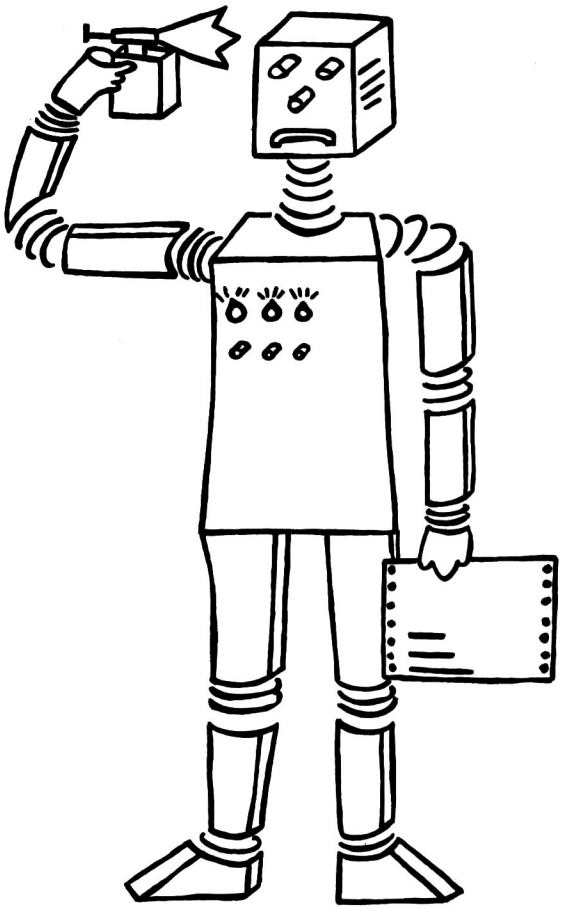
SEKI Working Paper

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany



Ansatzpunkte für heuristische Methoden bei der Vervollständigung

Jörg Denzinger

SEKI Working Paper SWP-90-01

Ansatzpunkte für heuristische Methoden bei der Vervollständigung

Jörg Denzinger

Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
D-6750 Kaiserslautern
F.R.G.

Diese Arbeit wurde vom Sonderforschungsbereich 314,
"Künstliche Intelligenz - Wissensbasierte Systeme", D4-Projekt, unterstützt

Abstract

We deal with the most important features of the Knuth-Bendix completion procedure considered as search problems. There are four indeterminisms in this algorithm : ordering, computation of normalforms, interreduction and selection of critical pairs. They are represented as search trees in order to show the programmer's choices when implementing such an algorithm. The problems are outlined by examples and some possible heuristic criteria for their solutions are given.

Wir betrachten den Knuth-Bendix-Vervollständigungsalgorithmus unter dem Aspekt von Suchproblemen. Die vier Indeterminismen : Ordnung, Normalformbildung, Interreduction und Auswahl des nächsten kritischen Paares werden als Suchbäume uniform dargestellt, um die Wahlmöglichkeiten eines Programmierers bei dem Implementation einer Vervollständigung aufzuzeigen. Die auftretenden Probleme werden an Hand von Beispielen geschildert und Ansatzpunkte für Heuristiken zur Lösung dieser Probleme gegeben.

1. Einführung

Im folgenden werden wir den von Knuth und Bendix in [KB70] entwickelten Vervollständigungsalgorithmus auf Suchprobleme hin untersuchen. Der A^*-Algorithmus ([Ni80]) bietet dann ein Lösungsverfahren für diese Suchprobleme. Insbesondere können damit (noch zu entwickelnde) Heuristiken zum schnellen Finden von Lösungen herangezogen werden. Ziel dieser Arbeit ist nicht die Entwicklung solcher Heuristiken sondern die Betrachtung des KB-Algorithmus als eine Menge von Suchproblemen, um Entwicklern von Vervollständigungssystemen die dabei bestehenden Freiheiten aufzuzeigen.
Ein Suchproblem wird beschrieben durch einen Suchbaum, der den aktuellen Stand der Suche beschreibt.
Wir werden zunächst eine informelle, motivierende Beschreibung des KB-Algorithmus geben (Kap. 2) und diese dann, nach einer Zusammenstellung der Grundlagen und Notationen (Kap. 3), soweit präzisieren, daß wir die jeweiligen Suchprobleme und ihre Stellung im Algorithmus angeben zu können. In Kapitel 4 werden dann die Suchprobleme, die bei der Knuth-Bendix-Vervollständigung auftreten, in einer einheitlichen Form, eben den Suchbäumen, dargestellt. Für jedes Suchproblem werden außerdem einige Kriterien angegeben, die von Heuristiken für das jeweilige Problem benutzt werden können.
Die Beschreibung der Suchprobleme wird sehr allgemein gehalten, um möglichst viele der bekannten Vorgehensweisen zu enthalten. Dadurch kann ein Knoten sehr viele Nachfolger haben. Für konkrete Anwendungen muß deshalb eine Vorauswahl in Form von Strategien getroffen werden, wie wir es in unseren Beispielen, der besseren Darstellbarkeit wegen, tun (vgl. 4.3., 4.4b.).

2. Der KB-Algorithmus

Ausgangspunkt des KB-Algorithmus ist eine Menge E von Gleichungen. Die Seiten einer Gleichung sind Terme, eventuell spezialisiert zu Worten oder Polynomen. Ziel des KB-Algorithmus ist die Konstruktion eines eindeutig terminierenden Regelsystems R aus E. Als erstes sind dazu Gleichungen s=t zu Regeln $l \to r$ zu richten. Dazu wird eine Ordnung

> auf den Termen benötigt, so daß l>r gilt. Durch zusätzliche Bedingungen (Wohlordnungen, Simplifikationsordnungen) an solche Termordnungen entstehen die Reduktionsordnungen, die die Termination von Regelsystemen garantieren. Allerdings ist der Ablauf des KB-Algorithmus zu einer festen Regelmenge sehr von der Wahl dieser Reduktionsordnung abhängig. Einige Ordnungen können zudem während des Laufes erweitert werden. Eine Regel kann auf Terme angewendet werden, indem Teilterme des Terms, auf die die linke Seite der Regel paßt (mit einem Match), durch die substituierte rechte Seite ersetzt werden.
Um eindeutig terminierende Regelsysteme zu beschreiben, benötigt man noch den Begriff der Konfluenz. Ein Regelsystem R heißt konfluent, falls sich zwei verschiedene Terme, die sich durch Anwendung von Regeln aus einem Term ableiten lassen, durch beliebig viele weitere Anwendungen von Regeln aus R zu einem gemeinsamen Term reduzieren lassen. Ein konfluentes, terminierendes Regelsystem heißt dann eindeutig terminierend. Die Konstruktion eines eindeutig terminierenden Regelsystems zu einer Menge von Gleichungen ist nicht immer möglich. Der KB-Algorithmus, der diese Konstruktion vorzunehmen versucht, terminiert dann entweder nicht oder bricht ab, weil die benutzte Reduktionsordnung eine neu entstandene Gleichung nicht richten kann (fail).
Der zentrale Punkt der Konstruktion eines eindeutig terminierenden Regelsystems durch den KB-Algorithmus ist die Hinzunahme neuer Gleichungen, um die Konfluenz des Regelsystems zu erzwingen. Diese Gleichungen entstehen durch kritische Paare zu schon vorhandenen Regeln. Welche kritischen Paare erzeugt und welche dieser erzeugten Paare zur Bildung neuer Gleichungen wann benutzt werden, beeinflußt stark die Effizienz des Algorithmus.
Schließlich wird bei den meisten Varianten des KB-Algorithmus versucht, das aktuelle Regelsystem minimal zu repräsentieren. Dazu werden neue Regeln benutzt, um ältere Regeln zu reduzieren. Diese Interreduktion hängt ebenso wie die Reduktion der kritischen Paare davon ab, welche Normalform eines Terms bezüglich des aktuellen Regelsystems auf welche Weise gebildet wird.
Sowohl Reduktion als auch kritische-Paar-Bildung können modulo einer Gleichungstheorie erfolgen. Damit ist bei Vergleichen zwischen Termen nicht mehr die syntaktische Gleichheit, sondern die Gleichheit bezüglich dieser Theorie zu benutzen. Eine oft benötigte Theorie ist die gleichzeitige Assoziativität und Kommutativität mancher Funktionssymbole (AC), wie zum Beispiel bei der Addition "+" und Multiplikation "*" von natürlichen Zahlen. Alle vorher beschriebenen Teile des KB-Algorithmus werden auch hier benötigt, so daß unsere weiteren Überlegungen auch auf die Vervollständigung modulo einer Theorie Anwendung finden können.

3. Grundlagen

Sei $\mathcal{F}$ eine Menge von Funktionssymbolen und $\mathcal{U}$ eine Menge von Variablen. Die Menge $\mathcal{T}$ der Terme ist rekursiv wie folgt definiert. Ist $x \in \mathcal{U}$, so ist $x \in \mathcal{T}$. Ist $f \in \mathcal{F}$ ein n-stelliges Funktionssymbol, und sind $t_1,...,t_n \in \mathcal{T}$, dann ist auch $f(t_1,...,t_n) \in \mathcal{T}$.
Eine Ordnung > auf Termen heißt Noethersch, falls es keine unendliche Kette $t_1,...,t_n,... \in \mathcal{T}$ gibt, mit $t_1>t_2>...>t_n>...$.
Eine Funktion $\sigma : \mathcal{U} \rightarrow \mathcal{T}$ heißt Substitution, falls es nur endlich viele $x_1,...,x_n \in \mathcal{U}$ gibt

mit $\sigma(x_i) \neq x_i$. Gemäß der rekursiven Struktur der Terme wird σ durch $\sigma(f(t_1,...,t_n)) = f(\sigma(t_1),...,\sigma(t_n))$ auf Terme fortgesetzt.
Ist R ein Regelsystem, $l_1 \Rightarrow r_1$, $l_2 \Rightarrow r_2 \in R$ und gibt es einen Term s in l_1 (Schreibweise: $l_1[n \Leftarrow s]$) an der Stelle n, s keine Variable, und eine Substitution σ, so daß $\sigma(s) = \sigma(l_2)$ gilt, dann heißt

$$\langle \sigma(l_1[n \Leftarrow \sigma(r_2)]) \ , \ \sigma(r_1) \rangle$$

ein kritisches Paar von $l_1 \Rightarrow r_1$ und $l_2 \Rightarrow r_2$. Allgemein beschreibt das Quadrupel $(l_1 \Rightarrow r_1, l_2 \Rightarrow r_2, n, \sigma)$ die kritische Stelle zu diesem kritischen Paar.
Die Anwendung einer Regel (Reduktion) wird formal so beschrieben. Sei $t \in \mathcal{T}$ und $l \Rightarrow r$ in R. Ist s ein Teilterm von t an der Stelle n und gibt es eine Substitution σ mit $\sigma(l) = s$, dann kann t zu $t[n \Leftarrow \sigma(r)]$ reduziert werden (Schreibweise: $t \Longrightarrow t[n \Leftarrow \sigma(r)]$).

4. Suchprobleme beim KB-Algorithmus

Aus der Beschreibung des KB-Algorithmus in Kapitel 2 lassen sich vier Indeterminismen herausfiltern, die bei der Implementierung von Vervollständigungsalgorithmen präzisiert und damit determiniert werden müssen.

- Wahl der Reduktionsordnung
- Normalformbestimmung
- Interreduktion eines Regelsystems
- Behandlung kritischer Stellen durch Erzeugen neuer Regeln

Leider können diese vier Punkte nicht unabhängig voneinander gesehen werden, da zum Beispiel die Notwendigkeit der Erweiterung einer Reduktionsordnung nur dadurch entsteht, daß neue Gleichungen, die aus kritischen Paaren entstehen, zu Regeln gerichtet werden müssen. Ebenso wird die Interreduktion natürlich nur dann benötigt, wenn neue Regeln erzeugt werden, usw.
Wir werden deshalb bei der genauen Untersuchung eines Punktes davon ausgehen, daß die anderen Punkte irgendwie realisiert sind, müssen aber zum Beispiel beim Punkt "kritische Stellen" berücksichtigen, daß ein Term mehrere Normalformen haben kann.
Zur Darstellung der vier Punkte als Suchprobleme benutzen wir Bäume. Dann sind folgende Elemente zu einem Problem zu spezifizieren:
Die Wurzel des Baumes beschreibt den Ausgangspunkt der Suche, die (inneren) Knoten des Baumes beschreiben Zwischenschritte bei der Lösung, die Kanten die Übergänge von einem Knoten in einen anderen, unter Berücksichtigung von Übergangsregeln, und die Blätter schließlich Endpunkte, die entweder Lösungen des Problems darstellen, oder das Wissen, daß dieser Zweig des Baumes zu keiner Lösung führen kann (fail-Blätter).

4.1. Die Wahl der Reduktionsordnung

In diesem Abschnitt betrachten wir die ganze KB-Vervollständigung unter dem Aspekt der Reduktionsordnung. So gesehen besteht die Hauptaufgabe darin, ein Termpaar mit der Ordnung zu richten, also den größeren Term zu bestimmen. Da Partialordnungen als Reduktionsordnungen benutzt werden, muß es bei zwei Termen keinen größten geben. Diese Aufgabe wird, abgesehen von der initialen Wahl der Ordnung, dadurch zum Suchproblem, daß manche Ordnungen, wie z.B. die RPO ([Der85]) oder LPO ([KL80])

erweiterbar sind. Dadurch ist bei zwei unvergleichbaren Termen eine Wahlmöglichkeit gegeben, was sich in folgender Suchbaumstruktur ausdrückt.

Wurzel :	Ein Gleichungssystem E und ein leeres Regelsystem R
innere Knoten :	Ein Gleichungssystem und ein Regelsystem
Blätter :	Ein leeres Gleichungssystem und ein eindeutig terminierendes Regelsystem oder **fail**.
Kanten :	Termordnungen
Übergang :	Die Nachfolger M (mit Ordnung $>_M$) eines Knotens N (mit Ordnung $>_N$) entstehen dadurch, daß entweder aus dem Regelsystem durch kritische-Paarbildung neue Gleichungen erzeugt werden und durch Anwendung der Termordnung $>_M$ der betreffenden Kante Gleichungen aus E gerichtet und in R aufgenommen werden, oder nur in N noch nicht richtbare Gleichungen aus E nun richtbar werden. Dabei gilt für die neue Ordnung $>_M$: $>_N \subseteq >_M$. Ist diese Bedingung nicht erfüllt, dann ist M **fail**.

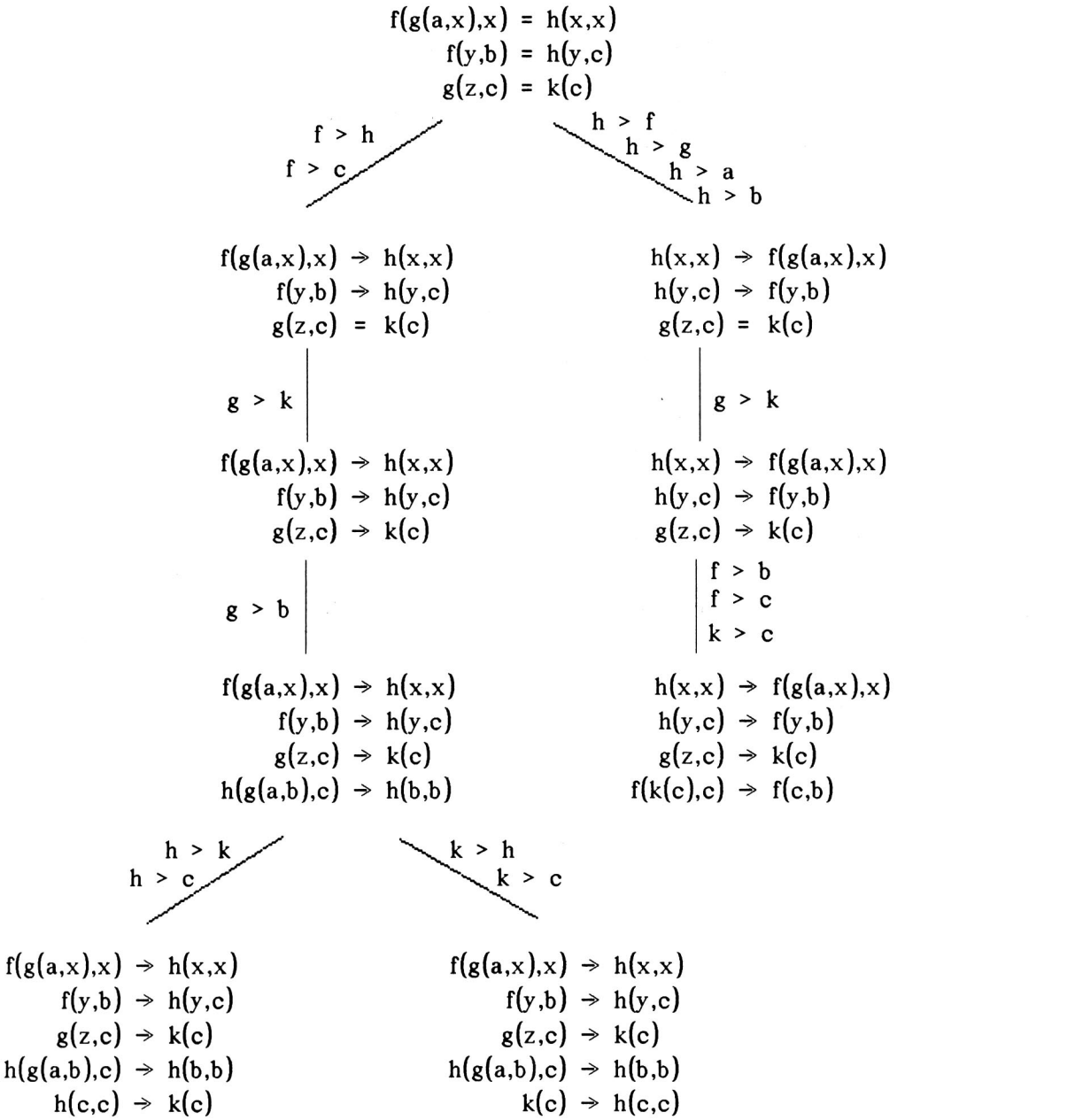
Es gibt viele Reduktionsordnungstypen, die für feste Gleichungsmengen unterschiedliche Abläufe des KB-Algorithmus bewirken. Manche dieser Ordnungstypen kann man, den Umständen gemäß, verfeinern, wie zum Beispiel die RPO, die mit einer Vorordnung arbeitet. Startet man mit leerer Vorordnung und erweitert diese Vorordnung nach Bedarf, so entsteht eine Kette von Ordnungen $>_0 \subseteq >_1 \subseteq \ldots \subseteq >_k$, wie bei den Knotenübergängen gefordert.
Andere Ordnungen haben diese Eigenschaften nicht, so daß nach Wahl einer solchen Ordnung in einem Knoten N entweder alle Blätter von N aus mit **fail** beschriftet sind, oder alle Blätter von N aus dasselbe Regelsystem enthalten oder **fail**-Blätter sind.
Im folgenden Beispiel können wir natürlich nicht alle bekannten Reduktionsordnungen zur Verzweigung im Baum benutzen. Wir beschränken uns auf einen Typ, die LPO, und variieren die Vorordnung.
Die LPO > zu einer Vorordnung $>_P$ ist wie folgt definiert (vgl. [KL80]). Es ist

$t = f(t_1,\ldots,t_n) > g(s_1,\ldots,s_m) = s$
falls a) $\exists i : t_i \geq s$
oder b) $f >_P g$ und $\forall j : t > s_j$
oder c) $f = g$ und $t_1 \ldots t_n \geq^* s_1 \ldots s_m$ und $\forall j : t > s_j$.

Zusätzlich gilt $t > x$ für alle t, in denen die Variable x vorkommt.
Wir starten die Vervollständigung mit leerem $>_P$ und dem Gleichungssystem

$f(g(a,x),x) = h(x,x)$
$f(y,b) = h(y,c)$
$g(z,c) = k(c)$

Zunächst sind nacheinander diese Gleichungen zu richten. Dies kann schon auf mehrere Arten geschehen. Ordnungen, die zu unvergleichbaren Seiten einer Gleichung führen und damit fail-Knoten erzeugen, werden wir nicht berücksichtigen.



Je nach Wahl der Ordnung können also verschieden lange Pfade mit völlig verschiedenen Endregelsystemen entstehen.
Eine Heuristik für dieses Problem könnte zur Bestimmung eines kurzen Pfades zu einem Nicht-**fail**-Blatt folgende Kriterien benutzen :

- Aussehen des aktuellen Regel- und Gleichungssystems, eventuell vorkommende bekannte Regeln wie A-, D- oder C-Eigenschaften von Funktionssymbolen.
- Ähnlichkeit des aktuellen Regel- und Gleichungssystems mit vom Benutzer vorgegebenen Regeln.
- Größe des Regel- und Gleichungssystems

etc.

4.2. Bestimmung von Normalformen eines Terms

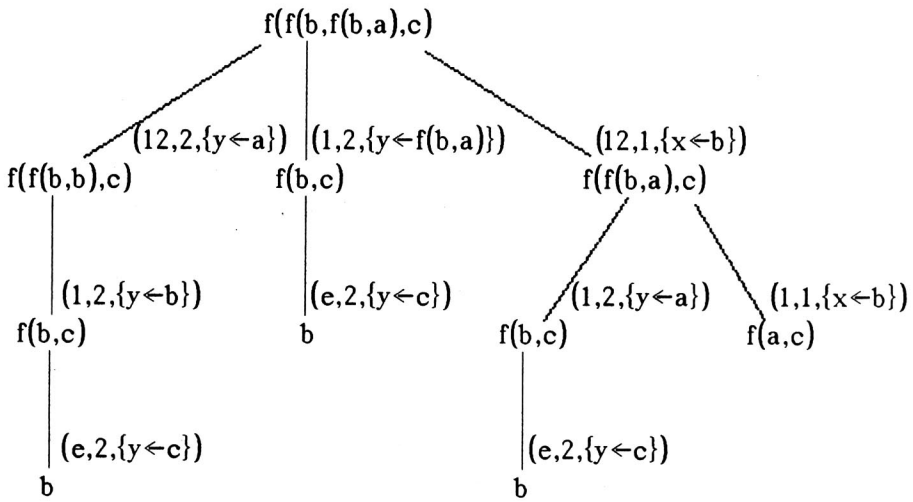
Die Bestimmung der Normalform eines Terms ist eine häufig benutzte Operation des KB-Algorithmus. So müssen nach der Bildung eines kritischen Paares die beiden das Paar bildenden Terme auf Normalform gebracht werden. In der Regel braucht man nur eine Normalform, aber da während der Vervollständigung das aktuelle Regelsystem meistens noch nicht eindeutig terminierend ist, kann es mehrere Normalformen zu einem Term geben.
Als Suchproblem kann die Operation "Normalformbildung" zu einem festen Regelsystem R so beschrieben werden.

Wurzel :	ein Term
innere Knoten :	Terme
Blätter :	irreduzible Terme (bezüglich des aktuellen Regelsystems R)
Kanten :	Tripel $(n,l\to r,\sigma)$, mit n Position in einem Term, $l\to r$ Regel des aktuellen Regelsystems und σ Substitution.
Übergang :	Der Nachfolger M eines Knotens N mit Kante $(n,l\to r,\sigma)$ entsteht dadurch, daß an Position n des Terms t von N $\sigma(r)$ gesetzt wird, weil $\sigma(l) = t/n$ gilt.

Da eine Regel an verschiedenen Positionen eines Terms anwendbar sein kann und die Anwendbarkeit für alle Regeln getestet werden muß, können die entstehenden Bäume sehr groß werden. Insbesondere gibt es in den Bäumen sehr viel Redundanz, wie das folgende Beispiel zeigt.
Das aktuelle Regelsystem R sei

R = { 1: f(x,a) → a,
2: f(b,y) → b }

und der Term



Es gibt 2 Normalformen b und f(a,c), wobei b auf 3 Arten erzeugt werden kann. Für die Effizienz des KB-Algorithmus ist es wichtig, diese Operation mit wenig Schritten auszuführen. Andererseits können bei manchen Strategien zur Bildung von kritischen Paaren alle Normalformen benötigt werden (siehe [Sa86] und 4.4b.). Deswegen wird es ganz von den Anforderungen der jeweiligen Variante abhängen, welche Strategien und Heuristiken hier zur Anwendung kommen.
Mögliche Kriterien sind

- Ähnlichkeit zu einer gewünschten Normalform (z.B. die andere Seite des kritischen Paares und ihre Normalformen).
- Systematische Strategien zum Finden genau einer Normalform (inner- oder outermost Strategie, etc.)

Bei der Implementierung eines möglichst vielseitigen Vervollständigungssystems sollte der Möglichkeit, eine Menge von Normalformen zu erhalten, Rechnung getragen werden.

4.3. Interreduktion

Um die Effizienz von Vervollständigungsalgorithmen zu erhöhen, versucht man, das aktuelle Regelsystem R möglichst klein zu halten. Das heißt insbesondere, daß keine linke oder rechte Seite einer Regel in R mit anderen Regeln aus R reduzierbar sein sollte. Kommen neue Regeln in das Regelsystem, so muß R umgeformt werden, um dieser Bedingung zu genügen. Diese Interreduktion hängt natürlich von der Reihenfolge ab, in der versucht wird, eine Regel auf Reduzierbarkeit mit anderen Regeln zu testen. Deshalb ergibt sich folgendes Suchproblem.

Wurzel :	ein Regelsystem R
innere Knoten :	Regelsystem R
Blätter :	Regelsystem R, wobei für jede Regel $l \to r \in R$ gilt : l und r sind bezüglich $R \setminus \{l \to r\}$ irreduzibel.
Kanten :	2 Mengen von Regeln R_{out} und R_{in}.
Übergang :	Der Nachfolger M eines Knotens N mit Kante (R_{out}, R_{in}) entsteht dadurch, daß die Regeln aus R_{out} im Regelsystem R_M des Nachfolgers nicht mehr vorkommen und durch die Regeln aus R_{in} ersetzt werden. Bezeichnet R_N das Regelsystem des Knoten N, so gilt also $R_M = R_N \setminus R_{out} \cup R_{in}$. Weiter gilt für jedes $l \to r$ in R_{out}, daß entweder l oder r bezüglich $R_N \setminus \{l \to r\}$ reduzierbar ist.

Die Interreduktion benutzt die Bildung der Normalformen aus 4.2.. Betrachtet man ein beliebiges Noethersches Regelsystem R, so kann es Regeln $l_1 \to r_1$ und $l_2 \to r_2$ geben, sodaß l_1 oder r_1 mit $l_2 \to r_2$ reduzierbar sind, etwa zu l_3 und r_3. Sind l_4 und r_4 Normalformen von l_3 und r_3 bezüglich $R \setminus \{l_1 \to r_1\}$, so haben $R_1 = R \setminus \{l_1 \to r_1\} \cup \{l_4 \to r_4\}$ (falls $l_4 > r_4$) und R die selben Äquivalenzklassen, aber R enthält redundante Information. Deshalb kann man R_1 als "kleiner" als R bezeichnen. Dies wird besonders deutlich, wenn $l_4 \equiv r_4$ gilt.

Dann hat R_1 eine Regel weniger als R und damit gibt es auch weniger kritische Stellen (und damit weniger Schritte bei der Vervollständigung !).
Konnte eine Regel zu einer anderen reduziert werden, so können mit dieser neuen Regel andere Regeln nun reduzierbar werden und der Prozess iteriert sich. Allerdings bricht dieser Prozess bei Noetherschen Regelsystemen ab. Da die Interreduktion im KB-Algorithmus nach jeder neuen Regel durchgeführt wird, sollte im Suchbaum ein Ast zu einem Blatt gefunden werden, bei dem möglichst wenige Reduktionsversuche und Normalformbildungen benötigt werden. Dabei ist es sehr hilfreich, daß vor einer Interreduktion das Regelsystem R aus einem Regelsystem R_{int}, das in sich interreduziert ist, und einem Regelsystem $R_{neu} = R \backslash R_{int}$ besteht.
Im Beispiel in Bild 1 gehen wir davon aus, daß immer nur eine Normalform eines Terms berechnet wird. Außerdem benutzen wir beim Übergang von einem Knoten zu einem Nachfolger jeweils nur eine Regel aus R_{neu}, um alle anderen Regeln in R auf Reduzierbarkeit zu testen. Natürlich sind auch andere Strategien denkbar, wie zum Beispiel immer eine Regel aus R_{int} mit allen Regeln aus R_{neu} zu testen, aber dadurch hätten die Knoten zu viele Nachfolger um sie hier darzustellen. Sei also
$R = R_{int} \cup R_{neu}$,
$R_{int} = \{h(c,f(b,x))\rightarrow b,\ i(g(x,y))\rightarrow y,\ f(b,g(c,d))\rightarrow b\}$, $R_{neu} = \{h(x,y)\rightarrow y,\ i(x)\rightarrow x\}$.
Die Kanten sind zur besseren Lesbarkeit nicht mit R_{out} und R_{in} markiert. Die Elemente aus R_{out} sind natürlich die Regeln, die nicht mehr vorhanden sind und die Elemente aus R_{in} sind die neuen Regeln, die zu R_{neu} hinzukommen. Die Elemente von R_{neu} stehen in den Knoten oberhalb des Striches.
An diesem Beispiel ist erkennbar, daß das interreduzierte Regelsystem auf Pfaden verschiedener Länge erreicht werden kann. Für die Effizienz der Vervollständigung sollte natürlich ein möglichst kurzer Pfad gewählt werden. Aber die Interreduktion kann auch zu verschiedenen Blättern führen, wie das nächste, stark gekürzte, Beispiel zeigt.

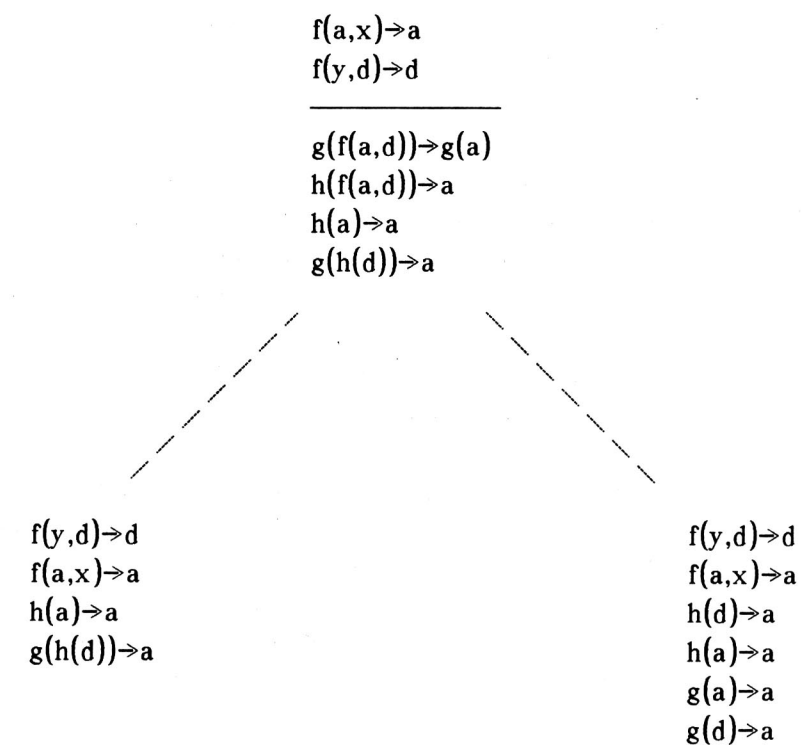
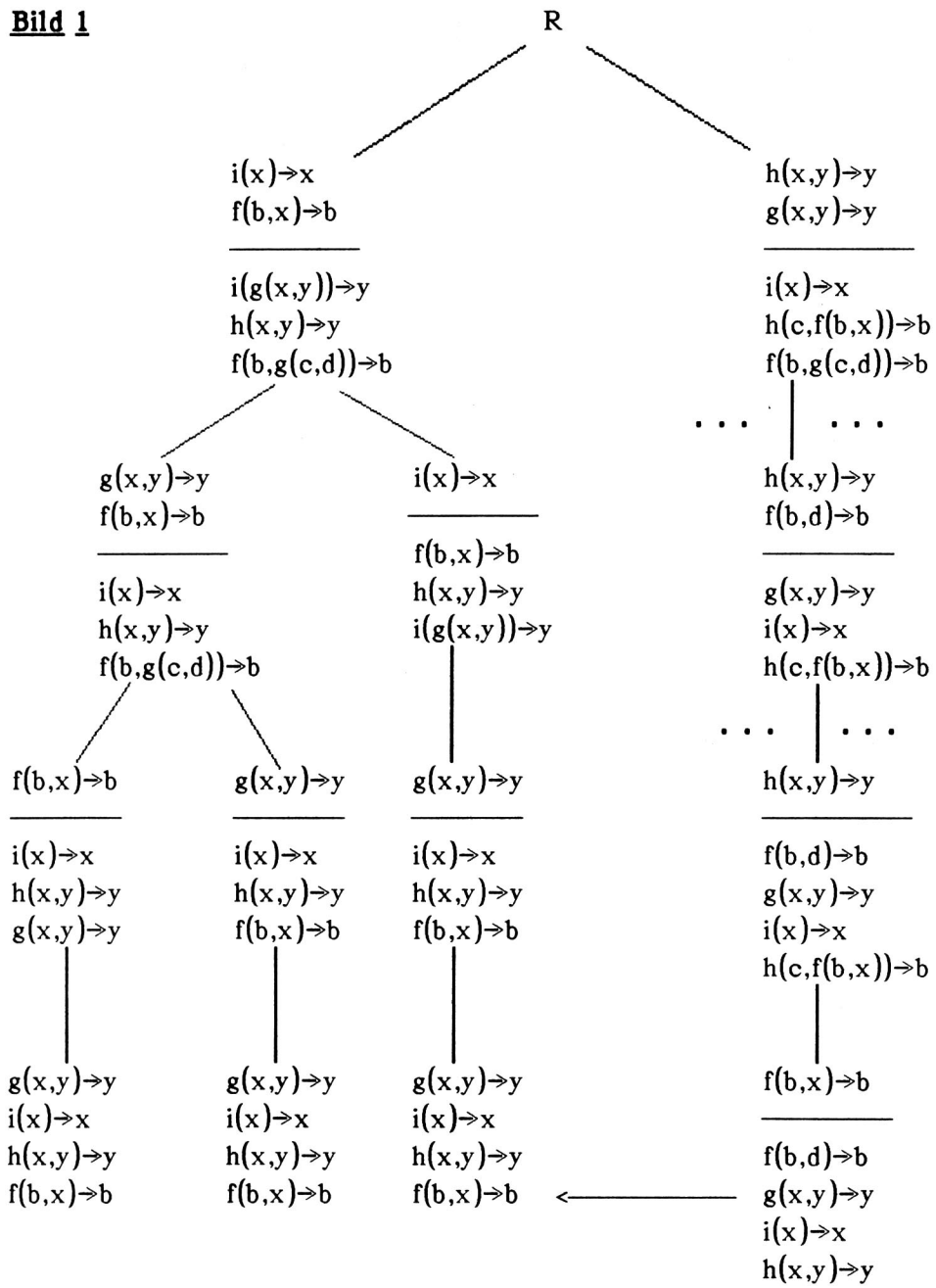


Bild 1



Es sind folgende Kriterien zur Einschränkung des Suchraums denkbar :

- Gestalt der Regeln : wenige Funktionssymbole (= klein), viele verschiedene Variablen (= allgemein anwendbar)
- Ähnlichkeit der Regeln mit gewünschten Regeln

etc.

4.4. Kritische Paare

Die zentrale Idee der Vervollständigung ist die Erzwingung der Konfluenz eines Regelsystems durch Hinzunahme neuer Regeln. Dazu sind zunächst Terme zu erkennen, die zu Divergenzen bei der Reduktion führen (kritische Stellen) und dann sind diese Divergenzen durch gezielte Aufnahme einer oder mehrerer neuer Regeln in das Regelsystem aufzulösen, also die Konfluenz herzustellen. Dabei kann eine kritische Stelle durch verschiedene Mengen neuer Regeln aufgelöst werden (siehe 4.4b). Dies führt zu einer größeren Anzahl an Verzweigungen. Das eigentliche Suchproblem besteht aber in der Abfolge in der Behandlung der kritischen Stellen (siehe 4.4a).

4.4a) Abarbeiten kritischer Stellen

Durch die Vervollständigung soll ein konfluentes Regelsystem entstehen. Man kann an Hand der Regeln feststellen, ob ein Regelsystem schon konfluent ist, indem man die kritischen Stellen auf Konfluenz untersucht. Mit einer kritischen Stelle sind immer zwei Regeln verbunden. Eine kritische Stelle zwischen zwei Regeln ist eine Position n in einer der Regeln $l \to r$, sodaß l/n mit der linken Seite der anderen Regel $s \to t$ unifizierbar ist. Der Term $\sigma(l)[n \leftarrow \sigma(s)]$, wobei σ der mgu zu l/n und s ist, läßt sich dann mit beiden Regeln reduzieren. Zum einen entsteht mit $l \to r$ $\sigma(r)$ und zum anderen mit $s \to t$ $\sigma(l)[n \leftarrow \sigma(t)]$. Die Terme $\sigma(r)$ und $\sigma(l)[n \leftarrow \sigma(t)]$ heißen dann kritisches Paar.
Zur Beschreibung des Suchproblems "kritische Paare" benutzen wir die Menge CP zu einem Regelsystem R, die alle die kritischen Stellen zu Regeln in R enthält, von denen noch nicht bekannt ist, ob sie in R zusammenführbar sind.

Wurzel : Ein Regelsystem R und die Menge CP aller kritischen Stellen zu R.

innere Knoten : Ein Regelsystem R und eine nicht-leere Menge CP.

Blätter : Ein Regelsystem R (eindeutig terminierend) und eine leere Menge CP oder **fail.**

Kanten : Ein Element aus CP der Form ($l \to r \in R$, $s \to t \in R$, n Position in l, σ) und eine Menge NR neuer Regeln.

Übergang : Für das Element aus CP, das an der Kante steht, wird das kritische Paar gebildet und dieses kritische Paar ist in dem System R des Nachfolgeknotens, das aus dem System des Vorgängers durch Addition der Menge NR hervorgeht, zusammenführbar. Die Menge CP des Nachfolgeknotens wird um alle kritischen Positionen, die mit eventuell neu hinzugekommenen Regeln gebildet werden können erweitert, während das an der Kante stehende Element aus CP gelöscht wird. Soll die Interreduktion benutzt werden, so gilt außerdem für das System R, daß es interreduziert ist.

Beim Übergang von einem Knoten zu einem anderen wird also eine kritische Stelle durch Hinzunahme einer Menge von Regeln NR "unkritisch", das heißt, es gibt einen Term, auf den beide Terme des kritischen Paares zu dieser Stelle mit dem neuen Regelsystem reduziert werden können. Durch das Abarbeiten einer kritischen Stelle können auch

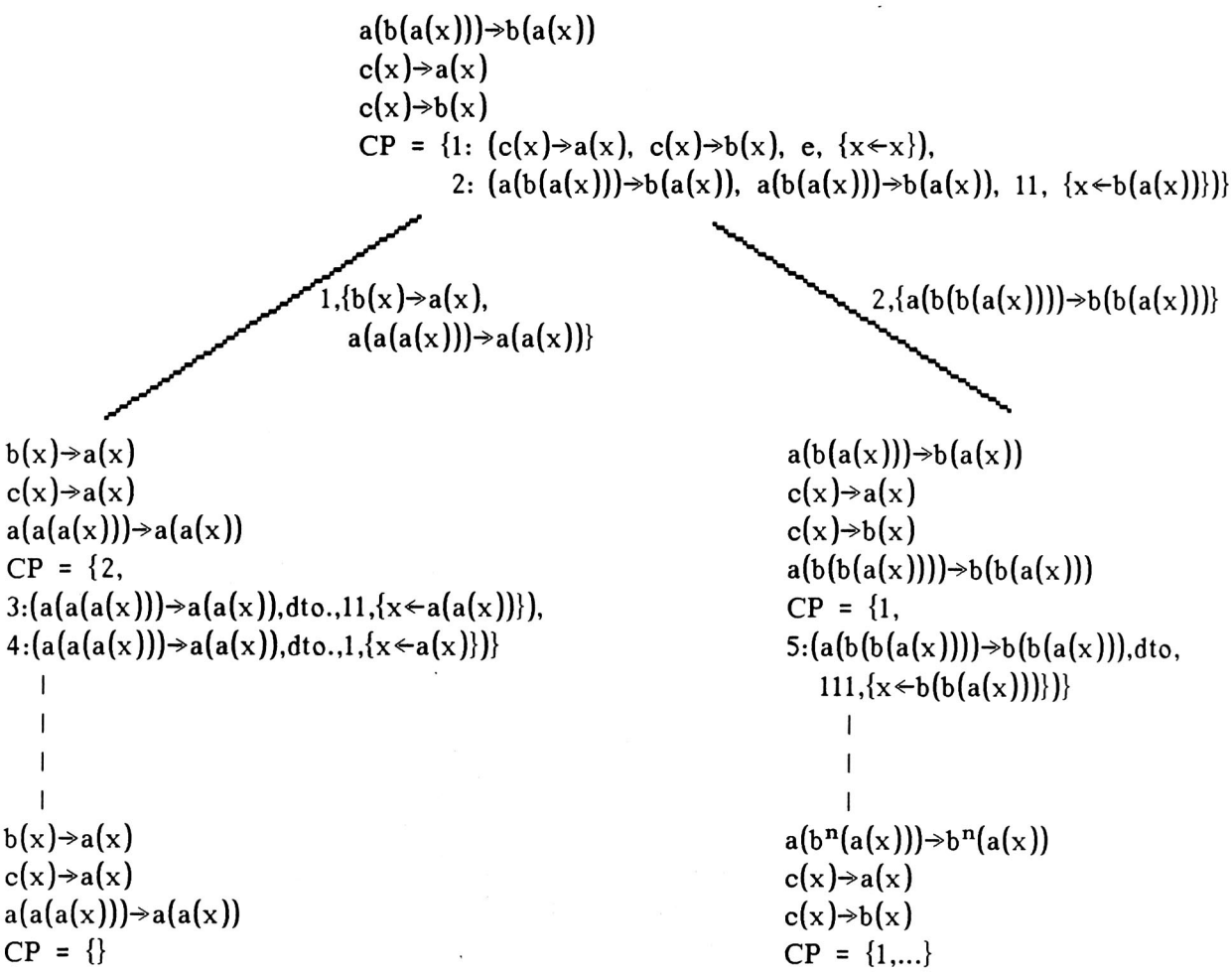
andere kritische Stellen "unkritisch", d.h. zusammenführbar werden. Wenn diese Elemente aus CP dann abgearbeitet werden, ist die Menge NR leer. (Man könnte natürlich auch an eine Kante mehrere Elemente aus CP schreiben. Dadurch würde allerdings die Anzahl der Nachfolger eines Knotens drastisch erhöht. Deswegen haben wir darauf verzichtet.) Weiter können Verfahren, die Kriterien zur Bestimmung und Elimination "unkritischer" kritischer Stellen, wie z.B. die Verfahren von Kapur et al. ([KMN88]) oder Winkler ([Wi84]), benutzen, als Heuristiken zur Begrenzung des Suchbaumes angesehen werden. Eine kritische Stelle ist allgemein dann nicht mehr "kritisch", wenn das dazugehörige kritische Paar konfluent ist. Das Garantieren der Konfluenz wird in 4.4b. beschrieben.
Das Ziel weiterer Heuristiken wird es sein, einen möglichst kurzen Pfad zu einem nicht-**fail**-Blatt zu finden. Deshalb sollten durch Heuristiken solche kritischen Stellen abgearbeitet werden, die entweder zu mächtigen (für die Interreduktion !) Regeln oder zu Regeln, die im eindeutig terminierenden Endsystem vorkommen, führen. Durch die Interreduktion werden Regeln und Gleichungen durch andere ersetzt oder gelöscht. Alle kritischen Stellen mit diesen Regeln und Gleichungen sind damit hinfällig. Wie oben bei "unkritisch" gewordenen kritischen Stellen werden diese kritischen Stellen durch Kanten mit leerer Regelmenge NR abgearbeitet.
Außerdem führt jede Regel, die während des Übergangs von einem Knoten zu einem Nachfolger in das Regelsystem aufgenommen wird, zu neuen kritischen Stellen, die in die Menge CP des Nachfolgers aufgenommen werden müssen.
Bevor wir den Suchbaum für ein Beispiel aufbauen, muß noch geklärt werden, wie eine kritische Stelle zu behandeln ist, damit sie danach "unkritisch" ist.

4.4b) Behandlung kritischer Stellen

Das Problem, die Konfluenz kritischer Stellen zu garantieren, läßt sich im Prinzip auf die Normalformbildung von Termen zurückführen. Aus einer kritischen Stelle gewinnt man ein kritisches Paar, also zwei Terme. Eine kritische Stelle ist oder wird "unkritisch", wenn diese beiden Terme zusammenführbar sind, also einen gemeinsamen Nachfolger haben. Diese Nachfolger entstehen durch Reduktion der Terme mit dem aktuellen Regelsystem. Gibt es keinen gemeinsamen Nachfolger der Terme, so wird die Konfluenz durch Hinzunahme neuer Regeln, die aus Nachfolgern der Terme gebildet werden, erzwungen.
In vielen Verfahren wird zu beiden Termen jeweils eine Normalform gebildet und aus diesen Normalformen eine Regel gemacht, sofern sie nicht gleich sind. Da, wie in 4.2. gesehen, Terme während des Vervollständigungsprozesses mehrere Normalformen haben können, kann es dabei also vorkommen, daß zwei Terme eine gemeinsame Normalform haben, diese aber nicht erzeugt wird, und deshalb eine unnötige Regel (= unnötige Erweiterung des Suchraums) erzeugt wird.
Andererseits kann es in manchen Anwendungen vorteilhaft sein, alle Normalformen der beiden Terme zu bilden und dann mehrere Regeln zu erzeugen, eventuell auch dann, wenn es eine gemeinsame Normalform gibt. Zum Beispiel kann man so manchmal vermeiden, aus zwei unvergleichbaren Termen eine Regel machen zu müssen, was ja gleichbedeutend mit einem Abbruch des Algorithmus ohne Erfolg wäre.
Deswegen haben wir in 4.4a die Kanten nicht nur mit dem Element aus CP beschriftet, sondern auch mit den hinzugenommenen Regeln, da ja eine kritische Stelle auf mehrere

Arten (= mit mehreren verschiedenen neuen Regelsystemen) abgearbeitet werden kann.
Im folgenden Beispiel werden wir eine kritische Stelle jeweils nur auf eine Art abarbeiten, da wir zunächst zeigen wollen, daß die Wahl der abzuarbeitenden Stelle schon zu sehr unterschiedlichen Regelmengen führen kann.
Wir betrachten folgende Regelmenge :

R = {a(b(a(x)))→b(a(x)),
c(x)→a(x),
c(x)→b(x)}



5. Fazit

Wir haben in diesem Bericht die vier beim KB-Algorithmus auftretenden Suchprobleme vorgestellt. Dabei haben wir diese Suchprobleme in einer einheitlichen Form dargestellt und erläutert. Außerdem versuchten wir, zu jedem Suchproblem Ansatzpunkte für die Entwicklung von Heuristiken zu geben. Nun sind noch die Auswirkungen des Suchproblemansatzes auf die Effizienz der Vervollständigung gegenüber dem herkömmlichen (linear nur einen Zweig betrachtenden) Ansatz näher zu betrachten.

Die Wichtigkeit der Zurücknahme einer getroffenen Wahl bei der Reduktionsordnung wurde schon in vielen Systemen erkannt. So bietet zum Beispiel das System RRL [KZ86] durch die "undo"-Operation Backtracking-Möglichkeiten zu früher vorgenommenen Entscheidungen, die im wesentlichen mit der Ordnung zusammenhängen. Damit kann die Berechnung nach Entdeckung eines fail-Blattes an einer früheren Stelle in anderer Weise wieder aufgenommen werden. Allerdings sind weiterhin durch die Ordnung bedingte, unendliche Berechnungsfolgen (Divergenz) möglich, die durch Wahl einer anderen Ordnung vermieden werden könnten. Eine Implementierung des KB-Algorithmus als Suchproblem mit dem Schwerpunkt Reduktionsordnung könnte in diesem Fall Abhilfe schaffen! Allerdings besteht auch dann das Manko, das auch auf alle anderen hier vorgestellten Suchprobleme zutrifft, nämlich, daß eine solche Implementierung einen sehr großen Speicherbedarf hat, da immer mehrere Knoten des Suchbaums und dadurch mehrere Regelsysteme und die dazugehörigen Verwaltungsinformationen gespeichert sein müssen.
Versucht man nun, die Größe des Suchbaums durch Heuristiken klein zu halten, so kann im speziellen Fall der Wahl der Reduktionsordnung gerade die Lösung, die zu einem endlichen Regelsystem führt, übergangen werden. Allerdings dürften die erwähnten "Ähnlichkeits"-Heuristiken, die als ein Schritt in Richtung wissensbasiertes Vervollständigen angesehen werden können, sehr geeignet sein, die gewünschten Lösungen nicht zu eliminieren.
Das Suchproblem der Bestimmung der Normalform eines Terms und damit verbunden das Suchproblem des Findens eines interreduzierten Regelsystems wird bei einer Implementierung meistens weniger effizient sein als lineare Implementierungen dieser Funktionen. Wie wir in den Beispielen gesehen haben, werden die selben Blätter auf vielen, unterschiedlich langen, Pfaden erreicht und die Anzahl der wirklich verschiedenen Lösungen ist meistens nicht groß. Für Wortersetzungssysteme wurde im System COSY [Sa86] das Finden der Normalform(en) eines Terms als Suchproblem implementiert. Dabei gab es in einigen Beispielen, trotz des höheren Aufwands beim Berechnen der Normalform(en), durch die größere Auswahl an kritischen Paaren Effizienzgewinne. Insbesondere wenn der Vervollständigungsprozess durch "Ähnlichkeits"- Heuristiken gesteuert werden soll, sind möglichst viele Lösungen der beiden Probleme interessant, sodaß der höhere Aufwand gegenüber linearen Implementierungen durch Effizienzsteigerungen beim Gesamtalgorithmus wettgemacht werden kann. Dies liegt in der in 4.4b) geschilderten Verzahnung beider Teiloperationen mit der Abarbeitung kritischer Stellen begründet.
Das Suchproblem der nächsten abzuarbeitenden kritischen Stelle schließlich stellt eine große Möglichkeit zu Effizienzsteigerungen dar. Auch in linearen Implementierungen ist hier der Aufsatzpunkt für Heuristiken, nur können diese nur das aktuelle Regelsystem und die aktuellen kritischen Stellen zur Beurteilung und Auswahl heranziehen. Außerdem kann eine einmal getroffene Entscheidung nicht mehr rückgängig gemacht werden.
Implementiert man die Vervollständigung in der in 4.4. geschilderten Weise als Suchproblem, so können die Auswirkungen der Wahl einer kritischen Stelle zur Bearbeitung im Nachfolgeknoten und dessen Nachfolgern festgestellt werden (d.h. wie groß ist das Regelsystem nach der Interreduktion, wurden nicht-richtbare Gleichungen erzeugt, wieviele neue Regeln wurden erzeugt, etc.). Je nach Ergebnis kann der Nachfolgeknoten mit hoher oder niedriger Priorität weiterentwickelt werden. Außerdem kann in jedem

Schritt an einer anderen Stelle des Baumes weiter entwickelt werden.
Da die Knoten auf der selben Ebene des Baumes völlig unabhängig voneinander sind, ergibt sich hier auch die Möglichkeit der Parallelverarbeitung. Leider besteht ein großes Problem dieser Art der Implementierung der Vervollständigung in der großen Zahl von Nachfolgern eines Knotens. Möchte man alle Nachfolger in der in 4.4a) beschriebenen Art berechnen und weiterverfolgen, so werden die Grenzen der Speicherkapazität eines Rechners schnell erreicht.
Deswegen empfiehlt es sich, jeweils nur einige durch Heuristiken bestimmte Nachfolger des aktuell ausgewählten Knotens zu berechnen. Diese Auswahl kann durch ähnliche Kriterien wie bei linearen Implementierungen erfolgen. Nach einigen Schritten sollten dann alle erzeugten Knoten beurteilt werden und nocheinmal ein großer Teil davon eliminiert werden. Im Zuge dieser Beurteilung könnten auch neue Knoten erzeugt werden, die die "guten" Teile (Regeln und wirklich "kritische" Stellen) mehrerer anderer Knoten in sich vereinigen, sodaß die Berechnung der nun eliminierten Knoten nicht völlig unnötig war.

5. Literaturverzeichnis

[Der85] Dershowitz, N. :
Termination.
1st RTA, LNCS 202, 1985, pp. 180-224

[KB70] Knuth, D.E. ; Bendix, P.B. :
Simple Word Problems in Universal Algebra.
Computational Algebra, J. Leech, Pergamon Press, 1970, pp. 263-297

[KL80] Kamin, S. ; Levy, J.-J. :
Attempts for generalizing the recursive path orderings.
Unveröffentlichter Bericht, Department of Computer Science, University of Illinois, 1980

[KMN88] Kapur, D. ; Musser, D.R. ; Narendran, P. :
Only Prime Superpositions Need be Considered in the Knuth-Bendix Completion Procedure.
J. of Symbolic Comp. 6, 1988, pp. 19-36

[KZ86] Kapur , D. ; Zhang, H. :
RRL : A Rewrite Rule Laboratory, User's Manual.
Unveröffentlichtes Papier bei General Electric, 1986

[Ni80] Nilsson, N.J. :
Principles of Artificial Intelligence.
Tioga, Palo Alto, Calif., 1980

[Sa86] Sattler-Klein, A. :
COSY - Ein benutzerfreundliches Testsystem für Reduktionsstrategien in Wortersetzungssystemen.
Projektarbeit, Universität Kaiserslautern, 1986

[Wi84] Winkler, F. :
The Church-Rosser Property in Computer Algebra and Special Theorem Proving : an Investigation of Critical Pair Completion Algorithms.
Dissertation der J.-Kepler-Universität, Linz, 1984