = \{A^1, A^2, A^4\}$ und $\mathcal{A} = \{A^1, A^2, A^5\}$ haben ebenfalls alle Komponenten von $D(\mathcal{A})$ das gleiche Vorzeichen. Man erhält

$$S_0(A) = \{(1\;1\;3\;0\;0), (1\;1\;0\;3\;0), (1\;1\;0\;0\;3)\}. \quad \blacksquare$$

Die Vektoren aus $S_0(A)$ bestimmen den Untervektorraum der Lösungen aus $\mathbb{Q}^n$ für die diophantischen Gleichungen. Alle anderen Lösungen aus $\mathbb{Q}^n$ sind dann Linearkombinationen von Vektoren aus $S_0(A)$ mit rationalen Koeffizienten. Die Vektoren aus $S(A)$, d. h.

[1] Die Trägermenge $C(\vec{s})$ eines Vektors $\vec{s}$ ist die Menge der Indizes, deren Komponente einen Wert ungleich Null hat: $C(\vec{s}) = \{i \mid \vec{s}(i) \neq 0\}$.

die Lösungsvektoren aus $\mathbb{N}^n$, sind Linearkombinationen von linear unabhängigen Vektoren aus $S_0(A)$ mit positiven rationalen Koeffizienten(siehe [7]).

$$S(A) = \left\{ \sum_{\vec{s} \in S_0(A)} \lambda_{\vec{s}} \vec{s} \in \mathbb{N}^n \mid \lambda_{\vec{s}} \in \mathbb{Q}^+ \right\}.$$

Nun müssen also noch die restlichen minimalen natürlichzahligen Lösungen aus $\mu S(A)$ in dem von $S_0(A)$ in $\mathbb{Q}^n$ aufgespannten Untervektorraum gesucht werden. Das Problem besteht darin, alle Kombinationen der $\lambda_{\vec{s}}$ zu finden, so daß die Linearkombination ein ganzzahliger Vektor ist. Da man nur die minimalen Lösungen $\mu S(A)$ sucht, genügt es, $0 \leq \lambda_{\vec{s}} < 1$ zu betrachten. Ein Vektor aus einer Linearkombination mit einem $\lambda_{\vec{s}} \geq 1$ ist größer als $\vec{s}$ und daher nicht minimal. (Aber auch Linearkombinationen, bei denen $\lambda_{\vec{s}} < 1$ gilt, können zu nicht minimalen Lösungen führen.) Um das im folgenden beschriebene Verfahren durchführen zu können, werden nur die Linearkombinationen von bezüglich $\mathbb{Q}$ linear unabhängigen Vektoren aus $S_0(A)$ betrachtet. Dies führt allerdings dazu, daß alle Teilmengen von $S_0(A)$ betrachtet werden müssen, die linear unabhängig und bezüglich der Teilmengenordnung maximal sind.

Sei M eine Matrix, deren Spalten k linear unabhängige Vektoren aus $S_0(A)$ sind. Gesucht werden alle $\vec{l} = (\lambda_{M^1} \dots \lambda_{M^k}) \in [0 \dots 1[^k$, so daß $\vec{s} = M\vec{l}$ natürlichzahlig ist. Um diese zu finden, wird M zuerst durch ganzzahlige elementare Zeilenumformungen in die Form $\begin{bmatrix} T \\ 0 \end{bmatrix}$ gebracht, wobei T eine quadratische obere Dreiecksmatrix mit positiven Zahlen auf der Diagonalen ist. Ganzzahlige elementare Umformungen einer Matrix sind die Vertauschung zweier Zeilen, die Multiplikation aller Elemente einer Zeile mit -1 und die Addition einer mit einer ganzen Zahl multiplizierten Zeile zu einer anderen Zeile. Die Transformation einer Matrix mit solchen Umformungen entspricht der Multiplikation mit einer unimodularen Matrix U ($|U| = \pm 1$), also $UM = \begin{bmatrix} T \\ 0 \end{bmatrix}$.

Es gilt $\vec{s} = M\vec{l} \in \mathbb{Z}^n$ genau dann, wenn $T\vec{l} = \vec{x} \in \mathbb{Z}^k$. Da T eine Matrix ist die aus linear unabhängigen Vektoren besteht, ist T invertierbar, d. h. T^{-1} existiert. Durch Umformen erhält man die Gleichung $\vec{l} = T^{-1}\vec{x}$, wobei $\vec{x}$ ein ganzzahliger Vektor ist. Da nun ein Weg existiert, $\vec{l}$ zu berechnen, braucht man noch eine Einschränkung auf eine endliche Menge von zu betrachtenden Vektoren $\vec{x}$, um alle $\vec{l}$ bestimmen zu können. Man erhält diese Einschränkung durch die Begrenzung auf die minimalen Lösungen, d. h. auf $\vec{l} \in [0 \dots 1[^k$. Daher setzt man $\vec{l} = T^{-1}\vec{x} - \lfloor T^{-1}\vec{x} \rfloor$ setzen, wobei $\lfloor x \rfloor$ die streichende Rundung ist. Da $M\vec{l_1} = M(T^{-1}\vec{x})$ ganzzahlig ist (nach Konstruktion von T) und $M\vec{l_2} = M\lfloor T^{-1}\vec{x} \rfloor$ ganzzahlig ist ($\lfloor T^{-1}\vec{x} \rfloor$ und M sind ganzzahlig) ist auch $M\vec{l} = M(\vec{l_1}\vec{l_2})$ ganzzahlig.

Sei $d = |T|$ die Determinate von T, d. h. das Produkt der Zahlen auf der Diagonalen von T und $\vec{d} = (T^1(1) \dots T^k(k)) \in \mathbb{N}^k$ der Vektor der Diagonalen. Dann ist d ein gemeinsames Vielfaches der Nenner in T^{-1} und damit sind alle Elemente in dT^{-1} und $dT^{-1}\vec{x}$ ganzzahlig. Daher gilt $\vec{l} = T^{-1}\vec{x} - \lfloor T^{-1}\vec{x} \rfloor = \frac{1}{d}[dT^{-1}\vec{x}]_d$, wobei $[n]_d$ der positive Rest der ganzzahligen Division von n durch d ist.

Da zu jedem $\vec{x} \in \mathbb{Z}^k$ ein $\vec{x}' \in \mathbb{Z}^k$ mit $0^k <^k \vec{x}' \ll \vec{d}$ existiert, für das $[dT^{-1}\vec{x}]_d = [dT^{-1}\vec{x}']_d$ gilt (siehe [19]), genügt es, um alle $\vec{l}$ zu erhalten, für $\vec{x}$ Vektoren einzusetzen, die $0^k <^k$

$\vec{x}' \ll \vec{d}$ erfüllen, deren Komponenten also alle kleiner sind als die entsprechenden Werte auf der Diagonale von T. Für alle so berechnete $\vec{l}$ ist $M\vec{l}$ eine ganzzahlige Lösung von $A\vec{x} = 0^n$ und berechnet man für alle Mengen von linear unabhängigen Vektoren aus $S_0(A)$ nach diesem Verfahren die neuen Lösungen, erhält man eine Menge, die die restlichen minimalen Lösungen aus $\mu S(A)$ enthält. Da nicht alle der so berechneten Lösungen minimal sind, muß man die nicht minimalen noch aussortieren.

$$\mu S(A) \subseteq S_0(A) \cup \{M\vec{l} \mid \begin{array}{l} \text{die Spalten von } M \text{ sind linear unabhängige} \\ \text{Vektoren aus } S_0(A), \\ UM = \begin{bmatrix} T \\ 0 \end{bmatrix} \text{ mit unimodularem } U, \\ \vec{l} = \frac{1}{d}[dT^{-1}\vec{x}]_d \text{ mit } \vec{x} \ll \vec{d} \text{ und } \vec{x} \in \mathbb{N}^k\} \end{array}$$

Der gesamte Algorithmus ist in Abbildung 4.3 dargestellt. Durch eine geschickte Wahl des Zeitpunkts der Multiplikation und Division mit d kann man im Programm das Rechnen mit Brüchen oder gar Gleitkommazahlen vermeiden. Führt man schon während der Invertierung von T die Multiplikation mit d durch, treten keine Brüche auf, da dT^{-1} ganzzahlig ist. Um bei der Berechnung von $\vec{l}$ Brüche zu vermeiden, führt man die Division durch d erst nach der Multiplikation von $d\vec{l}$ mit M durch. Man berechnet also erst $M(d\vec{l})$ und teilt dann durch d. Das Ergebnis der Division ist ein ganzzahliger Lösungsvektor.

Beispiel 4.2 (Fortsetzung von Beispiel 4.1) Da die drei Vektoren aus $S_0(A)$ linear unabhängig sind, kann man sie direkt zu einer Matrix zusammenfassen. Durch elementare Zeilentransformation oder gleichbedeutend durch Multiplikation mit der unimodularen Matrix U erhält man $\begin{bmatrix} T \\ 0 \end{bmatrix}$:

$$UM = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ -3 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\text{also} \qquad T = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \qquad d = 9 \qquad \text{und} \qquad T^{-1} = \frac{1}{9}\begin{bmatrix} 9 & -3 & -3 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

Für alle $\vec{x} \ll (1\ 3\ 3)$ muß nun $\vec{l}\frac{1}{d}[dT^{-1}\vec{x}]_d$ und danach $\vec{s} = M\vec{l}$ berechnet werden. Für $\vec{x} = (0\ 1\ 0)$, einen dieser acht Vektoren, erhält man

$$\vec{l} = \frac{1}{9}\left[\begin{bmatrix} 9 & -3 & -3 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\right]_9 = \frac{1}{9}\left[\begin{pmatrix} -3 \\ 3 \\ 0 \end{pmatrix}\right]_9 = \frac{1}{9}\begin{pmatrix} 6 \\ 3 \\ 0 \end{pmatrix} \quad \text{und}$$

$$\vec{s} = M\vec{l} = \frac{1}{9}(9\ 9\ 18\ 9\ 0) = (1\ 1\ 2\ 1\ 0).$$

1. $S := \emptyset$; $V := \emptyset$.
2. Für alle $\{A^{i_1}, \ldots, A^{i_{m+1}}\} \subseteq \{A^1, \ldots, A^n\}$
   $D(\mathcal{A}) := \begin{vmatrix} \vec{e}_{i_1} & \ldots & \vec{e}_{i_{m+1}} \\ A^{i_1} & \ldots & A^{i_{m+1}} \end{vmatrix}$;
   Wenn $D(\mathcal{A}) \in \mathbb{N}^n$ oder $-D(\mathcal{A}) \in \mathbb{N}^n$ dann
   $S := S \cup \{D(\mathcal{A}) \div ggt(\{D(\mathcal{A})(1), \ldots, D(\mathcal{A})(n)\})\}$.
3. Für alle maximalen Mengen von linear unabhängigen Vektoren $\{\vec{m}_1, \ldots, \vec{m}_k\} \subseteq S$
   $M := [\vec{m}_1 \ldots \vec{m}_k]$;
   Transformiere durch ganzzahlige elementare Zeilentransformation M zu $\begin{bmatrix} T \\ 0 \end{bmatrix}$ mit T obere Dreiecksmatrix mit positiven Zahlen auf der Diagonalen;
   Berechne T^{-1}, die Inverse von T;
   $d := |T|$; $\vec{d} := (T^1(1) \ldots T^k(k))$;
   Für alle $\vec{x} \in \mathbb{N}^k$ mit $0 <^k \vec{x} \ll \vec{d}$
   $\vec{l} := \frac{1}{d}[dT^{-1}\vec{x}]_d$;
   $V := V \cup \{M\vec{l}\}$.
4. Für alle $\vec{s} \in V$
   Wenn kein $\vec{t}$ in $S \cup V$ existiert mit $\vec{t} <^n \vec{s}$ dann
   $S := S \cup \{\vec{s}\}$.
5. Stopp. S ist $\mu S(A)$, die Menge der minimalen Lösungen.

Abbildung 4.3: *Algorithmus von E. Domenjoud zur Berechnung der minimalen Lösungen* $\mu S(A)$

Sieben der acht so berechneten Lösungen sind minimal. Zusammen mit den Lösungen aus $S_0(A)$ erhält man schließlich

$$\begin{aligned} \mu S(A) = \{ &(1\ 1\ 3\ 0\ 0), (1\ 1\ 0\ 3\ 0), (1\ 1\ 0\ 0\ 3), \\ &(1\ 1\ 2\ 1\ 0), (1\ 1\ 2\ 0\ 1), (1\ 1\ 1\ 2\ 0), \\ &(1\ 1\ 0\ 2\ 1), (1\ 1\ 1\ 0\ 2), (1\ 1\ 0\ 1\ 2), \\ &(1\ 1\ 1\ 1\ 1)\}. \quad \blacksquare \end{aligned}$$

4.2.1 Optimierung der Aufzählung

Der größte Aufwand im Verfahren von Domenjoud liegt im häufigen Durchlaufen der Schleife in Schritt 3, wenn q, die Anzahl der Vektoren in $S_0(A)$, größer ist als p, die Dimension des Lösungsraums. Die häufige Wiederholung ist notwendig, weil die Zeilentransformation von M und die anschließende Invertierung von T nur mit linear unabhängigen Vektoren durchgeführt werden können. Aus diesem Grund müssen diese Schritte für alle maximalen Teilmengen von linear unabhängigen Vektoren aus $S_0(A)$ durchgeführt werden. Dies ist mindestens $\binom{q}{p}$-mal, nämlich für alle p-elementigen Teilmengen $S = \{\vec{s}_1, \ldots, \vec{s}_p\} \subseteq S_0(A)$ notwendig, da mehr als p Vektoren aus $S_0(A)$ immer linear abhängig sind. Ist eine solche Teilmenge aus $S_0(A)$ nicht linear unabhängig, sind noch mehr Schleifendurchläufe

nötig. Statt dieser einen Menge müssen dann wieder alle ihre linear unabhängigen Teilmengen betrachtet werden. Dies sind analog zu oben mindestens p Teilmengen, nämlich alle Teilmengen mit $p-1$ Elementen.

Die in der Schleife in Schritt 3 durchgeführte Operation entspricht der Suche aller ganzzahligen Vektoren in dem von den Vektoren $\{\vec{s}_1, \ldots, \vec{s}_k\}$ aufgespannten Hyperspat

$$P(S) = \left\{ \sum_{i=1}^{k} \lambda_i \vec{s}_i \mid \lambda_i \in [0 \ldots 1[\right\}.$$

Da sich diese Hyperspate für die verschiedenen Teilmengen von $S_0(A)$ überschneiden, werden die gesuchten Lösungen mehrfach aufgezählt.

Die Anzahl der Schleifendurchläufe in Schritt 3 kann am einfachsten verringert werden, indem die Anzahl der Vektoren aus $S_0(A)$, die für diesen Schritt betrachtet werden müssen, verringert wird. Dies kann im allgemeinen Anwendungsfall nur für eine sehr eingeschränkte Menge von Vektoren getan werden. Im speziellen Fall der AC-Unifikation tritt diese Situation aber sehr häufig ein.

Für die AC-Unifikation haben die minimalen Lösungen der diophantischen Gleichungen häufig die Höhe eins, d. h. die Komponenten sind entweder 0 oder 1. Mit so einem Vektor $\vec{m}_i$ mit $h(\vec{m}_i) = 1$ braucht die Transformation in Schritt 3 des Algorithmus nicht durchgeführt zu werden. Dazu wird gezeigt, daß eine damit berechnete Lösung $\vec{s}$ entweder nicht von $\vec{m}_i$ abhängt oder nicht minimal ist. Ist $\vec{l}(i)$, der Koeffizient von $\vec{m}_i$ bei der Linearkombination zur Erzeugung von $\vec{s}$, größer Null, so muß $\vec{m}_i \leq^n \vec{s}$ gelten, da alle anderen Werte der Linearkombination positiv sind, $\vec{s}$ ganzzahlig ist und $\vec{m}_i$ die Höhe eins hat. In diesem Fall ist $\vec{s}$ also nicht minimal. Sonst ist $\vec{l}(i) = 0$, d. h. zur Berechnung von $\vec{s}$ wird $\vec{m}_i$ nicht benötigt und $\vec{s}$ kann in Schritt 3 auch durch $\{\vec{m}_1, \ldots, \vec{m}_k\} \setminus \{\vec{m}_i\}$ berechnet werden. Alle Lösungen in $S_0(A)$ mit Höhe eins können in Schritt 3 also außer acht gelassen werden, und die Berechnung muß nur „Für alle maximalen Mengen von linear unabhängigen Vektoren $\{\vec{m}_1, \ldots, \vec{m}_k\} \subseteq V$ mit $h(\vec{m}_i) > 1$“ durchgeführt werden.

Diese Einschränkung ist sehr speziell und wird im allgemeinen Anwendungsfall nur selten zutreffen. Für die AC-Unifikation treten aber überwiegend Lösungen der Höhe 1 auf, wie in [16] angesprochen wird. Daher kann durch diese Einschränkung die Bearbeitungszeit in vielen Fällen erheblich gesenkt werden (siehe Beispiele im Anhang).

Während mit der vorhergehenden Optimierung die Anzahl der zu betrachtenden Lösungen aus $S_0(A)$ verringert wurde, soll nun ein erster Ansatz vorgestellt werden, mit dem die Anzahl der zu betrachtenden Teilmengen verringert wird, indem die oben angesprochene Überschneidung der Hyperspate vermieden wird. Wenn die Dimension des Lösungsraums der diophantischen Gleichungen $q = 3$ ist, kann damit die Komplexität des Algorithmus verringert werden.

Zuerst werden einige Eigenschaften der Vekoren in $S_0(A)$ aufgezählt, die später benötigt werden. Für zwei Vektoren $\vec{s}, \vec{t} \in S_0(A)$ mit $\vec{s} \neq \vec{t}$ gilt

1. $\vec{s}, \vec{t} \in \mathbb{N}^n$, d. h. die Vektoren in $S_0(A)$ sind natürlichzahlige Vektoren;

2. $\vec{s} \not<^n \vec{t}$ und $\vec{t} \not<^n \vec{s}$, d. h. die Vektoren in $S_0(A)$ sind minimal bezüglich $<^n$;
3. $C(\vec{s}) \not\subset C(\vec{t})$ und $C(\vec{t}) \not\subset C(\vec{s})$, d. h. die Trägermengen der Vektoren in $S_0(A)$ sind minimal bezüglich der Teilmengenbeziehung.
4. $C(\vec{s}) \neq C(\vec{t})$, d. h. die Trägermengen der Vektoren in $S_0(A)$ sind paarweise verschieden.

Die ersten drei Eigenschaften gelten wegen der Definition von $S_0(A)$; die vierte, weil es sonst eine Lösung mit kleinerer Trägermenge gäbe, wie man sich leicht überlegen kann.

Im folgenden sei nun die Dimension des Lösungsraums der diophantischen Gleichungen, d. h. die Dimension bezüglich $\mathbb{Q}$ von $S_0(A)$ ist $q = 3$. Dann gilt, daß mehr als drei Vektoren aus $S_0(A)$ bezüglich $\mathbb{Q}$ immer linear abhängig sind.

Das folgende Lemma zeigt eine wichtige Eigenschaft für die Vektoren in $S_0(A)$. Es besagt, daß ein Vektor $\vec{s}_4 \in S_0(A)$ nicht eine Linearkombination von drei anderen Vektoren $\vec{s}_1, \vec{s}_2, \vec{s}_3 \in S_0(A)$ mit positiven Koeffizienten sein kann.

Lemma 4.1 *Für vier verschiedene Vektoren $\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4 \in S_0(A)$ gibt es eindeutige μ_1, μ_2, μ_3 mit $\vec{s}_4 = \mu_1\vec{s}_1 + \mu_2\vec{s}_2 + \mu_3\vec{s}_3$, $\mu_i < 0$ für genau ein $i \in \{1,2,3\}$ und $\mu_i \neq 0$ für alle $i \in \{1,2,3\}$.*

Beweis: Da $S_0(A)$ die Dimension drei hat, gibt es μ_1, μ_2, μ_3 mit $\vec{s}_4 = \mu_1\vec{s}_1 + \mu_2\vec{s}_2 + \mu_3\vec{s}_3$.

Beweis zu $\mu_i \neq 0$ für alle $i \in \{1,2,3\}$:
Wenn für zwei oder drei i $\mu_i = 0$ gilt, so ist dies ein Widerspruch zur Minimalität der Vektoren in $S_0(A)$ bezüglich $<^n$, weil sonst $\vec{s}_4 = 0^n$ oder $\vec{s}_4 = \mu_j\vec{s}_j$ und damit $\vec{s}_4 <^n \vec{s}_j$, $\vec{s}_4 >^n \vec{s}_j$ oder $\vec{s}_4 = \vec{s}_j$ wäre. Ist genau ein $\mu_i = 0$, z. B. $\mu_3 = 0$ so gilt

$$\vec{s}_4 = \mu_1\vec{s}_1 + \mu_2\vec{s}_2$$

Angenommen $\mu_1 > 0$ und $\mu_2 > 0$. Für die Trägermenge von $\vec{s}_4$ gilt dann $C(\vec{s}_1) \subset C(\vec{s}_4)$, weil $C(\vec{s}_1) \neq C(\vec{s}_2)$. Dies widerspricht der Voraussetzung, daß die Vektoren in $S_0(A)$ eine minimale Trägermenge haben. Angenommen $\mu_1, \mu_2 < 0$. Dies führt zu einem Widerspruch zu der Voraussetzung, daß $\vec{s}_4 \in \mathbb{N}^n$. Angenommen $\mu_1 > 0$ und $\mu_2 < 0$. Weil die Trägermengen von $\vec{s}_1$ und $\vec{s}_2$ verschieden und bezüglich der Teilmengenordnung unvergleichbar sind, existiert ein j mit $\vec{s}_1(j) = 0$ und $\vec{s}_2(j) > 0$ und damit $\vec{s}_4(j) < 0$. Dies ist ein Widerspruch zu der Voraussetzung, daß $\vec{s}_4 \in \mathbb{N}^n$. Der Fall $\mu_1 < 0$ und $\mu_2 > 0$ ist analog zum letzten Fall und damit ist gezeigt, daß in allen Fällen ein Widerspruch auftritt. Also gilt $\mu_i \neq 0$ für alle $i \in \{1,2,3\}$.

Beweis zu μ_1, μ_2, μ_3 sind eindeutig:
Angenommen es gibt ν_1, ν_2, ν_3 mit $\vec{s}_4 = \nu_1\vec{s}_1 + \nu_2\vec{s}_2 + \nu_3\vec{s}_3$ und $\{\nu_1, \nu_2, \nu_3\} \neq \{\mu_1, \mu_2, \mu_3\}$. Dann gilt $(\mu_1 - \nu_1)\vec{s}_1 + (\mu_2 - \nu_2)\vec{s}_2 + (\mu_3 - \nu_3)\vec{s}_3 = 0^n$. Ist z. B. $\mu_1 \neq \nu_1$ so folgt aus

$$\vec{s}_1 = \frac{\nu_2 - \mu_2}{\mu_1 - \nu_1}\vec{s}_2 + \frac{\nu_3 - \mu_3}{\mu_1 - \nu_1}\vec{s}_3 \qquad \vec{s}_4 = \left(\mu_2 + \mu_1\frac{\nu_2 - \mu_2}{\mu_1 - \nu_1}\right)\vec{s}_2 + \left(\mu_3 + \mu_1\frac{\nu_3 - \mu_3}{\mu_1 - \nu_1}\right)\vec{s}_3.$$

Dies ist ein Widerspruch zum schon bewiesenen Teil des Lemmas, daß $\mu_i \neq 0$ für alle μ_i.

Beweis zu $\mu_i < 0$ für genau ein $i \in \{1,2,3\}$:
Angenommen $\mu_i > 0$ für alle $i \in \{1,2,3\}$. Wie oben führt dies zu einem Widerspruch zu der Voraussetzung, daß $\vec{s}_4$ eine minimale Trägermenge hat.

Angenommen $\mu_i < 0$ für genau zwei $i \in \{1,2,3\}$, z. B. $\mu_1 > 0$ und $\mu_2, \mu_3 < 0$. Dann gilt die gleiche Argumentation, weil $\mu_1\vec{s}_1 = (-\mu_2)\vec{s}_2 + (-\mu_3)\vec{s}_3 + \vec{s}_4$ mit $\mu_1, (-\mu_2), (-\mu_3) > 0$.

Angenommen $\mu_i < 0$ für alle $i \in \{1,2,3\}$. Dann gilt $\vec{s}_4 \notin \mathbb{N}^n$, da $\vec{s}_1, \vec{s}_2, \vec{s}_3 \in \mathbb{N}^n$. Dies widerspricht der Voraussetzung, daß die Vektoren in $S_0(A)$ natürlichzahlige Vektoren sind.

Es bleibt also nur die Möglichkeit, daß $\mu_i < 0$ für genau ein $i \in \{1,2,3\}$. □

Eine Folgerung aus dem Lemma ist, daß drei Vektoren aus $S_0(A)$ immer linear unabhängig sind, weil dies sonst zu einem Widerspruch zu $\mu_i \neq 0$ führen würde.

Nun soll für den Fall, daß $S_0(A)$ genau vier Vektoren enthält, gezeigt werden, wie die Anzahl der zu betrachtenden Mengen von linear unabhängigen Vektoren von vier auf zwei halbiert werden kann. Sei also $S_0(A) = \{\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4\}$. Dann gilt bei einer entsprechenden Numerierung nach Lemma 4.1

$$\mu_1\vec{s}_1 + \mu_2\vec{s}_2 = \mu_3\vec{s}_3 + \vec{s}_4 \quad \text{mit } \mu_i > 0. \tag{4.1}$$

Im Algorithmus aus Abbildung 4.3 müssen nun alle vier Mengen $P_i = P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4\} \setminus \{\vec{s}_i\})$ mit $i \in \{1,2,3,4\}$ nach ganzzahligen Vektoren durchsucht werden. In Abbildung 4.4 soll die Lage der vier Vektoren im dreidimensionalen Raum verdeutlicht werden.

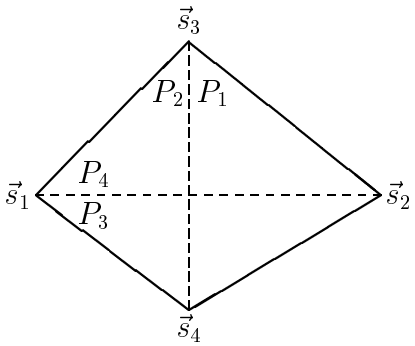

Abbildung 4.4: *Schnitt durch die von $\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4$ aufgespannten Spate.*

Die Abbildung zeigt einen Schnitt durch den Raum und die aufgespannten Spate. An den Ecken des Vierecks sind die Schnittpunkte der vier Vektoren mit der Ebene. Wegen Gleichung 4.1 müssen die vier Vektoren genau wie abgebildet angeordnet sein. Wie man sieht, überschneiden sich die vier Spate und in der Schnittfläche gilt $P_1 \cup P_2 = P_3 \cup P_4$. Im allgemeinen gilt dies jedoch nicht. Es gilt aber, wie das folgende Lemma zeigt, daß ein ganzzahliger Vektor, der in $P_1 \cup P_2$ und nicht in $P_3 \cup P_4$ liegt, nicht minimal bezüglich $<^n$ sein kann. Dies bedeutet, daß alle minimalen Vektoren in $P_1 \cup P_2$ auch in $P_3 \cup P_4$

sind. Man braucht also nur P_3 und P_4 zu durchsuchen, um alle minimalen, ganzzahligen Vektoren in $P_1 \cup P_2 \cup P_3 \cup P_4$ zu erhalten.

Lemma 4.2

Aus $\vec{s} \in P(\{\vec{s}_2, \vec{s}_3, \vec{s}_4\})$ *und* $\mu_1\vec{s}_1 + \mu_2\vec{s}_2 = \mu_3\vec{s}_3 + \vec{s}_4$ *mit* $\mu_i > 0$ *folgt*

$\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3\})$ *oder* $\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{s}_4\})$ *oder* $\vec{s}$ *nicht minimal zu* $\{\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4\}$.

Beweis: Aus $\vec{s} \in P_1 = P(\{\vec{s}_2, \vec{s}_3, \vec{s}_4\})$, folgt

$$\vec{s} = \lambda_2\vec{s}_2 + \lambda_3\vec{s}_3 + \lambda_4\vec{s}_4 \quad \text{mit } \lambda_i \in [0 \ldots 1[.$$

Setzt man

$$\vec{s}_4 = \mu_1\vec{s}_1 + \mu_2\vec{s}_2 - \mu_3\vec{s}_3 \qquad \text{bzw.} \qquad \vec{s}_3 = \frac{\mu_1}{\mu_3}\vec{s}_1 + \frac{\mu_2}{\mu_3}\vec{s}_2 - \frac{1}{\mu_3}\vec{s}_4$$

in diese Gleichung ein, so erhält man

$$\begin{aligned} \vec{s} &= \lambda_4\mu_1\vec{s}_1 + (\lambda_2 + \lambda_4\mu_2)\vec{s}_2 + (\lambda_3 - \lambda_4\mu_3)\vec{s}_3 && (4.2)\\ &= \lambda_3\frac{\mu_1}{\mu_3}\vec{s}_1 + (\lambda_2 + \lambda_3\frac{\mu_2}{\mu_3})\vec{s}_2 + (\lambda_4 - \lambda_3\frac{1}{\mu_3})\vec{s}_4 && (4.3)\end{aligned}$$

I. Fall: $\lambda_3 \geq \lambda_4\mu_3$
In diesem Fall sind alle drei Koeffizienten in 4.2 größer oder gleich Null. Sind alle drei kleiner 1, liegt $\vec{s}$ in P_4; ist einer größer oder gleich 1, ist $\vec{s}$ größer als der zugehörige Vektor und damit nicht minimal.

II. Fall: $\lambda_3 < \lambda_4\mu_3$, also $\lambda_4 > \lambda_3\frac{1}{\mu_3}$
In diesem Fall sind die Koeffizienten in 4.3 alle positiv. Analog zum ersten Fall liegt $\vec{s}$ dann in P_3 oder ist nicht minimal.

Aus den beiden Fällen folgt die Behauptung. □

Da für $\vec{s} \in P_2$ das Lemma analog gilt und umgekehrt auch aus $\vec{s} \in P_3$ oder $\vec{s} \in P_4$ folgt, daß $\vec{s} \in P_1$ oder $\vec{s} \in P_2$ oder $\vec{s}$ nicht minimal ist, gilt für alle minimalen, ganzzahligen Vektoren $\vec{s} \in P_1 \cup P_2 \cup P_3 \cup P_4$, daß sowohl $\vec{s} \in P_1 \cup P_2$ als auch $\vec{s} \in P_3 \cup P_4$ gilt. Man braucht also nur P_1 und P_2 oder P_3 und P_4 zu durchsuchen.

Nun soll dieses Verfahren auf Fälle übertragen werden, in denen $S_0(A)$ mehr als vier Vektoren enthält. Man wählt dazu drei Vektoren $\vec{s}_1, \vec{s}_2, \vec{s}_3 \in S_0(A)$ aus. Für alle anderen Vektoren $\vec{s} \in S_0(A)$ gilt nach Lemma 4.1

$$\vec{s} = \mu_1\vec{s}_1 + \mu_2\vec{s}_2 + \mu_3\vec{s}_3 \quad \text{mit } \mu_i < 0 \text{ für genau ein } i \in \{1, 2, 3\}.$$

Die übrigen Vektoren aus $S_0(A)$ werden nach Lemma 4.1 eindeutig in die drei Mengen S_1', S_2' und S_3' aufgeteilt. Sie sind definiert durch

$$S_i' = \{\vec{s} \in S_0(A) \setminus \{\vec{s}_1, \vec{s}_2, \vec{s}_3\} \mid \vec{s} = \nu_1\vec{s}_1 + \nu_2\vec{s}_2 + \nu_3\vec{s}_3 \text{ mit } \nu_i < 0 \text{ und } \nu_j > 0 \text{ für } j \neq i\}.$$

Außerdem wird $S_i = S_i' \cup \{\vec{s}_1, \vec{s}_2, \vec{s}_3\} \setminus \{\vec{s}_i\}$ definiert. Es gilt $S_1' \mathring{\cup} S_2' \mathring{\cup} S_3' \mathring{\cup} \{\vec{s}_1, \vec{s}_2, \vec{s}_3\} = S_0(A)$. Berechnet man alle ganzzahligen Vektoren aus $P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3\})$ und die in $P(S_1)$, $P(S_2)$ und $P(S_3)$, so erhält man nach dem folgenden Lemma alle ganzzahligen Vektoren in $P(S_0(A))$.

Lemma 4.3 *Für alle $\vec{t}_1, \vec{t}_2, \vec{t}_3 \in S_0(A)$ und $\vec{s} \in P(\{\vec{t}_1, \vec{t}_2, \vec{t}_3\})$ gilt $\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3\})$ oder $\vec{s} \in P(S_i)$ für ein $i \in \{1,2,3\}$ oder $\vec{s}$ ist bezüglich $>^n$ nicht minimal zu $S_0(A)$.*

Beweis: Das Lemma wird durch eine Fallunterscheidung nach der Anzahl der Vektoren aus $\{\vec{t}_1, \vec{t}_2, \vec{t}_3\}$, die in $\{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$ sind, bewiesen. Dabei seien $\vec{t}_1, \vec{t}_2, \vec{t}_3$ und $\vec{s}_1, \vec{s}_2, \vec{s}_3$ in Fall II und III so numeriert, daß $\vec{t}_1 = \vec{s}_1$ bzw. $\vec{t}_1 = \vec{s}_1$ und $\vec{t}_2 = \vec{s}_2$.

I. Fall: $\{\vec{t}_1, \vec{t}_2, \vec{t}_3\} = \{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$
Es folgt sofort $\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3\})$.

II. Fall: $\vec{t}_1 = \vec{s}_1$, $\vec{t}_2 = \vec{s}_2$, $\vec{t}_3 \notin \{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$

1. Ist $\vec{t}_3 \in S_1'$, so gilt $\vec{t}_3 = \nu_1\vec{s}_1 + \nu_2\vec{s}_2 + \nu_3\vec{s}_3$ mit $\nu_1 < 0$. Nach Lemma 4.2 folgt aus

$$\vec{s} \in P(\{\vec{t}_3, \vec{s}_1, \vec{s}_2\} \quad \text{und} \quad \vec{t}_3 + (-\nu_1)\vec{s}_1 = \nu_2\vec{s}_2 + \nu_3\vec{s}_3 \text{ mit } (-\nu_1), \nu_2, \nu_3 > 0$$

$$\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3\}) \text{ oder } \vec{s} \in P(\{\vec{t}_3, \vec{s}_2, \vec{s}_3\}) \text{ oder } \vec{s} \text{ nicht minimal.}$$

   Daraus folgt die Behauptung, da $\{\vec{s}_2, \vec{s}_3, \vec{t}_3\} \subseteq S_1$.

2. Ist $\vec{t}_3 \in S_2'$, so folgt die Behauptung analog zu 1.

3. Ist $\vec{t}_3 \in S_3'$, so gilt $\{\vec{t}_1, \vec{t}_2, \vec{t}_3\} \subseteq S_3$ und damit $\vec{s} \in P(S_3)$.

III. Fall: $\vec{t}_1 = \vec{s}_1$, $\vec{t}_2, \vec{t}_3 \notin \{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$
Nach Lemma 4.1 gelten

$$\begin{aligned} \vec{t}_2 &= \mu_1\vec{s}_1 + \mu_2\vec{s}_2 + \mu_3\vec{t}_3 \quad \text{mit genau einem } \mu_i < 0; \\ \vec{t}_3 &= \nu_1\vec{s}_1 + \nu_2\vec{s}_2 + \nu_3\vec{s}_3 \quad \text{mit genau einem } \nu_i < 0. \end{aligned}$$

Durch Einsetzen erhält man

$$\vec{t}_2 = (\mu_1 + \mu_3\nu_1)\vec{s}_1 + (\mu_2 + \mu_3\nu_2)\vec{s}_2 + (\mu_3\nu_3)\vec{s}_3. \tag{4.4}$$

Nun folgt eine Fallunterscheidung, in welchen S_i sich $\vec{t}_2$ und $\vec{t}_3$ befinden.

1. $\vec{t}_2, \vec{t}_3 \in S_1'$
   Nach der Definition von S_1' gilt dann $\nu_3 > 0$ und $\mu_3\nu_3 > 0$. Daraus folgt $\mu_3 > 0$ also entweder $\mu_1 < 0$ oder $\mu_2 < 0$. Sei also $\mu_1 < 0$ (für $\mu_2 < 0$ gilt der Beweis analog). Dann folgt nach Lemma 4.2 aus

$$\vec{s} \in P(\{\vec{s}_1, \vec{t}_2, \vec{t}_3\}) \quad \text{und} \quad \vec{t}_2 + (-\mu_1)\vec{s}_1 = \mu_2\vec{s}_2 + \mu_3\vec{t}_3 \text{ mit } (-\mu_1), \mu_2, \mu_3 > 0$$

$$\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{t}_3\}) \text{ oder } \vec{s} \in P(\{\vec{s}_2, \vec{t}_2, \vec{t}_3\}) \text{ oder } \vec{s} \text{ nicht minimal.}$$

   Daraus folgt die Behauptung, da $\{\vec{s}_2, \vec{t}_2, \vec{t}_3\} \subseteq S_1$ und für $\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{t}_3\})$ die Behauptung schon in Fall II bewiesen wurde.

2. $\vec{t}_2, \vec{t}_3 \in S_2'$
   Da $\vec{s}_1 = \vec{t}_1 \in S_2$ folgt $\vec{s} \in P(S_2)$.

3. $\vec{t}_2 \in S_1'$, $\vec{t}_3 \in S_3'$
   Nach der Definition von S_1' und S_3' gilt dann $\mu_3\nu_3 > 0$ und $\nu_3 < 0$. Daraus folgt $\mu_3 < 0$ also $\mu_1, \mu_2 > 0$. Dann folgt nach Lemma 4.2 aus

$$\vec{s} \in P(\{\vec{s}_1, \vec{t}_2, \vec{t}_3\}) \quad \text{und} \quad \vec{t}_2 + (-\mu_3)\vec{t}_3 = \mu_1\vec{s}_1 + \mu_2\vec{s}_2 \text{ mit } \mu_1, \mu_2, (-\mu_3) > 0$$

$$\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{t}_2\}) \text{ oder } \vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{t}_3\}) \text{ oder } \vec{s} \text{ nicht minimal.}$$

   Daraus folgt die Behauptung, da für $\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{t}_2\})$ und $\vec{s} \in P(\{\vec{s}_1, \vec{s}_2, \vec{t}_3\})$ die Behauptung schon in Fall II bewiesen wurde.

4. $\vec{t}_2 \in S_2'$, $\vec{t}_3 \in S_3'$
   In diesem Fall gilt der Beweis aus 3, da auch hier $\nu_3 < 0$ und $\mu_3\nu_3 > 0$ gelten.

Die Fälle $\vec{t}_2, \vec{t}_3 \in S_3'$ und $\vec{t}_2 \in S_1'$, $\vec{t}_3 \in S_2'$ sind analog zu 2 bzw. 3 zu beweisen, da sich die verwendeten Eigenschaften von Vektoren aus S_2' und S_3' genau analog verhalten.

IV. Fall: $\vec{t}_1, \vec{t}_2, \vec{t}_3 \notin \{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$
Nach Lemma 4.1 gilt

$$\vec{s}_1 = \mu_1\vec{t}_1 + \mu_2\vec{t}_2 + \mu_3\vec{t}_3 \quad \text{mit genau einem } \mu_i < 0$$

Ist $\mu_1 < 0$ so folgt nach Lemma 4.2 aus

$$\vec{s} \in P(\{\vec{t}_1, \vec{t}_2, \vec{t}_3\}) \quad \text{und} \quad \frac{1}{\mu_3}\vec{s}_1 + \left(-\frac{\mu_1}{\mu_3}\right)\vec{t}_1 = \frac{\mu_2}{\mu_3}\vec{t}_2 + \vec{t}_3 \text{ mit } \frac{1}{\mu_3}, \left(-\frac{\mu_1}{\mu_3}\right), \frac{\mu_2}{\mu_3} > 0$$

$$\vec{s} \in P(\{\vec{s}_1, \vec{t}_1, \vec{t}_2\}) \text{ oder } \vec{s} \in P(\{\vec{s}_1, \vec{t}_1, \vec{t}_3\}) \text{ oder } \vec{s} \text{ nicht minimal.}$$

Daraus folgt die Behauptung, da für $\vec{s} \in P(\{\vec{s}_1, \vec{t}_1, \vec{t}_2\})$ und $\vec{s} \in P(\{\vec{s}_1, \vec{t}_1, \vec{t}_2\})$ die Behauptung schon in Fall III bewiesen wurde. Für $\mu_2 < 0$ und $\mu_3 < 0$ gilt der Beweis analog.

Mit den gezeigten vier Fällen folgt die Behauptung. □

Lemma 4.3 zeigt wie die Anzahl der zu betrachtenden Teilmengen von $S_0(A)$ gesenkt werden kann und liefert auch die Struktur für einen Algorithmus. Nach der Berechnung von $S_0(A)$ wählt man drei Vektoren $\{\vec{s}_1, \vec{s}_2, \vec{s}_3\} \subseteq S_0(A)$ aus und verteilt die restlichen Vektoren auf S_1', S_2' und S_3'. Dann berechnet man die ganzzahligen Vektoren in $P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3\})$ und wendet die gleiche Methode auf S_1, S_2 und S_3 an. Wenn eine dieser drei Mengen nur zwei Vektoren enthält, braucht sie nicht weiter betrachtet zu werden, da dann $P(S_i) \subset P(\{\vec{s}_1, \vec{s}_2, \vec{s}_3\})$ weil $S_i \subset \{\vec{s}_1, \vec{s}_2, \vec{s}_3\}$. Ist $|S_i| = q_i$ und $|S_0(A)| = q = q_1+q_2+q_3-3$, so gilt für a_q, die Anzahl der nach der Methode aus Lemma 4.3 zu bearbeitenden Teilmengen, $a_2 = 0$ und $a_q = a_{q_1} + a_{q_2} + a_{q_3} + 1$. Die Lösung für diese rekursive Formel ist $a_q = q - 2$. Im Algorithmus in Abbildung 4.3 müssen $\binom{q}{3}$ Teilmengen bearbeitet werden, da alle dreielementigen Teilmengen von $S_0(A)$ betrachtet werden müssen. Durch das Verfahren aus Lemma 4.3 kann die Anzahl der zu bearbeitenden Teilmengen also von $\binom{q}{3}$ auf $q - 2$ gesenkt werden.

4.2.2 Bedingungen der AC-Unifikation

Auch für dieses Verfahren soll versucht werden, die Bedingungen 3.1 und 3.2 einzuhalten.

Bedingung 3.1, die besagt, daß eine Spalte, der ein Fremdterm zugeordnet ist, keinen Wert größer 1 enthalten darf, ist nur schwer während des Algorithmus zu testen. Ein einfaches Gegenbeispiel zeigt, daß die Vektoren in $S_0(A)$, die diese Bedingung nicht erfüllen, in Schritt 3 nicht weggelassen werden dürfen.

Beispiel 4.3 Für $A = [1\ 1\ -2]$ ist die Lösungsmenge $S_0(A) = \{(2\ 0\ 1), (0\ 2\ 1)\}$. Sind den ersten beiden Spalten Fremdterme zugeordnet, so erfüllen beide Vektoren nicht Bedingung 3.1. Die in Schritt 3 des Algorithmus aus diesen beiden Vektoren entstandene Lösung (1 1 1) erfüllt sie aber und muß dem Algorithmus zur elementaren AC-Unifikation als Ergebnis geliefert werden. ■

Damit ist die wirkungsvollste Verbesserung, die Verringerung der Vektoren in V, nicht möglich. Während der Berechnungen in Schritt 3 selbst ist eine wirkliche Verbesserung nur schwer vorstellbar. Zwar ist es möglich, für die Komponenten von $\vec{l}$ Maximalwerte anzugeben (im obigen Beispiel wären das für die beiden Vektoren jeweils 1/2), aber eine wirkliche Verbesserung ist nur durch eine Einschränkung der zu betrachtenden $\vec{x}$ denkbar. Diese ist aber wegen der komplizierten Berechnung von $\vec{l}$ (positive und negative Werte in T^{-1}, Bildung des Divisionsrestes) wohl kaum möglich. Die Lösungen, die Bedingung 3.1 verletzen, können also erst nach ihrer Berechnung aussortiert werden.

Bedingung 3.2 läßt sich einfacher einhalten. Nach ihr dürfen alle Lösungen außer acht gelassen werden, bei denen die Fremdterme, die den Spalten mit einem Wert größer 0 zugewiesen sind, nicht unifizierbar sind. Benutzt man einen Vektor $\vec{m}_i$, der Bedingung 3.2 nicht erfüllt, in Schritt 3 zur Berechnung von $\vec{l}$ und ist die zugehörige Komponente $\vec{l}(i) > 0$, so erfüllt auch das Ergebnis der Linearkombination $\vec{s} = M\vec{l}$ die Bedingung 3.2 nicht. Ist $\vec{l}(i) = 0$, so hängt das Ergebnis nicht von diesem Vektor ab. Es ist also möglich, alle Vektoren aus $S_0(A)$, die Bedingung 3.2 nicht erfüllen, schon vor Schritt 3 wegzulassen. So kann die in Schritt 3 zu betrachtende Menge von Vektoren verkleinert werden, was die Berechnung erheblich beschleunigt.

In Abbildung 4.5 ist der verbesserte Algorithmus abgebildet.

1. $S := \emptyset$; $V := \emptyset$.
2. Für alle $\{A^{i_1}, \ldots, A^{i_{m+1}}\} \subseteq \{A^1, \ldots, A^n\}$
   $D(\mathcal{A}) := \begin{vmatrix} \vec{e}_{i_1} & \ldots & \vec{e}_{i_{m+1}} \\ A^{i_1} & \ldots & A^{i_{m+1}} \end{vmatrix}$;
   Wenn $D(\mathcal{A}) \in \mathbb{N}^n$ oder $-D(\mathcal{A}) \in \mathbb{N}^n$ und
   $f_i = f_j$ **für alle** i **und** j **mit** $f_i, f_j \in \Sigma$ **und** $D(\mathcal{A})(i), D(\mathcal{A})(j) > 0$ dann
   $S := S \cup \{D(\mathcal{A}) \div ggt(\{D(\mathcal{A})(1), \ldots, D(\mathcal{A})(n)\})\}$.
3. Für alle maximalen Mengen von linear unabhängigen Vektoren
   $\{\vec{m}_1, \ldots, \vec{m}_k\} \subseteq V$ mit $h(\vec{m}_i) > 1$
   $M := [\vec{m}_1 \ldots \vec{m}_k]$;
   Transformiere durch ganzzahlige elementare Zeilentransformation M zu $\begin{bmatrix} T \\ 0 \end{bmatrix}$
   mit T obere Dreiecksmatrix mit positiven Zahlen auf der Diagonalen;
   Berechne T^{-1}, die Inverse von T;
   $d := |T|$; $\vec{d} := (T^1(1) \ldots T^k(k))$;
   Für alle $\vec{x} \in \mathbb{N}^k$ mit $0 <^k \vec{x} \ll \vec{d}$
   $\vec{l} := \frac{1}{d}[dT^{-1}\vec{x}]_d$;
   $V := V \cup \{M\vec{l}\}$.
4. Für alle $\vec{s} \in V$
   Wenn kein $\vec{t}$ in $S \cup V$ existiert mit $\vec{t} <^n \vec{s}$ und
   $\vec{s}(i) < 2$ für alle i mit $f_i \in \Sigma$ und
   $f_i = f_j$ für alle i und j mit $f_i, f_j \in \Sigma$ und $\vec{s}(i), \vec{s}(j) > 0$ dann
   $S := S \cup \{\vec{s}\}$.
5. Stopp. S ist $\mu S(A)$, die Menge der minimalen Lösungen.

Abbildung 4.5: *Algorithmus von E. Domenjoud zur Berechnung der minimalen Lösungen $\mu S(A)$ mit Optimierungen*

4.3 Verfahren von Pottier

Das Verfahren von Pottier [19] ist das komplizierteste der drei hier vorgestellten. Bei ihm werden die diophantischen Gleichungen in Polynomen kodiert. Mit einem Vervollständigungsverfahren erhält man eine Gröbner-Basis für das von diesen Polynomen gebildete Ideal. Die Gröbner-Basis liefert ein Regelsystem, mit dem alle Lösungen aus $S(A)$ mit aufsteigender Länge aufgezählt werden können. Die Regeln des Regelsystems entsprechen Vektoren des $\mathbb{Z}^n$ und generieren den Untervektorraum des $\mathbb{Z}^n$, der die Lösungen der diophantischen Gleichungen enthält.

4.3.1 Kodierung der diophantischen Gleichungen in Polynomen

Die Idee beim Verfahren von Pottier besteht darin Gleichungen der Form

$$A\vec{y} + \vec{z} = \begin{pmatrix} t \\ \vdots \\ t \end{pmatrix} \text{ mit } \begin{pmatrix} t \\ \vdots \\ t \end{pmatrix}, \vec{z} \in \mathbb{N}^m \text{ und } \vec{y} \in \mathbb{N}^n$$

durch Polynome über einem Ring $K[T, Z_1, \ldots, Z_m, Y_1, \ldots, Y_n]$ zu repräsentieren. Zur Vereinfachung der Schreibweise ist $Z^{\vec{a}} = Z_1^{\vec{a}(1)} \ldots Z_m^{\vec{a}(m)}$ und $Y^{\vec{b}} = Y_1^{\vec{b}(1)} \ldots Y_n^{\vec{b}(n)}$ für $\vec{a} \in \mathbb{N}^m$ und $\vec{b} \in \mathbb{N}^n$. Für einen Vektor $\vec{c} \in \mathbb{Z}^k$ ist $\vec{c}^+$ (bzw. $\vec{c}^-$) der Vektor, der die positiven (bzw. den Betrag der negativen) Komponenten von $\vec{c}$ enthält, d. h. $\vec{c} = \vec{c}^+ - \vec{c}^-$ und $\vec{c}^+, \vec{c}^- \in \mathbb{N}^k$.

Alle Polynome des folgenden Algorithmus bestehen aus zwei Monomen mit den Koeffizienten 1 und -1, haben die Form $T^t Z^{\vec{z}^+} Y^{\vec{y}^+} - Z^{\vec{z}^-} Y^{\vec{y}^-}$ mit $\vec{z} \in \mathbb{N}^m$, $\vec{y} \in \mathbb{N}^n$ und erfüllen die Bedingung $A\vec{y} + \vec{z} = t^m$. Da in den Polynomen nur positive Exponenten erlaubt sind, müssen die positiven und negativen Komponenten der Vektoren getrennt werden und in zwei verschiedenen Monomen repräsentiert werden.

Als erstes wird für jede Spalte A^j von A ein Polynom aufgestellt. Dies hat die Form

$$P_j = Z^{A^{j+}} - Y_j Z^{A^{j-}} \text{ für } j \in \{1, \ldots, m\}$$

und stellt die Gleichung $A(-\vec{e}_j) + A^j = 0^m$ dar. Damit die Vervollständigung arbeiten kann, benötigt man noch ein zusätzliches Polynom:

$$P_0 = T Z_1 \ldots Z_m - 1.$$

Diese $m + 1$ Polynome bilden das Ideal $\mathcal{I}$. Gesucht wird jetzt die Spur auf dem Ring $K[Y_1, \ldots, Y_n]$, d. h. das Unterideal $\mathcal{J}$, das die Polynome enthält, die kein Z_i oder T enthalten. Dies sind die Polynome $T^t Z^{\vec{z}^+} Y^{\vec{y}^+} - Z^{\vec{z}^-} Y^{\vec{y}^-}$ mit $t = 0$ und $\vec{z} = 0^m$. Nach der oben genannten Gleichung, die die Polynome des Ideals $\mathcal{I}$ erfüllen, ist daher $A\vec{y} = 0^n$. Die Polynome des Ideals $\mathcal{J}$ entsprechen also den ganzzahligen Lösungen der diophantischen Gleichungen.

Der Grad eines Monoms ist die Summe seiner Exponenten, also $deg(T^t Z^{\vec{z}} Y^{\vec{y}}) = t + l(\vec{z}) + l(\vec{y})$. Auf zwei Monomen $m_1 = T^{t_1} Z^{\vec{z}_1} Y^{\vec{y}_1}$ und $m_2 = T^{t_2} Z^{\vec{z}_2} Y^{\vec{y}_2}$ ist die Ordnung $>_m$ definiert durch

$$\begin{array}{lll} m_1 >_m m_2 & \text{genau dann, wenn} & t_1 > t_2 \text{ oder} \\ & & t_1 = t_2 \text{ und } \vec{z}_1 >_{\text{lex}} \vec{z}_2 \text{ oder} \\ & & t_1 = t_2, \vec{z}_1 = \vec{z}_2 \text{ und } deg(m_1) > deg(m_2) \text{ oder} \\ & & t_1 = t_2, \vec{z}_1 = \vec{z}_2, deg(m_1) = deg(m_2) \text{ und } \vec{y}_1 >_{\text{lex}} \vec{y}_2. \end{array}$$

Um eine Basis für $\mathcal{J}$ zu erhalten, wird durch Vervollständigung mit der oben definierten Reduktionsordnung $>_m$ eine Gröbner-Basis $B_{\mathcal{I}}$ für $\mathcal{I}$ bestimmt. Wegen der benutzten Reduktionsordnung ist dann $B_{\mathcal{J}}$, die Teilmenge von Polynomen aus $B_{\mathcal{I}}$, in denen kein Z_i oder T vorkommt, eine Gröbner-Basis für $\mathcal{J}$. Sei $B_{\mathcal{J}} = \{Y^{\vec{a}_1} - Y^{\vec{b}_1}, \ldots, Y^{\vec{a}_p} - Y^{\vec{b}_p}\}$ diese Basis, wobei $Y^{\vec{a}_i}$ jeweils das größere Monom bezüglich $>_m$ ist. Dies ist ohne Beschränkung der Allgemeinheit möglich, da P in $B_{\mathcal{J}}$ durch $-P$ ausgewechselt werden kann.

Beispiel 4.4 Für die Matrix

$$A = \begin{bmatrix} 1 & 2 & -1 & -1 & -1 \\ 2 & 1 & -1 & -1 & -1 \end{bmatrix}$$

werden die Polynome

$$\begin{array}{ll} P_0 = TZ_1Z_2 - 1 & P_3 = 1 - Y_3Z_1Z_2 \\ P_1 = Z_1{Z_2}^2 - Y_1 & P_4 = 1 - Y_4Z_1Z_2 \\ P_2 = {Z_1}^2Z_2 - Y_2 & P_5 = 1 - Y_5Z_1Z_2 \end{array}$$

aufgestellt. Durch die Vervollständigung erhält man

$$\begin{array}{rcl} B_{\mathcal{I}} & = & \{Z_1 - Y_2Y_4, Z_2 - Y_1Y_4, T - Y_4\} \cup B_{\mathcal{J}} \text{ mit} \\ B_{\mathcal{J}} & = & \{Y_1Y_2{Y_4}^3 - 1, Y_3 - Y_4, Y_2 - Y_4\}. \quad \blacksquare \end{array}$$

4.3.2 Aufzählen der Lösungen

Die Vektoren $\vec{s}_i = \vec{a}_i - \vec{b}_i \in \mathbb{Z}^n$ für $i \in \{1, \ldots, p\}$ sind Lösungen der diophantischen Gleichungen und sie generieren den Untervektorraum des $\mathbb{Z}^n$, der alle Lösungen enthält. In diesem Untervektorraum müssen nun noch die minimalen natürlichzahligen Lösungen gesucht werden. Dazu wird das Regelsystem R mit den Regeln $Y^{\vec{a}_i} \to Y^{\vec{b}_i}$ für $i \in \{1, \ldots, p\}$ aufgestellt. Die Reduktionsrelation zu R wird mit $\longrightarrow$ bezeichnet und ihre reflexive, transitive Hülle mit $\stackrel{*}{\longrightarrow}$.

Weil $B_{\mathcal{J}}$ eine Gröbner-Basis ist, gilt $\vec{x} \in \mathbb{N}^n$ genau dann in $S(A)$, wenn $Y^{\vec{x}} \stackrel{*}{\longrightarrow} 1$. Wendet man die Regeln aus R in umgekehrter Reihenfolge an, erhält man einen Algorithmus, der beginnend mit $Y^{0^n} = 1$ alle Lösungen in $S(A)$ aufzählt. Sei $\longrightarrow^{-1}$ die inverse Reduktionsrelation zu $\longrightarrow$. Dann erhält man alle Lösungen aus $S(A)$ indem man den Baum aufbaut,

den man durch Reduktion von 1 mit $\longrightarrow^{-1}$ erhält. Da die Reduktionsrelation entsprechend gewählt ist, werden die Monome mit aufsteigendem Grad bzw. die Vektoren mit aufsteigender Länge aufgezählt. Dies liefert die Möglichkeit die Aufzählung abzubrechen, da man für die Länge der minimalen Lösungen aus $\mu S(A)$ eine obere Grenze angeben kann. Die Aufzählung an einem Knoten des Baumes kann also gestoppt werden, wenn der Vektor an diesem Knoten die Grenze überschreitet.

In [19] wird die folgende Schranke für die Länge der minimalen Lösungen vorgestellt, wobei r der Rang der Matrix ist.

$$\text{Für alle } \vec{s} \in \mu S(A) \text{ gilt } l(\vec{s} \leq (1 + max_i\{\sum_j |a_{ij}|\})^r = B$$

Da die Zeilen von A in der AC-Unifikation eigentlich immer linear unabhängig sind, kann man auf die aufwendige Berechnung von r verzichten und stattdessen m, die Anzahl der Zeilen verwenden.

Der Algorithmus ist in Abbildung 4.6 dargestellt und berechnet $\mu S(A)$.

Beispiel 4.5 Beim Aufzählen mit den Regeln

$$1 \rightarrow^{-1} Y_1 Y_2 {Y_4}^3, \quad Y_4 \rightarrow^{-1} Y_3 \quad \text{und} \quad Y_4 \rightarrow^{-1} Y_2$$

werden 93 Lösungen betrachtet. Die minimalen Lösungen sind

$$\begin{aligned} S_0(A) = \{ &(1\ 1\ 3\ 0\ 0), (1\ 1\ 0\ 3\ 0), (1\ 1\ 0\ 0\ 3), \\ &(1\ 1\ 2\ 1\ 0), (1\ 1\ 2\ 0\ 1), (1\ 1\ 1\ 2\ 0), \\ &(1\ 1\ 0\ 2\ 1), (1\ 1\ 1\ 0\ 2), (1\ 1\ 0\ 1\ 2), \\ &(1\ 1\ 1\ 1\ 1)\}. \ \blacksquare \end{aligned}$$

4.4 Vergleich der Verfahren

Beim Vergleich der drei Verfahren muß man zwischen einem allgemeinen Anwendungsfall und der speziellen Anwendung für die AC-Unifikation unterscheiden. Im Gegensatz zur allgemeinen Anwendung treten bei der Anwendung für die AC-Unifikation meist nur kleine Zahlen auf und entsprechend sind auch die Lösungen in diesem Fall kleiner. Dies ist von Bedeutung, da jedes der drei Verfahren einen Teil enthält, in dem minimale Lösungen durch Aufzählen von Vektoren gefunden werden. Die Größe des Bereichs, in dem Vektoren aufgezählt werden, d. h. die Anzahl der zu betrachtenden Vektoren, hängt von der Größe der Lösungen bzw. von der Größe der Zahlen in der Matrix A ab. Je größer die Lösungen sind, desto mehr Zeit nimmt daher der Aufzählungsanteil in den Verfahren ein. Während das Verfahren von Contejean und Devie nur aus einem Aufzählungsteil besteht, kann man die Verfahren von Domenjoud und von Pottier in drei Teile unterteilen. Im ersten Teil wird eine Basis für die Lösungen berechnet (Schritt 1 bis 2 im Verfahren von Domenjoud und Schritt 1 bis 3 im Verfahren von Pottier). Dann werden im zweiten Teil durch

1. $P_0 := TZ_1 \ldots Z_m - 1$;
   Für $j := 1$ bis m
   $P_j := Z^{A^{j^+}} - y_j Z^{A^{j^-}}$.
2. Bilde durch Vervollständigung von $\{P_0, \ldots, P_m\}$ mit der Reduktionsordnung $>_m$
   die Gröbner-Basis $B_{\mathcal{I}}$
3. $B_{\mathcal{J}} := \{P \in B_{\mathcal{I}} \mid \text{in } P \text{ kommt kein } Z_i \text{ oder } T \text{ vor}\}$;
   R ist das Regelsystem zu $B_{\mathcal{J}}$;
   $\longrightarrow^{-1}$ ist die inverse Relation zur Reduktionsrelation $\longrightarrow$ zu R.
4. $S := \emptyset$; $S_0 := \{0^n\}$; $i := 0$. (Beachte: $X^{0^n} = 1$)
5. Solange $S_i \neq \emptyset$
   $S := S \cup S_i$; $S_{i+1} := \emptyset$;
   Für alle $\vec{s} \in S_i$
   Für alle $\vec{t}$ mit $\vec{s} \longrightarrow^{-1} \vec{t}$
   Wenn $\vec{t} \notin S \cup S_{i+1}$ und $l(\vec{t}) < B$ dann
   $S_{i+1} := S_{i+1} \cup \{\vec{t}\}$;
   $i := i + 1$.
6. $V := \emptyset$;
   Für alle $\vec{s} \in V$
   Wenn kein $\vec{t}$ in S existiert mit $\vec{t} <^n \vec{s}$ dann
   $V := V \cup \{\vec{s}\}$.
6. Stopp. V ist $\mu S(A)$, die Menge der minimalen Lösungen.

Abbildung 4.6: *Algorithmus von L. Pottier zur Berechnung der minimalen Lösungen* $\mu S(A)$

Aufzählen die restlichen Lösungen bestimmt (Schritt 3 bzw. Schritt 4 und 5). Zuletzt werden dann die nicht minimalen Lösungen aussortiert (Schritt 4 bzw. Schritt 6). Das Verfahren von Contejean und Devie benötigt diesen dritten Teil nicht, da nur minimale Lösungen gefunden werden; es verwendet den ersten Teil nicht, da einfach alle Vektoren und nicht nur die Lösungen aufgzählt werden.

Das Verfahren von Contejean und Devie ist im allgemeinen Anwendungsfall schlechter als das Verfahren von Domenjoud. Weil bei ihm alle Vektoren, die kleiner als die minimalen Lösungen sind, aufgezählt werden, ist es bei Problemen mit großen Lösungen deutlich langsamer (siehe die letzten vier Beispiele in Tabelle A.1 im Anhang). Im speziellen Anwendungsfall der AC-Unifikation sind die Zahlen in A und die Lösungen aber meistens klein. In diesem Fall ist das Verfahren von Contejean und Devie manchmal schneller als das von Domenjoud. Weil sich die Fremdtermbedingungen besser als beim Verfahren von Domenjoud ausnutzen lassen, sind beide Verfahren für die AC-Unifikation mit Benutzung der Fremdtermbedingungen im Durchschnitt ungefähr gleich schnell (siehe Tabelle A.3). Wegen seiner Einfachheit ist das Verfahren von Contejean und Devie leicht zu implementieren, was einen zusätzlichen Vorteil schafft.

Die Bearbeitungszeit beim Verfahren von Domenjoud ist wesentlich weniger von der Größe

der Zahlen in A oder in den Lösungen abhängig. Diese spielt nur in Schritt 3 durch die Größe von D, der Diagonalen von T, eine Rolle. Die Zahlen auf der Diagonalen entstehen durch die Matrizentransformation, und ihre Größe hängt von der Größe der Lösungen aus $S_0(A)$ ab. Je größer D ist, desto mehr Vektoren $\vec{x}$ müssen für die Berechnung der restlichen Lösungen betrachtet werden. Viel entscheidender als die Größe der Zahlen ist für die Dauer des Verfahrens jedoch die Anzahl der zu betrachtenden Teilmengen der Spalten von A im ersten Teil und die Anzahl der linear unabhängigen Teilmengen von $S_0(A)$ im zweiten Teil. Da sich dies auch schon bei Problemen mit kleinen Zahlen auswirken kann, ist das ursprüngliche Verfahren von Domenjoud für die AC-Unifikation manchmal sehr viel schlechter als das Verfahren von Contejean (sihe Tabelle A.1). Durch die einfache Optimierung, die beim Auftreten von Vektoren der Höhe eins möglich ist, läßt sich das Verfahren so beschleunigen, daß es für das Lösen der diophantischen Gleichungen fast immer das schnellste ist (siehe Spalte 5 in Tabelle A.1). Da die Fremdtermbedingungen nicht so effizient ausgenutzt werden können, wird dieser Vorsprung für das Lösen der diophantischen Gleichungen mit den Bedingungen dann aber wieder vom Verfahren von Contejean und Devie ausgeglichen (siehe Tabelle A.3), so daß die beiden Verfahren hier gleich gut geeignet sind.

Ein Vorteil der Verfahren von Domenjoud und Pottier ist, daß schon nach dem ersten Bearbeitungsteil erkannt werden kann, ob es überhaupt natürlichzahlige Lösungen gibt (siehe die letzte Zeile in Tabelle A.1). Ist beim Verfahren von Domenjoud $S_0(A) = \emptyset$ oder gibt es beim Verfahren von Pottier keine Regel der Form $Y^{\vec{a}} \to 1$, kann die Aufzählung nicht gestartet werden und es existieren keine natürlichzahligen Lösungen. Da es bei der AC-Unifikation so gut wie gar nicht vorkommt, daß die diophantischen Gleichungen keine Lösung haben, spielt dieser Vorteil hier keine Rolle.

Die längsten Bearbeitungszeiten benötigt das Verfahren von Pottier (siehe Spalte 6 in Tabelle A.1). Dabei sind sowohl der erste Teil, die Erzeugung der Basis durch Vervollständigung, als auch der zweite Teil, die Aufzählung der Lösungen mit dem Regelsystem, den jeweiligen Teilen des Verfahrens von Domenjoud unterlegen. Da auch die Fremdtermbedingungen der AC-Unifikation nicht ausgenutzt werden können, ist es das ungeeignetste der drei hier vorgestellten Verfahren. Eine einfache Verbesserungsmöglichkeit besteht in der Verfeinerung der Schranke für die Länge der minimalen Lösungen. Einen Ansatz zur Vereinfachung und damit vielleicht die Möglichkeit zur Beschleunigung des Verfahrens oder zur Ausnutzung der Fremdtermbedingungen bietet die Veränderung der Repräsentation weg von den Polynomen. Diese können durch Vektoren von ganzen Zahlen ersetzt werden, da in der Implementierung ohnehin nur mit diesen gerechnet wird.

Kapitel 5

Wiederverwendung von AC-Lösungen

Eine häufige Vorgehensweise bei der Unifikation besteht darin, ein Unifikationsproblem in Teilprobleme aufzuteilen, um diese einzeln besser lösen zu können oder früher schon berechnete Lösungen für ein Teilproblem wiederverwenden zu können. Typisch für diese Vorgehensweise ist das Unifizieren zweier Substitutionen, das in der englischen Literatur Merging genannt wird. Beim Merging werden die schon ermittelten Unifikatoren zweier Unifikationsprobleme verwendet, um die Lösungen für das aus der Konjunktion dieser beiden Teilprobleme bestehende Unifikationsproblem zu berechnen.

In diesem Kapitel sollen einige Probleme aufgezeigt werden, die entstehen, wenn man ein Unifikationsproblem in zu kleine Teilprobleme aufteilt und diese einzeln löst. Der Kombinationsalgorithmus erlaubt diese Möglichkeit, weil die Regel *E-Res* nur auf einen Teil eines Unifikationsproblems angewendet werden muß. Dadurch ist es möglich, ein Unifikationsproblem in beliebig kleine Teilprobleme zu zerlegen, diese durch elementare E-Unifikation zu lösen und anschließend die gelösten Formen der Teilprobleme zu kombinieren und dabei zu unifizieren. Im ersten Abschnitt wird gezeigt, wann dies sinnvoll sein kann. Anschließend werden die Probleme gezeigt, die dabei bei Verwendung der AC-Theorie auftreten.

5.1 Motivation

In Kapitel 2 wird auf Seite 23 in Bedingung 2.3 Q_i, eine Menge von Termen definiert, die unifizierbar sein muß. Es sind dies Fremdterme, die über eine durch die elementare AC-Unifikation neu eingeführte Variable u_i gleichgesetzt werden. In Kapitel 3 wird dann deutlich, daß jeder Lösung der diophantischen Gleichungen $\vec{s}_i \in \mu S(A)$ genau ein solches Q_i zugeordnet ist. Die Forderung nach Unifizierbarkeit in Bedingung 3.2 liefert ein Abbruchkriterium für die Erzeugung von Lösungen aus $\mu S(A)$.

Aber auch wenn Q_i unifizierbar ist, kann die Effizienz gesteigert werden. Wenn $\vec{s}_i$ in mehreren passenden Teilmengen vorkommt, tritt das Problem der Unifikation von Q_i auch in

mehreren AC-gelösten Formen auf. Es erscheint daher sinnvoll, dieses zu Q_i gehörende Teilproblem einmal zu lösen und die so berechneten Lösungen in allen Unifikationsproblemen, in denen es auftaucht wiederzuverwenden; d. h. man wendet von vornherein in allen AC-gelösten Formen die Regel *E-Res* auf das zu Q_i gehörende Teilproblem an und ersetzt es ohne neue Berechnung durch seine schon vorhandene gelöste Form.

Beispiel 5.1 Für das Unifikationsproblem $P = f(z_1, b) + z_3 + z_4 \stackrel{?}{=} f(a, z_2) + a + b$ mit $\mathcal{F}_0 = \{f, a, b\}$, $\mathcal{F}_1 = \{+\}$ und $E_1 = AC^+$ ist die Matrix der diophantischen Gleichungen $A = [1\ 1\ 1\ -1\ -1\ -1]$, wobei den Spalten die Terme $f(z_1, b)$, z_3, z_4, $f(a, z_2)$, a, b in dieser Reihenfolge zugeordnet sind. Die AC^+-gelösten Formen sind

$$(\exists u_1) \quad u_1 \stackrel{?}{=} f(z_1, b) \ \wedge\ u_1 \stackrel{?}{=} f(a, z_2) \ \wedge\ z_3 \stackrel{?}{=} a \ \wedge\ z_4 \stackrel{?}{=} b \quad \text{und}$$
$$(\exists u_1) \quad u_1 \stackrel{?}{=} f(z_1, b) \ \wedge\ u_1 \stackrel{?}{=} f(a, z_2) \ \wedge\ z_3 \stackrel{?}{=} b \ \wedge\ z_4 \stackrel{?}{=} a.$$

Die Lösung $\vec{s}_1 = (1\ 0\ 0\ 1\ 0\ 0)$ wurde in beiden verwendet und so taucht das zur entsprechenden Termmenge $Q_1 = \{f(z_1, b), f(a, z_2)\}$ gehörende Teilproblem $P_{Q_1} = (\exists u_1) u_1 \stackrel{?}{=} f(z_1, b) \ \wedge\ u_1 \stackrel{?}{=} f(a, z_2)$ auch in beiden Formen auf. Anstatt P_{Q_1} zweimal zu lösen, ist es sinnvoll, die Unifikation nur einmal durchzuführen und P_{Q_1} durch die gefundene Lösung in beiden Formen zu ersetzen. Die gelöste Form zu P_{Q_1} ist $z_1 \stackrel{?}{=} a \ \wedge\ z_2 \stackrel{?}{=} b$ und in die beiden gelösten Formen oben eingesetzt erhält man $z_1 \stackrel{?}{=} a \ \wedge\ z_2 \stackrel{?}{=} b \ \wedge\ z_3 \stackrel{?}{=} a \ \wedge\ z_4 \stackrel{?}{=} b$ und $z_1 \stackrel{?}{=} a \ \wedge\ z_2 \stackrel{?}{=} b \ \wedge\ z_3 \stackrel{?}{=} b \ \wedge\ z_4 \stackrel{?}{=} a$. ■

Treten in einer AC-gelösten Form mehrere P_{Q_i} auf, so müssen nach der Ersetzung der P_{Q_i} durch ihre gelösten Formen diese unifiziert werden. Dies entspricht dem Merging von Substitutionen.

Beispiel 5.2 Für das Unifikationsproblem $P = f(z_1, b) + (z_1 * z_2) \stackrel{?}{=} f(h(z_3), z_2) + (h(a) * b)$ mit $\mathcal{F}_0 = \{f, h, a, b\}$, $\mathcal{F}_1 = \{+\}$, $\mathcal{F}_2 = \{*\}$ und $E_1 = AC^+$, $E_2 = AC^*$ ist die Matrix der diophantischen Gleichungen $A = [1\ 1\ -1\ -1]$, wobei den Spalten in dieser Reihenfolge die Terme $f(z_1, b)$, $z_1 * z_2$, $f(h(z_3), z_2)$ und $h(a) * b$ zugeordnet sind. Die AC^+-gelöste Form zu P ist

$$P_1 = (\exists u_1, u_2) u_1 \stackrel{?}{=} f(z_1, b) \ \wedge\ u_1 \stackrel{?}{=} f(h(z_3), z_2) \ \wedge\ u_2 \stackrel{?}{=} z_1 * z_2 \ \wedge\ u_2 \stackrel{?}{=} h(a) * b$$

Diese ist aus den Lösungen $\vec{s}_1 = (1\ 0\ 1\ 0)$ und $\vec{s}_2 = (0\ 1\ 0\ 1)$ aufgebaut, denen die Teilprobleme $P_{Q_1} = (\exists u_1) u_1 \stackrel{?}{=} f(z_1, b) \wedge u_1 \stackrel{?}{=} f(h(z_3), z_2)$ und $P_{Q_2} = (\exists u_2) u_2 \stackrel{?}{=} z_1 * z_2 \wedge u_2 \stackrel{?}{=} h(a) * b$ zugeordnet sind. Die gelöste Form zu P_{Q_1} ist $z_1 \stackrel{?}{=} h(z_3) \ \wedge\ z_2 \stackrel{?}{=} b$. Da P_{Q_2} AC-Terme enthält, hat es eine mehrelementige Menge von gelösten Formen $\{z_1 \stackrel{?}{=} h(a) \wedge z_2 \stackrel{?}{=} b; z_1 \stackrel{?}{=} b \ \wedge\ z_2 \stackrel{?}{=} h(a)\}$. Es ergeben sich durch das Ersetzen von P_{Q_1} und P_{Q_2} durch ihre gelösten Formen in P_1 zwei Unifikationsprobleme:

$$P_2 \ = \ z_1 \stackrel{?}{=} h(z_3) \ \wedge\ z_2 \stackrel{?}{=} b \ \wedge\ z_1 \stackrel{?}{=} h(a) \ \wedge\ z_2 \stackrel{?}{=} b$$
$$P_3 \ = \ z_1 \stackrel{?}{=} h(z_3) \ \wedge\ z_2 \stackrel{?}{=} b \ \wedge\ z_1 \stackrel{?}{=} b \ \wedge\ z_2 \stackrel{?}{=} h(a)$$

Während P_3 keine E-Lösung hat, hat P_2 die gelöste Form $z_1 \stackrel{?}{=} h(a) \ \wedge\ z_2 \stackrel{?}{=} b \ \wedge\ z_3 \stackrel{?}{=} a$. Dieser Unifikationsschritt entspricht dem Merging der Substitutionen $\{z_1 \mapsto h(z_3), z_2 \mapsto b\}$ und $\{z_1 \mapsto h(a), z_2 \mapsto b\}$ mit dem Ergebnis $\{z_1 \mapsto h(a), z_2 \mapsto b, z_3 \mapsto a\}$. ■

Eine weitere wichtige Anwendungsmöglichkeit des Merging besteht bei der Resolutionsmethode. Hat man zu jedem einzelnen Literal einer Klausel alle Resolutionsmöglichkeiten mit den dazu nötigen Substitutionen berechnet, kann man durch Merging dieser Substitutionen einen Unifikator berechnen, mit dem man auf allen Literalen die Resolution durchführen kann.

Ein weiterer Fall, in dem schon berechnete E-Lösungen wiederverwendet werden, kommt in vielen Unifikationsalgorithmen vor. Dabei werden die schon berecheneten Unifikatoren des ersten Unifikationsproblems auf das zweite angewendet. Die so entstandenen Instanzen werden gelöst und die dabei aufgestellten Unifikatoren mit den Unifikatoren des ersten Problems konkateniert. Dies geschieht z. B. bei der Unifikation von freien Termen.

Die Motivation für das Merging ist also die schon berechneten Lösungen für Unifikationsprobleme wieder benutzen zu können oder Probleme, die in mehreren Unifikationsproblemen als Teilproblem auftreten, nicht mehrfach berechnen zu müssen. Die naheliegendste Möglichkeit, dies zu tun, ist, die Substitutionen bzw. gelösten Formen zu unifizieren. Daß dies ein sehr ineffizientes Verfahren sein kann, zeigt das folgende Beispiel.

Beispiel 5.3 Die Unifikationsprobleme $P_1 = x_1 + x_2 \stackrel{?}{=} y_1 + y_1$ und $P_2 = y_1 + y_2 \stackrel{?}{=} x_1 + x_1$ haben minimale Mengen von E-Lösungen mit jeweils fünf Substitutionen. Um diese zu mergen müssen $5 \cdot 5 = 25$ Unifikationsprobleme gelöst werden. Da diese AC-Terme enthalten, haben sie minimale Lösungsmengen mit mehreren Elementen. Insgesamt erhält man so 186 Substitutionen als E-Lösungen für $P_1 \wedge P_2$. Diese Menge ist nicht minimal. Unifiziert man dagegen $P_1 \wedge P_2$ direkt mit dem Algorithmus zur elementaren AC-Unifikation erhält man eine minimale Menge von fünf E-Lösungen. Obwohl man so die schon berechnete Information nicht benutzt hat, ist dieser Weg wesentlich schneller. Außer dem ist das so berechnete Ergebnis besser, weil die Menge minimal ist. Auch der Weg, die Lösungssubstitution des einen Unifikationsproblems auf das andere anzuwenden und die so entstandenen Instanzen des Unifikationsproblems zu lösen, führt zu einem schlechteren Ergebnis. Da es für P_1 und P_2 jeweils fünf Lösungen gibt, müssen fünf Unifikationsprobleme gelöst werden, die in beiden Fällen zu 59 Unifikatoren führen. ■

Dieses Beispiel zeigt, daß das Benutzen von schon berechneten Lösungssubstitutionen für die AC-Theorie wesentlich schlechter sein kann, als die Unifikation ganz von vorn mit den ursprünglichen Termen zu beginnen. Gesucht wird nun ein Weg, mit dem man einen Teil der schon berechneten Informationen wieder benutzen kann, aber trotzdem eine minimale Menge von Lösungen erhält. Es soll also versucht werden, das Merging auf den Zwischenergebnissen der AC-Unifikation durchzuführen.

5.2 Die Zwischenschritte der AC-Unifikation

Als erstes werden die bei der AC-Unifikation auftretenden Teilschritte und die dabei entstehenden Zwischenergebnisse aufgezählt. Die Zwischenergebnisse sind

1. das Unifikationsproblem P.
2. die Matrix der diophantischen Gleichungen $A\vec{x} = 0$.
3. eine Repräsentation des Lösungsraumes $S_0(A)$ oder $\mathcal{R}$.
4. die Lösungsmenge $\mu S(A)$.
5. die passenden Teilmengen $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$.
6. die minimale, vollständige Menge von sequentiell gelösten Formen $\{P^{s_1}, \ldots, P^{s_n}\}$ bzw. E-Lösungen $\{\sigma_1, \ldots, \sigma_n\}$.

Der erste der dabei auftretenden Teilschritte ist das Aufstellen der diophantischen Gleichungen $A\vec{x} = 0$ aus dem Unifikationsproblem P, das die Eingabe der Unifikation ist. Im nächsten Schritt wird bei den Verfahren von Domenjoud und Pottier eine Darstellung für den Untervektorraum aller rationalen bzw. ganzzahligen Lösungen berechnet. Bei Domenjoud ist dies die Menge der minimalen Lösungen mit minimaler Trägermenge $S_0(A)$ und bei Pottier das Regelsystem $\mathcal{R}$. Im dritten Schritt wird durch Aufzählen die Menge der minimalen, natürlichzahligen Lösungen bestimmt. Dazu werden Grenzen benötigt, die das Aufzählen terminieren. Bei Domenjoud und Pottier werden nur Lösungen aufgezählt, die auf ihre Minimalität überprüft werden müssen. Beim Verfahren von Contejean und Devie werden alle Vektoren in aufsteigender Reihenfolge aufgezählt, so daß nur noch überprüft werden muß, ob es sich um Lösungen handelt. Im vierten Schritt werden die passenden Teilmengen der Lösungsmenge bestimmt und im letzten Schritt werden schließlich die Unifikatoren zusammengesetzt. Hier noch einmal die Zwischenergebnisse der einzelnen Schritte:

5.3 Weiterrechnen auf den Zwischenschritten

Da in jedem Zwischenergebnis der Teilschritte die gesamte Information enthalten ist, reicht es für das Mergen zweier Unifikationsprobleme P_1 und P_2 aus, für jedes Unifikationsproblem eines der Zwischenergebnisse auszuwählen. Aus diesen beiden Zwischenergebnissen lassen sich dann die E-Lösungen für die Konjunktion der beiden Unifikationsprobleme berechnen. Die einfachste Möglichkeit ist, die Konjunktion der beiden Unifikationsproblem, $P_1 \wedge P_2$, zu lösen. Benutzt man die beiden Mengen von seqentiell gelösten Formen $\{P_1^{s_1}, \ldots, P_1^{s_n}\}$ und $\{P_2^{s_1}, \ldots, P_2^{s_m}\}$, so muß man sämtliche Kombinationen, d. h. die $m \cdot n$ Unifikationsprobleme

$$P_1^{s_1} \wedge P_2^{s_1}, \ldots P_1^{s_1} \wedge P_2^{s_m}, \ldots P_1^{s_n} \wedge P_2^{s_1}, \ldots P_1^{s_n} \wedge P_2^{s_m}$$

lösen. Daß dies sehr ineffizient sein kann, wurde bereits im ersten Abschnitt des Kapitels gezeigt. Auch das Anwenden der Lösungssubstitutionen $\{\sigma_1, \ldots, \sigma_n\}$ des einen Unifikationsproblems auf das andere, d. h. das Lösen von $\sigma_1(P_2), \ldots, \sigma_n(P_2)$ mit anschließender Konkatenation der neu berechneten E-Lösungen mit dem jeweilgen σ_i, kann sehr ineffizient sein. Alle anderen Kombinationen von Zwischenergebnissen liegen auf einer anderen

Ebene, weil hier als Objekte nicht mehr nur Terme und Substitutionen auftreten sondern auch Zahlen, Vektoren und Matrizen. Es besteht natürlich bei jeder Kombination die Möglichkeit, die gegebenen Zwischenergebnisse getrennt weiter zu rechnen und die so erhaltenen gelösten Formen wie bei der direkten Verwendung der Endergebnisse zu mergen. Ein effizientes Verfahren zum Mergen müßte möglichst viel der schon berechneten Information verwenden, ohne den oben beschriebenen Nachteil zu haben, der bei der Verwendung der gelösten Formen auftritt.

Eine Verwendung der beiden Matrizen A_1 und A_2 ist natürlich sehr einfach, da man sie nur übereinanderschreiben und $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ lösen muß, aber dies bringt kaum einen Effizienzgewinn, da das Aufstellen der Matrizen keine aufwendige Operation ist. Der aufwendigste Schritt ist das Aufzählen beim Berechnen der minimalen Lösungen der diophantischen Gleichungen. Dieser Schritt sollte daher möglichst nicht wiederholt werden. Dies ist aber nicht möglich, da die Vektorraumbereiche, in denen bei den ursprünglichen Unifikationsproblemen und bei der Berechnung des Merge aufgezählt wird, im allgemeinen verschieden sind. Zwar ist der Lösungsvektorraum der diophantischen Gleichungen des Merge ein Untervektorraum des Lösungsraumes der ursprünglichen Unifikationsprobleme, aber da nur natürlichzahlige Lösungen gesucht werden, sind die Bereiche der minimalen Lösungen im allgemeinen nicht gleich, wie das folgende Beispiel zeigt.

Beispiel 5.4 Die diophantischen Gleichungen

$$\begin{bmatrix} 2 & 2 & -3 \end{bmatrix} \vec{x} = 0$$
$$\begin{bmatrix} 3 & -2 & 1 \end{bmatrix} \vec{x} = 0$$

haben die Lösungsmengen $\{(3,0,2),(0,3,2),(2,1,2),(1,2,2)\}$ und $\{(2,3,0),(0,1,2),(1,2,1)\}$. Die Lösungsmenge für den Merge dieser beiden Gleichungen

$$\begin{bmatrix} 2 & 2 & -3 \\ 3 & -2 & 1 \end{bmatrix} \vec{x} = 0$$

ist $\{(4,11,10)\}$. Diese Lösung ist aber bei keinem der Verfahren vorher schon aufgetreten. Beim Verfahren von Contejean und Devie wird bei der Lösung der beiden Gleichungen kein Vektor aufgezählt, der größer als eine der minimalen Lösungen ist. Beim Verfahren von Domenjoud werden bei der Aufzählung nur die noch fehlenden minimalen und keine überflüssigen Lösungen aufgezählt und beim Verfahren von Pottier beträgt die Grenze für die Länge der Lösungsvektoren in beiden Fällen 10. ■

Da alle Verfahren zum Lösen diophantischer Gleichungen mit einem Aufzählungsteil arbeiten und die Bereiche, in denen die Lösungen von A_1 und A_2 und von $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ liegen, im allgemeinen verschieden sind, ist es nur schwer vorstellbar, ein Verfahren zu finden, das aus $\mu S(A_1)$ und $\mu S(A_2)$ ohne Aufzählung $\mu S(A)$ berechnet. Das Aufzählen muß daher beim Merging erneut durchgeführt werden.

Da jeder Lösungsvektor aus $\mu S(A)$ eine Linearkombination der Lösungsvektoren der ursprünglichen diophantischen Gleichungen ist

$$\forall \vec{s} \in \mu S(A) : \vec{s} = \sum_{\vec{t} \in \mu S(A_1)} a_{\vec{t}} \vec{t}, a_{\vec{t}} \in \mathbb{N},$$

kann man den Test, ob eine Teilmenge $\mathcal{P} \subseteq \mu S(A)$ passend ist, auf die schon durchgeführten Tests für die Teilmengen von $\mu S(A_1)$ zurückführen. Sei $\mathcal{P}_{\vec{s}} = \{\vec{t} \in \mu S(A_1) | a_{\vec{t}} > 0\}$. $\mathcal{P}$ ist genau dann groß bzw. klein genug, wenn $\bigcup_{\vec{s} \in \mathcal{P}} \mathcal{P}_{\vec{s}}$ groß bzw. klein genug ist. Man braucht also die einzelnen Tests nicht mehr auf allen Komponenten der Lösungsvektoren durchführen, sondern muß nur noch die Vereinigungsmenge bilden und überprüfen, wie das Testergebnis für diese Menge bei einem der ursprünglichen Unifikationsprobleme war. Diese Methode bringt allerdings kaum einen Zeitgewinn, da die Berechnung der a_t bzw. $\mathcal{P}_{\vec{s}}$ sehr zeitaufwendig ist. Aber selbst wenn die a_t bzw. $\mathcal{P}_{\vec{s}}$ durch die vorherigen Schritte schon vorlägen, würde diese Methode kaum einen Zeitgewinn bringen, da die normalen Tests durch Kodierung der Teilmengen in Bitvektoren sehr schnell sind und nur einen vernachlässigbaren Anteil an der Gesamtzeit der AC-Unifikation haben.

Kapitel 6

Zusammenfassung

In dieser Arbeit wurde gezeigt, wie ein effizienter Algorithmus zur allgemeinen AC-Unifikation entwickelt werden kann. Die wichtigsten Bestandteile eines solchen Algorithmus sind ein Kombinationsverfahren, ein Verfahren zur elementaren AC-Unifikation und ein Verfahren zum Lösen diophantischer Gleichungen.

Als erstes wurde daher in Kapitel 2 das Verfahren von A. Boudet zur Unifikation mehrerer regulärer, kollapsfreier Theorien beschrieben und zur Kombination von AC-Theorien mit der freien Theorie verwendet. Wegen der speziellen Form des Ergebnisses der elementaren AC-Unifikation konnten in diesem Verfahren mehrere Schritte zu einem neuen zusammengefaßt werden, so daß ein Bestandteil des Kombinationsverfahrens, die Abstraktionsvariablen, nicht mehr benötigt werden. Gleichzeitig ergaben sich neue Bedingungen, die die Lösungen der elementaren AC-Unifikation erfüllen müssen.

In Kapitel 3 wurde dann ein Verfahren zur elementaren AC-Unifikation vorgestellt und gezeigt, wie sich die Bedingungen aus Kapitel 2 darin realisieren lassen. Durch Einhalten der Bedingungen kann das Aufstellen eines Teils der Lösungen, die im Kombinationsverfahren nicht zu einer Lösung führen, frühzeitig verhindert werden. Zusätzlich wurde noch gezeigt, wie sich neue Bedingungen für die Lösungen der diophantischen Gleichungen, die im Verfahren zur elementaren ACUnifikation benötigt werden, ergeben.

Zur Lösung der diophantischen Gleichungen wurden in Kapitel 4 drei Verfahren vorgestellt. Dabei wurde jeweils versucht die Bedingungen aus Kapitel 3 in die Algorithmen einzubauen, was für die drei Verfahren unterschiedlich gut gelang. Dem entsprechend sind die Verfahren zur Verwendung in der AC-Unifikation unterschiedlich gut geeignet.

Schließlich wurde in Kapitel 5 noch auf die Probleme eingegangen, die beim Wiederverwenden schon berechneter Lösungen der AC-Unifikation entstehen. Dabei wurde klar, daß die gebräuchlichste Art dies zu tun, das Mergen der Substitutionen, zu sehr schlechten Ergebnissen führen kann und es häufig besser ist, die AC-Unifikation ganz von vorn zu beginnen und auf die schon berechneten Ergebnisse zu verzichten.

Anhang A

Laufzeitvergleiche

Die in dieser Arbeit vorgestellten Verfahren, d. h. der auf dem Kombinationsverfahren aus [2] beruhende Algorithmus zur allgemeinen AC-Unifikation nach [3], das Verfahren zur elementaren AC-Unifikation und die drei vorgestellten Verfahren zum Lösen diophantischer Gleichungen von Contejean und Devie [6], Domenjoud [7] und Pottier [19], wurden mit den in dieser Arbeit vorgestellten Bedingungen und Optimierungen auf einer Symbolics 3640 in Lisp implementiert. Dabei wurden bei der Implementierung der AC-Unifikation die in [3] vorgestellten Optimierungen berücksichtigt, die in dieser Arbeit nicht noch einmal beschrieben wurden. Es folgen einige Tabellen mit Beispielen, an denen die Wirkung der in Kapitel 2 und 3 aufgestellten Bedingungen, die Zeitverteilung auf die einzelnen Schritte der AC-Unifikation und die Unterschiede zwischen den drei Verfahren zum Lösen diophantischer Gleichungen gezeigt werden sollen.

Zur Berechnung der Laufzeiten wurde jedes Problem zehnmal gelöst und der Durchschnitt der zehn Laufzeiten gebildet. Die Angaben sind in Millisekunden.

In Tabelle A.1 stehen die Laufzeiten der drei vorgestellten Verfahren zur Berechnung der minimalen Lösungsmenge von diophantischen Gleichungen. Dabei wurden keine Fremdterme übergeben und also auch keine der Bedingungen 3.1 oder 3.2 benutzt. In Spalte 5 ist ein Wert eingetragen, wenn es in $S_0(A)$ einen Vektor der Höhe 1 gibt, d. h. wenn die für diesen Fall in Kapitel 4 beschriebene Optimierung möglich ist. Ein Punkt bedeutet, daß diese Berechnung sehr lange dauert (mehr als 2 min.). Die ersten Beispiele sind typische Probleme, die bei der AC-Unifikation auftreten. Mit steigender Größe der Zahlen in A wird ein Auftreten während der AC-Unifikation immer seltener und die letzten Beispiele kommen bei der AC-Unifikation überhaupt nicht vor.

In Tabelle A.3 stehen die Laufzeiten für die Unifikation einiger AC-Terme und die Zeiten, die von den Verfahren zum Lösen der jeweiligen diophantischen Gleichungen mit Fremdtermbedingungen benötigt wurden. Das Verfahren von Pottier ist dabei nicht berücksichtigt, weil bei ihm die Fremdtermbedingungen nicht genutzt werden können und es einem Vergleich mit den beiden anderen Verfahren nicht standhält. Beim Verfahren von Domenjoud wurde die in Kapitel 4 beschriebene Optimierung benutzt, die möglich ist, wenn ein Vektor in $S_0(A)$ die Höhe 1 hat. In Spalte 2 steht die Anzahl der Lösungen in $\mu S(A)$, die

die Bedingungen 3.1 und 3.2 erfüllen. In Spalte 3 steht die Anzahl der vom Algorithmus berechneten AC-Lösungen. Die Gesamtzeit der AC-Unifikation wurde mit dem Verfahren von Domenjoud gemessen, da so wegen der bei diesem Verfahren meist konstanten Bearbeitungszeit innerhalb eines Abschnitts (siehe unten) der Vergleich der Gesamtzeiten besser möglich ist.

In den Unifikationsproblemen sind $x, y, z, u, v, w \in \mathcal{X}$ Variablen und $a, b, c, d \in \mathcal{C}$ Konstanten. Wie üblich ist $+ \in \mathcal{F}_1$ ein AC-Symbol und $f \in \mathcal{F}_0$ ein freies Symbol. Zur Übersichtlichkeit wurden als Fremdterme meistens Konstanten benutzt, da die Ergebnisse bei Verwendung anderer Fremdterme im wesentlichen gleich sind. Die Matrix der diophantischen Gleichungen A ist pro Abschnitt nur einmal abgebildet, weil sie für alle Beispiele eines Abschnitts gleich ist.

Zu jedem Unifikationsproblem mit Fremdtermen steht in der ersten Zeile des jeweiligen Abschnitts das Unifikationsproblem, das durch Variablenabstraktion entstanden ist, und die Werte in der ersten Zeile beziehen sich auf die Lösung dieses Problems ohne Bedingungen. In den restlichen Zeilen des Abschnitts stehen dann Unifikationsprobleme mit Fremdtermen, die mit Beachtung der Bedingungen gelöst wurden. Um die Wirkung der Bedingungen beim Auftreten von Fremdtermen in einem Unifikationsproblem zu sehen, muß man die Werte dieser Zeile also mit denen aus der ersten Zeile dieses Abschnitts vergleichen.

Man sieht deutlich, wie die Anzahl der Lösungen und Unifikatoren und die Bearbeitungszeit geringer werden, wenn die Unifikationsprobleme ohne Variablenabstraktion und mit den Fremdtermbedingungen gelöst werden. Besonders im zweiten und vierten Abschnitt wird deutlich, daß durch das Benutzen der Bedingungen schon bei nur einem Fremdterm die Bearbeitungszeit erheblich gesenkt werden kann. Dabei wird auch deutlich, daß die Gesamtbearbeitungszeit hauptsächlich von der Anzahl der Unifikatoren abhängt und daß beim Auftreten vieler Unifikatoren das Berechnen der passenden Teilmengen und das anschließende Aufstellen der Substitutionen den weitaus größten Anteil an der Gesamtbearbeitungzeit hat, während die Zeit zum Lösen der diophantischen Gleichungen einen sehr kleinen Anteil hat.

In Spalte 4 und 5 kann man sehen, wie sich die Fremdtermbedingungen auf die Bearbeitungszeiten der Verfahren von Contejean und Devie und von Domenjoud auswirken. Während in der ersten Zeile jedes Abschnitts stets die Zeit angegeben ist, die zum Lösen der diophantischen Gleichungen ohne Fremdterme und damit ohne Benutzung der Bedingungen benötigt werden, stehen in den darauf folgenden Zeilen die Zeiten, die sich bei Benutzung der Bedingungen mit den jeweiligen Fremdtermen ergeben. Während man beim Verfahren von Contejean und Devie erkennen kann, wie sich die Bearbeitungszeiten dadurch verringern, bleiben sie beim Verfahren von Domenjoud meist gleich. Dies liegt daran, daß die Vektoren, die die bei diesem Verfahren benutzbare Fremdtermbedingung 3.2 nicht erfüllen, fast immer auch die Höhe 1 haben und daher auch schon ohne Fremdterme von der in diesem Fall möglichen Optimierung betroffen sind. Lediglich im dritten Abschnitt kann man erkennen, wie sich die Bearbeitungszeit beim Verfahren von Domenjoud durch Verwenden der Fremdtermbedingungen verringert.

Tabelle A.1: *Vergleich der drei Verfahren zum Lösen diophantischer Gleichungen*

A	(2)	(3) ms	(4) ms	(5) ms	(6) ms
$\begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}$	4	19	74	13	534
$\begin{bmatrix} 1 & 1 & 1 & -1 & -1 & -1 \end{bmatrix}$	9	62	6730	45	11469
$\begin{bmatrix} 2 & -1 & -1 \end{bmatrix}$	3	18	17		572
$\begin{bmatrix} 3 & -1 & -1 \end{bmatrix}$	4	33	21		571
$\begin{bmatrix} 4 & -1 & -1 \end{bmatrix}$	5	45	31		570
$\begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 1 & -1 & -2 \end{bmatrix}$	2	56	23	12	459
$\begin{bmatrix} 1 & 2 & -2 & -1 \\ 1 & 0 & -2 & 0 \end{bmatrix}$	2	66	21		602
$\begin{bmatrix} 1 & 2 & -1 & -1 \\ 2 & 1 & -1 & -1 \end{bmatrix}$	4	147	26		539
$\begin{bmatrix} 1 & 2 & -1 & -1 & -1 \\ 2 & 1 & -1 & -1 & -1 \end{bmatrix}$	10	374	99		1339
$\begin{bmatrix} 1 & 2 & -1 & 0 & -2 & -1 \\ 1 & -1 & -2 & 2 & 0 & 1 \\ 2 & 0 & 1 & -1 & -2 & 0 \end{bmatrix}$	13	14186	2102	1382	17391
$\begin{bmatrix} 1 & 2 & -3 & -2 & -4 \\ 2 & 1 & -3 & 2 & 5 \end{bmatrix}$	10	1809	261		85673
$\begin{bmatrix} 10 & -7 & -8 & 3 & -11 \\ 12 & -9 & -7 & 3 & 13 \end{bmatrix}$	240	.	30000		.
$\begin{bmatrix} -10 & 0 & 20 & -1 & -21 \\ 9 & 1 & -17 & 2 & 19 \end{bmatrix}$	0	.	15		.

Legende auf der nächsten Seite

Tabelle A.2: Legende zu Tabelle A.1

(2) Anzahl der Lösungen in $\mu S(A)$.

(3) Bearbeitungszeit beim Verfahren von Contejean und Devie.

(4) Bearbeitungszeit beim Verfahren von Domenjoud.

(5) Bearbeitungszeit beim optimierten Verfahren von Domenjoud.

(6) Bearbeitungszeit beim Verfahren von Pottier.

Tabelle A.3: *Beispiele zur AC-Unifikation*

P bzw. A	Anzahl der Lös. in $\mu S(A)$	Anzahl der AC–Lösungen	Contej. Zeit in ms	Domenj. optimiert Zeit in ms	Gesamt–zeit in ms
$x+y=u+v$ $\begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}$	4	7	19	15	98
$a+y \stackrel{?}{=} u+v$	4	4	21	14	67
$a+b \stackrel{?}{=} u+v$	4	2	21	16	45
$a+y \stackrel{?}{=} c+v$	3	2	16	15	53
$a+b \stackrel{?}{=} c+v$	2	0	12	15	35
$a+b \stackrel{?}{=} a+v$	3	1	16	16	33
$x+x+x \stackrel{?}{=} u+v+w$ $\begin{bmatrix} 3 & -1 & -1 & -1 \end{bmatrix}$	10	981	99	106	17219
$x+x+x \stackrel{?}{=} a+v+w$	7	44	79	109	685
$x+x+x \stackrel{?}{=} a+b+w$	3	2	40	101	138
$x+x+x \stackrel{?}{=} a+b+c$	0	0	18	99	105
$x+x+y \stackrel{?}{=} u+v+w$ $\begin{bmatrix} 2 & 1 & -1 & -1 & -1 \end{bmatrix}$	9	381	55	69	4791
$x+x+y \stackrel{?}{=} u+v+w$	6	8	38	47	145
$x+x+y \stackrel{?}{=} u+v+w$	4	2	26	35	373
$x+y \stackrel{?}{=} z+z \wedge$ $x+z \stackrel{?}{=} u+v$ $\begin{bmatrix} 1 & 1 & -2 & 0 & 0 \\ 1 & 0 & 1 & -1 & -1 \end{bmatrix}$	9	483	247	199	22427
$x+a \stackrel{?}{=} z+z \wedge$ $x+z \stackrel{?}{=} u+v$	7	44	122	199	962
$x+a \stackrel{?}{=} z+z \wedge$ $x+z \stackrel{?}{=} b+v$	6	20	164	196	482
$x+y \stackrel{?}{=} z+z \wedge$ $x+z \stackrel{?}{=} b+c$	2	0	116	193	218
$a+y \stackrel{?}{=} z+z \wedge$ $a+z \stackrel{?}{=} u+v$	5	8	115	195	330
$x+f(y) \stackrel{?}{=} z+z \wedge$ $x+z \stackrel{?}{=} f(u)+v$	4	4	114	194	275

Literaturverzeichnis

[1] F. Baader. Unification Theory. In K. U. Schulz, editor, *Word Equations and Related Topics, 1. Int. Workshop, IWWERT'90, LNCS 572*, pp. 151-170. Springer-Verlag, 1992.

[2] A. Boudet. *Unification dans le Mélange de théories Equationelles.* PhD thesis, Université de Paris-Sud, Orsay, February 1990.

[3] A. Boudet. Competing for the *AC*-Unification Race.

[4] B. Buchberger. Gröbner basis: an algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Recent Trends in Multidimensional System Theory.* 1985.

[5] M. Clausen, A. Fortenbacher. Efficient Solution of Linear Diophantine Equations. In C. Kirchner, editor, *Unification.* Academic Press, 1990.

[6] E. Contejean, H. Devie. Solving systems of linear diophantine equations. In *UNIF '89. Extended Abstracts of the 3rd Int. Workshop on Unification*, pp. 123-126. SEKI Report SR-89-17, Universität Kaiserslautern, 1989.

[7] E. Domenjoud. Solving Systems of Linear Diophantine Equations: An Algebraic Approach. In *Proc. of UNIF '90.* Leeds, 1990.

[8] F. Fages. Associative-Commutative Unification. *Journal of Symbolic Computation, Vol. 3, No. 3,* pp. 257-275, 1987.

[9] T. Guckenbiehl, A. Herold. *Solving Linear Diophantine Equations.* SEKI Report, SEKI-85-IV-KL, Universität Kaiserslautern, 1985.

[10] A. Herold, J. Siekmann. *Unification in Abelian Semigroups.* SEKI Report, SEKI-85-III-KL, Universität Kaiserslautern, 1985.

[11] A. Herold. *Combination of Unification Algorithms in Equational Theories.* Dissertation, Universität Kaiserslautern, 1987.

[12] J. M. Hullot. Associative commutative pattern matching. In *Proc. of the 6th IJCAI, Vol. 1*, pp. 406-412. Tokyo, 1979.

[13] A. Kandri-Rody, D. Kapur. Algorithms for computing Gröbner bases of polynomial ideals over various euclidean rings. In *Proc. EUROSAM 84, LNCS 174.* Springer-Verlag, 1984.

[14] C. Kirchner. *Méthodes et Outils de Conception Systematique d'Algorithmes d'Unification dans les Théories Equationnelles.* Thèse d'Etat, Univ. Nancy, France, 1985.

[15] D. E. Knuth, P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pp. 263-297. Pergamon Press, 1970.

[16] P. Lincoln, J. Christian. Adventures in Associative-Commutative Unification. In C. Kirchner, editor, *Unification*, pp. 393-416. Academic Press, 1990.

[17] W. Nutt. *The Unification Hierarchy is Undecidable.* SEKI Report SR-89-06, Universität Kaiserslautern, 1989.

[18] G. Plotkin. Building-in equational theories. In *Machine Intelligence 7*, pp. 73-90. Edinburgh Univ. Press, 1972.

[19] L. Pottier. Minimal solutions of linear diophantine systems: bounds and algorithms. In R. V. Book, editor, *RTA-91, Rewriting Techniques and Applications, Como, LNCS 488*, pp. 162-173. Springer-Verlag, 1991.

[20] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM, Vol. 12, No. 1*, pp. 23-41, 1965.

[21] M. Schmidt-Schauß. *Unification in a Combination of Arbitrary Disjoint Equational Theories.* SEKI-Report SR-87-16, Universität Kaiserslautern, 1987.

[22] J. H. Siekmann. *Unification and Matching Problems.* PhD. Thesis, University of Essex, 1978.

[23] M. Stickel. A Complete Unification Algorithm for Associative-Commutative Functions. In *Advanced Papers of the 4th IJCAI, Vol. 1*, pp. 71-82. Tblisi, 1975.

[24] M. Stickel. A Complete Unification Algorithm for Associative-Commutative Unification. *Journal of the ACM, Vol. 28, No. 3,* pp. 423-434, 1981.

[25] E. Tidén. *First-Order Unification in Combinations of Equational Theories.* Thesis, Stockholm, 1986.

[26] K. A. Yelick. Unification in Combinations of Collaps-free Regular Theories. *Journal of Symbolic Computation, Vol. 3,* pp. 153-181, 1987.

Index