Über die Entscheidbarkeit der formalen Erreichbarkeit von Prozeduren bei Monadischen Programmen

Lippe, Wolfram-Manfred

First published in:
Programmiersprachen, S. 124 - 134, Berlin 1976, ISBN 3-540-07619-0

ÜBER DIE ENTSCHEIDBARKEIT DER FORMALEN ERREICHBARKEIT VON PROZEDUREN BEI MONADISCHEN PROGRAMMEN

Wolfram-Manfred Lippe

Institut für Informatik und angewandte Mathematik
Christian-Albrechts-Universität, D-23 Kiel

1. Einleitung

Beim Bau von Übersetzern für ALGOL-ähnliche Programmiersprachen erfordert die Behandlung von Prozeduren einen besonderen Aufwand. Deswegen ist es wünschenswert, Algorithmen zu besitzen, die gewisse Eigenschaften von Programmen mit Prozeduren bereits zur Übersetzungszeit entscheiden können, um so durch eine differenziertere Behandlung der Prozeduren unnötigen Aufwand einzusparen und effizientere Zielprogramme zu erzeugen. Zu diesen Eigenschaften gehören z.B.

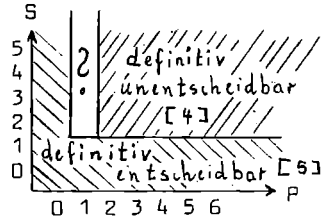
1) die formal korrekte Parameterübergabe in einem Programm mit Prozeduren,
2) die formale Erreichbarkeit einer Prozedur,
3) die formale Rekursivität einer Prozedur,
4) die formal starke Rekursivität einer Prozedur,
5) die Makro-Programm-Eigenschaft eines Programms mit Prozeduren.

Bei Programmen mit formal korrekter Parameterübergabe braucht zur Laufzeit nicht geprüft zu werden, ob aktuelle und formale Parameter zueinander passen. Nicht formal stark rekursive Prozeduren und erst recht nicht formal rekursive Prozeduren dürfen wie Blöcke behandelt werden, d.h. ihre Festspeicher können innerhalb der Festspeicher der kleinsten umfassenden Prozeduren untergebracht werden, und eigene Display- (Index-) Register sind unnötig. Programme mit der Makro-Programm-Eigenschaft benötigen zur Laufzeit überhaupt keine Prozedurorganisation, was die Laufzeit erheblich verkürzt. Gewiß kann die makroartige Expansion des Originalprogramms zu unerträglichen statischen Programmlängen führen, in der Systemprogrammierung, insbesondere beim Compilerbau, wird man den erhöhten Speicherbedarf trotzdem in Kauf nehmen, weil sonst wegen der häufigen Prozeduraufrufe ein relativ großer Zeitverlust eintritt.

Für ALGOL 60 sind die Eigenschaften 1)-5) unentscheidbar, ebenso 2)-5) für ALGOL 68, selbst wenn nur Identifikatoren als aktuelle Parameter zugelassen sind [5,6].

Die bis jetzt bekannten Resultate bzgl. der formalen Erreichbarkeit von Prozeduren in generellen Programmen lassen sich durch folgendes Diagramm reprä-

sentieren, das die Programme nach maximaler Parameterzahl P pro Prozedur und nach maximaler Prozedurschachtelungstiefe S einteilt:



Wir wollen jetzt eine umfangreiche Programmteilklasse definieren, und für sie die Entscheidbarkeit nachzuweisen versuchen. Der definitive Entscheidbarkeitsbereich sollte natürlich möglichst weit über den obigen hinausragen.

Zum Beweis der Unentscheidbarkeit der formalen Erreichbarkeit von Prozeduren wird in [5] für jedes Post'sche Korrespondenzsystem $\mathfrak{L}$ ein Programm $\mathfrak{N}_2$ mit einer Prozedur M1 effektiv konstruiert, so daß $\mathfrak{L}$ genau dann eine Lösung hat, wenn M1 in $\mathfrak{N}_2$ formal erreichbar ist. Es fällt auf, daß in $\mathfrak{N}_2$ die entscheidenden Prozeduranweisungen $a_0(a_1,\ldots,a_n)$ von der Art sind, daß alle formalen Identifikatoren a_i paarweise verschieden sind. Wenn wir nun ein Programm monadisch nennen, falls in jeder Prozeduranweisung $a_0(a_1,\ldots,a_n)$ für die formalen Identifikatoren a_i,a_j $a_i=a_j$ ist, so führt der zitierte Beweis zu der Vermutung, daß die formale Erreichbarkeit von Prozeduren für monadische ALGOL 60-Programme entscheidbar ist.

Diese Vermutung kann noch nicht vollständig bewiesen werden. Der Beweis ist jedoch für zwei große Teilklassen monadischer Programme möglich, erstens für monadische Programme mit maximaler Prozedurschachtelungstiefe 2 und maximaler Parameterzahl 2 pro Prozedur, zweitens für monadische Programme mit maximaler Prozedurschachtelungstiefe 3 und maximaler Parameterzahl 1 pro Prozedur. Aus dem Beweis des ersten Falls ist ersichtlich, daß die Beschränkung der Parameterzahl im Grunde unerheblich ist. Im ersten Fall benutzen wir als Hilfsmittel Baumsysteme [8], im zweiten Fall Stacksysteme [1]. Von beiden Systemen ist bekannt, daß das Zustandserreichbarkeitsproblem algorithmisch lösbar ist.
Der Einfachheit halber beschränken wir uns auf die Sprache ALGOL 60-P (P=pure), die gegenüber ALGOL 60 folgende hauptsächlichen Einschränkungen und Veränderungen besitzt:

a) keine Funktionsprozeduren,
b) kein "call by value",
c) nur Identifikatoren als aktuelle Parameter von Prozeduranweisungen,
d) alle Prozedurrümpfe sind in Rumpfklammern eingeschlossen,
e) leerer Spezifikationsteil im Prozedurkopf.

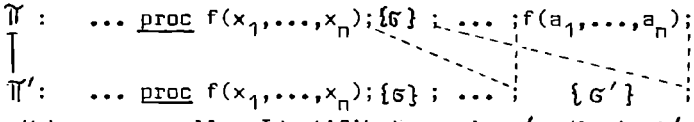
Für eine vollständige Definition von ALGOL 60-P siehe [5].

2. Definitionen

Ein formales Algol 60-P-Programm π ist eine Zeichenkette aus Grundsymbolen, die durch die formalen Regeln der kontextfreien Grammatik $\mathcal{G}_P$ für ALGOL 60-P auf das Axiom <program> reduziert werden kann und für die die folgende Eigenschaft gilt: Zu jedem auftretenden Identifikator gibt es genau ein zugehöriges definierendes Auftreten. Ein formales Programm heißt übersetzbar, falls jedes Auftreten einer Konstanten oder eines Identifikators gemäß seiner Definition vernünftig erfolgt. Zwei formale Programme heißen identisch, falls sie sich nur durch eine zulässige Umbenennung von Identifikatoren unterscheiden, und wir wollen in Zukunft nicht zwischen identischen Programmen unterscheiden. Ein formales Programm heißt ausgezeichnet, falls verschiedene definierende Vorkommen von Identifikatoren durch verschiedene Identifikatoren bezeichnet sind. Ein formales Programm heißt partiell übersetzbar, falls nach dem Ersetzen aller Prozedurrümpfe durch leere Rümpfe das sich so ergebende Programm übersetzbar ist.

Definition: Sei π partiell übersetzbar. Ein Programm π' heißt Ergebnis aus π durch Anwendung der Kopierregel ($\pi \longmapsto \pi'$) falls gilt:

Sei $f(a_1,\ldots,a_n)$ eine Prozeduranweisung im Hauptprogramm von π. Sei ferner proc $f(x_1,\ldots,x_n);\{\sigma\}$ die zugehörige Prozedurdeklaration. Die partielle Übersetzbarkeit von π garantiert, daß die Anzahl der aktuellen und formalen Parameter übereinstimmt. Sei ferner π o.B.d.A. ausgezeichnet. Dann wird $f(a_1,..,a_n)$ durch einen modifizierten Rumpf $\{\sigma'\}$ ersetzt, indem die formalen Parameter x_i, die in $\{\sigma\}$ auftreten, durch die entsprechenden aktuellen Parameter a_i ersetzt werden.



Durch Umbenennung aller Identifikatoren in σ', die in σ' selbst gebunden sind (lokal), wird π' ausgezeichnet. $\{\sigma'\}$ heißt erzeugter Block; die Klammern { } bei $\{\sigma'\}$ heißen Aufrufklammern im Gegensatz zu den Rumpfklammern { } bei $\{\sigma\}$. $\overset{+}{\longmapsto}$ und $\overset{*}{\longmapsto}$ sollen die transitive bzw. transitive reflexive Hülle von $\longmapsto$ bezeichnen. Ein formales Programm π heißt original, falls { } nur als Rumpfklammern auftreten.

Definition: Sei π original. Die Menge T_π mit

$$T_\pi := \{\pi' / \pi \overset{*}{\longmapsto} \pi',\ \pi' \text{ besitzt höchstens einen innersten erzeugten Block}\}$$

heißt Ausführungsbaum von π.

Mit IGB(π') bezeichnen wir den innersten erzeugten Block von π'. Entfernen wir bei einem Programm $\pi' \in T_\pi$ alle Prozedurdeklarationen, so erhalten wir das zugehörige Hauptprogramm π'_m. Ersetzen wir alle Prozeduranweisungen in π'_m durch ein Sonderzeichen, z.B. call, so erhalten wir das reduzierte Hauptprogramm π_r

von π.

Definition: $T_{r\pi} := \{\pi'_r / \pi' \in T_\pi\}$ heißt reduzierter Ausführungsbaum von π.

Definition: Zwei originale Programme heißen formal äquivalent, falls ihre reduzierten Ausführungsbäume identisch sind.

Definition: Eine Prozedur in einem originalen Programm π heißt formal erreichbar, falls ein Programm $\pi' \in T_\pi$ ex., so daß IGB(π') ein modifizierter Rumpf einer Kopie dieser Prozedur ist.

Im folgenden werden wir uns mit der formalen Erreichbarkeit bei einem besonderen Typ von Programmen, den monadischen Programmen, beschäftigen.

Definition: Ein Programm π heißt monadisch, wenn in jeder Prozeduranweisung $a_0(a_1,\ldots,a_n)$ für die formalen Identifikatoren a_i, a_j gilt $a_i = a_j$.

Um die Beweise zu vereinfachen, machen wir ferner die unerhebliche Einschränkung, daß es neben Prozedurdeklarationen keine weiteren Deklarationen und neben Prozeduranweisungen keine weiteren Anweisungen geben soll.

Zum Beweis, daß die formale Erreichbarkeit von Prozeduren bei monadischen Programmen entscheidbar ist, führen wir unsere ALGOL 60-P-Programme in formal äquivalente Programme der modularen Programmiersprache ALGOL 60-P-G (closure-Sprache) über. Prozedurdeklarationen in ALGOL 60-P-G haben die Form

$$(*) \quad \underline{\text{proc}}\ f\ \langle y_1,\ldots,y_{m_f}\rangle\ (x_1,\ldots,x_{n_f});\ \{\S\};$$

Die Identifikatoren $y_1,\ldots,y_{m_f}$ heißen formale Parameter neuer Art, $x_1,\ldots,x_{n_f}$ sind die formalen Parameter alter Art. Die Klammern fallen weg, falls m_f bzw. n_f Null ist. Alle übrigen Deklarationen sehen wie in ALGOL 60-P aus. Ein Term ist eine endliche Zeichenreihe, die nach folgenden Regeln gebildet wird:

1. Identifikatoren sind Terme.
2. Wenn γ Identifikator ist und $\tau_1,\ldots,\tau_m$, $m \geq 1$, Terme sind, dann ist auch $\gamma\langle\tau_1,\ldots,\tau_m\rangle$ ein Term. τ_i heißt aktueller Parameter neuer Art.

Aus einem formalen ALGOL 60-P-Programm π' entsteht ein formales ALGOL 60-P-G-Programm π dadurch, daß die Prozedurköpfe in π' durch neue Köpfe der Form (*) ersetzt werden und daß man in π' angewandt vorkommende Identifikatoren durch Terme auswechselt, so daß jeder Identifikator genau eine Definition besitzt. Auch alle übrigen Begriffe lassen sich ohne Schwierigkeiten auf ALGOL 60-P-G ausweiten.

Der Unterschied zwischen den beiden Sprachen liegt in der Methode, wie man Prozeduren f, die durch formale Prozeduranweisungen aufgerufen werden, durch aktuelle Parameter versorgt. In ALGOL 60-P werden alle aktuellen Parameter im Moment des Aufrufs übergeben. In ALGOL 60-P-G werden die aktuellen Parameter alter Art ebenfalls im Moment des Aufrufs übergeben, während die aktuellen Parameter neuer Art so betrachtet werden können, als ob sie bereits vorher übergeben worden wären, als der Prozeduridentifikator f als aktueller Parameter eines vorange-

gangenen Aufrufs auftrat. Auf Grund dieser Wirksamkeit der formalen Parameter neuer Art ist es möglich, jedes originale ALGOL 60-P-G-Programm in ein formal äquivalentes ALGOL 60-P-G-Programm ohne Prozedurschachtelung (modular) effektiv umzuformen.

3. Baumsysteme

Betrachtet man nur Programme mit maximaler Parameterzahl 1 und maximaler Prozedurschachtelungstiefe 2, so besitzen alle Terme, die bei den Prozeduranweisungen des Ausführungsbaumes auftreten können, die Form einer linearen Zeichenkette. Dieser Spezialfall wurde bereits in [7] untersucht, und die Entscheidbarkeit der formalen Erreichbarkeit mit Hilfe von regulären kanonischen Systemen gezeigt. Erhöht man bei den Programmen die Parameteranzahl, bzw. die Prozedurschachtelungstiefe, so besitzen die einzelnen Terme eine baumartige Struktur. Im Gegensatz zu den Baumautomaten, wie sie von Brainerd, Doner usw. betrachtet wurden, benötigen wir jedoch Systeme, die eine Modifikation des Eingabebaums durch Veränderung der Wurzel erlauben. Deswegen benutzen wir Baumsysteme, wie sie von Rounds [8] eingeführt wurden.

Definition: Ein markiertes Alphabet ist ein Paar (Σ,r), wobei Σ eine endliche Menge und $r: \Sigma \longrightarrow \mathbb{N}$ eine Abbildung ist. Sei ferner Σ_n durch

$$\Sigma_n := \{\sigma \in \Sigma / r(\sigma)=n\}$$

definiert.

Hiermit lassen sich nun Σ - Terme (Bäume) definieren:

Definition: Sei (Σ,r) ein markiertes Alphabet. Die Menge T_Σ° ist die kleinste Menge, derart daß

(a) $\Sigma_o \subseteq T_\Sigma^\circ$

(b) falls $t_o,\ldots,t_{n-1} \in T_\Sigma^\circ$ und $\sigma \in \Sigma_n \Rightarrow \sigma\langle t_o,\ldots,t_{n-1}\rangle \in T_\Sigma^\circ$

Was ein Unterterm eines Terms ist dürfte klar sein. Die t_i, $0 \leq i \leq n-1$, in $\sigma\langle t_o,\ldots,t_{n-1}\rangle$ bezeichnen wir auch als unmittelbare Unterterme, t_o als unmittelbaren linken Unterterm und t_{n-1} als unmittelbaren rechten Unterterm. Um Terme miteinander vergleichen zu können, benötigen wir noch

Definition: Zwei Σ-Terme $\sigma\langle t_o,\ldots,t_{n-1}\rangle$ und $\sigma'\langle t'_o,\ldots,t'_{k-1}\rangle$ heißen fastidentisch, falls $n=k$ und $t_i=t'_i$ für $i=0,\ldots,n-1$.

Definition: Seien $\sigma\langle t_o,\ldots,t_{n-1}\rangle$ und $\sigma'\langle t'_o,\ldots,t'_{k-1}\rangle$ zwei Σ-Terme. Dann heißt $\sigma'\langle t'_o,\ldots,t'_{k-1}\rangle$ Fastunterterm von $\sigma\langle t_o,\ldots,t_{n-1}\rangle$, falls ein Unterterm $\bar\sigma\langle \bar t_o,\ldots,\bar t_{k-1}\rangle$ von $\sigma\langle t_o,\ldots,t_{n-1}\rangle$ ex., so daß $\bar\sigma\langle \bar t_o,\ldots,\bar t_{k-1}\rangle$ und $\sigma'\langle t'_o,\ldots,t'_{k-1}\rangle$ fastidentisch sind.

Ist $\bar\sigma\langle \bar t_o,\ldots,\bar t_{k-1}\rangle$ unmittelbarer Unterterm von $\sigma\langle t_o,\ldots,t_{n-1}\rangle$, so bezeichnen wir $\sigma'\langle t'_o,\ldots,t'_{k-1}\rangle$ als unmittelbaren Fastunterterm von $\sigma\langle t_o,\ldots,t_{n-1}\rangle$.

Es sei A eine endliche Menge mit $A \cap \Sigma = \emptyset$. Wir erweitern Σ zu Σ' mit $\Sigma' = A \cup \Sigma_\emptyset$ und $\Sigma'_n = \Sigma_n$. Statt T_Σ°, schreiben wir auch $T_\Sigma(A)$.

Definition: Sei Q eine Menge sog. Zustände und $q,q' \in Q$. Dann heißt ein Paar $(q, \sigma\langle x_0,\ldots,x_n\rangle) \longrightarrow (q',u)$ mit $\sigma \in \Sigma_n$, $u \in T_\Sigma(\{x_0,\ldots,x_n\})$, $n \geq 0$, eine Baumproduktion. Die x_i bezeichnen wir als Variablen.

Definition: Ein Baumsystem G über Σ ist ein 4-tupel $(\Sigma,Q,S,\mathfrak{P})$ mit einer endlichen Menge Q von Zuständen, einer endlichen Menge $\mathfrak{P}$ von Baumproduktionen über Q und Σ, und S einer endlichen Untermenge von $Q \times T_\Sigma^0$ (Startkonfigurationen).

Die Wirkungsweise eines solchen Baumsystems ist gegeben durch:

Definition: Sei $(q,t),(q',t') \in Q \times T_\Sigma^0$. (q',t') wird aus (q,t) direkt erzeugt ($(q,t) \Longrightarrow (q',t')$), falls gilt:

(a) $t = \sigma\langle t_0,\ldots,t_{n-1}\rangle$, $n \geq 0$

(b) es gibt eine Baumproduktion $(q, \sigma\langle x_0,\ldots,x_{n-1}\rangle) \longrightarrow (q',u)$

(c) t' geht aus u dadurch hervor, daß die x_i in u durch t_i ersetzt werden.

Mit $\overset{*}{\Longrightarrow}$ bezeichnen wir die reflexive transitive Hülle von $\Longrightarrow$.

4. Programme mit zweiparametrigen Prozeduren und Prozedurschachtelungstiefe ≤ 2

Im folgenden beschäftigen wir uns mit Programmen, die nach der Umformung in ALGOL 60-P-G-Programme und der Beseitigung der Prozedurschachtelungen die allgemeine Gestalt

$\mathfrak{P}_1$ = begin proc $g_1(x_1^1,x_1^2)$;{ ... }; (Hauptteil von g_1)
⋮
proc $g_n(x_n^1,x_n^2)$; { ... };
proc $f_{11}\langle \bar{x}_1^1,\bar{x}_1^2\rangle (y_{11}^1,y_{11}^2)$;{ ... }; (Hauptteil von f_{11})
⋮
proc $f_{1m_1}\langle \bar{x}_1^1,\bar{x}_1^2\rangle (y_{1m_1}^1,y_{1m_1}^2)$; { ... };
⋮
proc $f_{n1}\langle \bar{x}_n^1,\bar{x}_n^2\rangle (y_{n1}^1,y_{n1}^2)$; { ... };
⋮
proc $f_{nm_n}\langle \bar{x}_n^1,\bar{x}_n^2\rangle (y_{nm_n}^1,y_{nm_n}^2)$; { ... };
⋮ } Hauptprogramm
end

besitzen.

Im Hauptprogramm können nur Anweisungen der Art $g_i(g_{i'},g_{i''})$, $1 \leq i,i',i'' \leq n$ auftreten.

Bei den Anweisungen $a_0(a_1,a_2)$, die im Hauptteil von g_i (i fest, $1 \leq i \leq n$) auftreten, kann a_k, k=0,1,2, ein formaler Identifikator x_i^1 bzw. x_i^2, ein globaler Identifikator $g_{i'}$ ($1 \leq i' \leq n$) oder ein Term $f_{ij}\langle x_i^1,x_i^2\rangle$ ($1 \leq j \leq m_i$) sein.

Bei den Anweisungen $a_0(a_1,a_2)$, die im Hauptteil von f_{ij} (i,j fest, $1 \leq i \leq n$, $1 \leq j \leq m_i$) auftreten, kann a_k, k=0,1,2, ein formaler Identifikator y_{ij}^1 oder y_{ij}^2 oder $\bar{x}_i^1$ oder $\bar{x}_i^2$, ein globaler Identifikator $g_{i'}$ ($1 \leq i \leq n$) oder ein Term

$f_{ij'}<\bar{x}_i^1,\bar{x}_i^2>$ $(1 \leq j' \leq m_i)$ sein.
Die durch die Kopierregel entstehenden Terme im Hauptprogramm besitzen nun offensichtlich die Gestalt binärer Bäume, deren Blätter aus Identifikatoren g_i bestehen. Den Zusammenhang zwischen den einzelnen Termen einer Anweisung zeigt

Satz 1: Bei allen Prozeduranweisungen $a_0(a_1,a_2)$, die in Hauptteilen von Programmen $\mathfrak{N}_1'$ des Ausführungsbaumes $T_{\mathfrak{N}_1}$ des ALGOL 60-P-G-Programms $\mathfrak{N}_1$ auftreten gilt: Bei je zwei Untertermen t_l,t_k von a_i,a_j ist einer der t_l,t_k Fastunterterm des andern.

Dies bedeutet insbesondere für die Terme a_i,a_j, daß einer stets Fastunterterm des andern ist. Beschränken wir uns auf monadische Programme, so lassen sich die Aussagen über den Zusammenhang zwischen den einzelnen Termen weiter verschärfen:

Satz 2: Bei allen Prozeduranweisungen $a_0(a_1,a_2)$, die in Hauptteilen von Programmen $\tilde{\mathfrak{N}}_1'$ des Ausführungsbaumes $T_{\tilde{\mathfrak{N}}_1}$ eines monadischen ALGOL 60-P-G-Programms $\tilde{\mathfrak{N}}_1$ auftreten, gilt für je zwei Terme a_i,a_j, $0 \leq i,j \leq 2$, eine der folgenden drei Eigenschaften:

a) $a_i=g_{i'}$ oder $a_j=g_{i'}$, $1 \leq i' \leq n$, oder
b) a_i und a_j sind fastidentisch oder
c) a_i ist unmittelbarer Fastunterterm von a_j oder umgekehrt.

Dieser Satz zeigt, daß sich die einzelnen Terme nur in dem unmittelbaren Bereich um die Wurzel unterscheiden. Durch das Notieren der Unterschiede zwischen den einzelnen Termen in einem Index an der Wurzel, lassen sich die einzelnen Terme der Prozeduranweisungen $a_0(a_1,a_2)$ zu einem Term zusammenfassen. Somit läßt sich zu unserem Programm $\tilde{\mathfrak{N}}_1$ ein Baumsystem $G_{\tilde{\mathfrak{N}}_1}$ konstruieren mit der Eigenschaft

Satz 3: Eine Prozedur p in $\tilde{\mathfrak{N}}_1$ ist genau dann formal erreichbar, falls es in $G_{\tilde{\mathfrak{N}}_1}$ eine Startkonfiguration s und es eine durch p bestimmte Konfiguration S_p gibt, so daß $s \overset{*}{\Longrightarrow} S_p$.

Da letzteres jedoch entschieden werden kann [8], erhält man unmittelbar

Satz4: Es ist entscheidbar, ob eine Prozedur in einem monadischen ALGOL 60-P-Programm mit höchstens zweiparametrigen Prozeduren und Prozedurschachtelungstiefe < 2 formal erreichbar ist.

5. Programme mit einparametrigen Prozeduren und Prozedurschachtelungstiefe ≤ 3

Im folgenden beschäftigen wir uns mit Programmen, die nach der Umformung in ALGOL 60-P-G-Programme und der Beseitigung der Prozedurschachtelung die allgemeine Gestalt

$\mathfrak{P}_2$ = begin proc $g_1(x_1)$; { ... };
⋮
proc $g_n(x_n)$; { ... };
proc $f_{11}\langle\bar{x}_1\rangle$ (y_{11}); { ... };
⋮
proc $f_{1m_1}\langle\bar{x}_1\rangle$ (y_{1m_1}); { ... };
⋮
proc $f_{n1}\langle\bar{x}_n\rangle$ (y_{n1}); { ... };
⋮
proc $f_{nm_n}\langle\bar{x}_n\rangle$ (y_{nm_n}); { ... };
proc $p_{111}\langle\bar{x}_1,\bar{y}_{11}\rangle$ (z_{111}); { ... };
⋮
proc $p_{111_{11}}\langle\bar{x}_1,\bar{y}_{11}\rangle$ $(z_{111_{11}})$; { ... };
⋮
proc $p_{nm_n1}\langle\bar{x}_n,\bar{y}_{nm_n}\rangle$ (z_{nm_n1}); { ... };
⋮
proc $p_{nm_n1_{m_n}}\langle\bar{x}_n,\bar{y}_{nm_n}\rangle$ $(z_{nm_n1_{m_n}})$; { ... };
⋮
end

besitzen.

Bei einem solchen Programm können im Hauptprogramm nur Anweisungen der Art $g_i(g_{i'})$, $1\leq i,i'\leq n$, stehen.

Bei den Anweisungen $a_0(a_1)$, die im Hauptteil von g_i (i fest, $1\leq i\leq n$) auftreten, kann a_k, k=0,1, ein formaler Identifikator x_i, ein globaler Identifikator $g_{i'}$, $(1\leq i'\leq n)$ oder ein Term $f_{ij}\langle x_i\rangle$ sein $(1\leq j\leq m_i)$.

Bei den Anweisungen $a_0(a_1)$, die im Hauptteil von f_{ij} (i,j fest, $1\leq i\leq n$, $1\leq j\leq m_i$) auftreten, kann a_k, k=0,1, ein formaler Identifikator alter Art y_{ij}, ein formaler Identifikator neuer Art $\bar{x}_i$, ein globaler Identifikator $g_{i'}$, $(1\leq i'\leq n)$, ein Term $f_{ij'}\langle\bar{x}_i\rangle$ $(1\leq j'\leq m_i)$ oder ein Term $p_{ijl}\langle\bar{x}_i,y_{ij}\rangle$ $(1\leq l\leq l_{ij})$ sein.

Bei den Anweisungen $a_0(a_1)$, die im Hauptteil von p_{ijl} (i,j,l fest, $1\leq i\leq n$, $1\leq j\leq m_i$, $1\leq l\leq l_{ij}$) auftreten, kann a_k, k=0,1, ein formaler Identifikator alter Art z_{ijl}, ein formaler Identifikator neuer Art $\bar{x}_i$ bzw. $\bar{y}_{ij}$, ein globaler Identifikator $g_{i'}$, $(1\leq i'\leq n)$, ein Term $f_{ij'}\langle\bar{x}_i\rangle$ $(1\leq j'\leq m_i)$ oder ein Term $p_{ijl'}\langle\bar{x}_i,\bar{y}_{ij}\rangle$ $(1\leq l'\leq l_{ij})$ sein.

Beschränkt man sich nun auf monadische Programme, so läßt sich zeigen

Satz 5: Für alle Prozeduranweisungen $a_0(a_1)$, die in Programmen $\tilde{\mathfrak{P}}_2$ des Ausführungsbaumes $T_{\tilde{\mathfrak{P}}_2}$ eines monadischen ALGOL 60-P-G-Programms auftreten können, gelten die folgenden beiden Eigenschaften:

Eigenschaft a:

1) $a_0 = g_i$ oder $a_1 = g_i$, $1 \leq i \leq n$, oder
2) a_0 und a_1 sind fastidentisch oder
3) a_0 besitzt nur einen einzigen unmittelbaren Unterterm und λ ist linker unmittelbarer Unterterm von a_1 oder umgekehrt oder
4) a_0 besitzt einen unmittelbaren Unterterm λ und λ und a_1 sind fastidentisch oder umgekehrt oder
5) a_0 besitzt nur einen einzigen unmittelbaren Unterterm λ und λ ist linker unmittelbarer Unterterm eines unmittelbaren Unterterms von a_1 oder umgekehrt.

Eigenschaft b:

Bei allen Termen a_i, i=0,1, der Art $a_i = \alpha\langle\beta_1,\beta_2\rangle$ gilt für die unmittelbaren Unterterme β_1,β_2 stets

1) $\beta_2 = g_i$, $1 \leq i \leq n$, oder
2) β_1 und β_2 sind fastidentisch oder
3) β_2 besitzt nur einen einzigen unmittelbaren Unterterm λ und λ ist linker unmittelbarer Unterterm von β_1 oder
4) β_1 ist einziger oder linker unmittelbarer Unterterm von β_2 oder
5) β_2 besitzt einen unmittelbaren Unterterm λ und β_1 ist einziger oder linker unmittelbarer Unterterm von λ.

Dieser Satz zeigt, daß auch in diesem Fall sich die einzelnen Terme nur in einem engen Bereich an der Spitze unterscheiden und sich somit zu einem einzigen Term zusammenfassen lassen. Eigenschaft b) zeigt darüberhinaus, daß, falls ein Term zwei unmittelbare Unterterme besitzt, diese Unterterme sich ebenfalls nur unwesentlich unterscheiden. Dies ermöglicht uns die Codierung eines Terms in eine lineare Zeichenkette. Somit läßt sich in diesem Fall die Entscheidbarkeit der formalen Erreichbarkeit einer Prozedur über Stacksysteme zeigen.

Definition: Ein Stack-System ist ein 4-tupel $\gamma = (\mathfrak{N}, \mathfrak{A}, S, \mathfrak{P})$ mit

(i) $\mathfrak{N}$ ist eine endliche nicht leere Menge (nicht terminale Symbole),
(ii) $\mathfrak{A}$ ist eine endliche nicht leere Menge (terminale Symbole) mit $\mathfrak{N} \cap \mathfrak{A} = \emptyset$,
(iii) $S \subset \mathfrak{A}\mathfrak{N}$ (Startworte),
(iv) $\mathfrak{P}$ ist eine endliche Menge von Produktionsregeln der Formen (mit $S', S'' \in \mathfrak{N}$; $A, B \in \mathfrak{A}$; Q, Q_1, Q_2 Variablen über $\mathfrak{A}^*$):
   1) $QAS' \longrightarrow QABS''$
   2) $QAS' \longrightarrow QAS''$
   3) $QAS' \longrightarrow QS''$
   4) $Q_1AS'Q_2 \longrightarrow Q_1AS''Q_2$
   5) $Q_1AS'Q_2 \longrightarrow Q_1S''AQ_2$

6) $Q_1 A S' B Q_2 \longrightarrow Q_1 A B S'' Q_2$

Die Arbeitsweise eines Stack-Systems ist gegeben durch

Definition: $x \Rightarrow y$ mit $x,y \in \Phi^* \mathfrak{N} \Phi^*$

$\Rightarrow$: falls $x=wy$, $y=wy'$ mit $w \in \Phi^*$, $y,y' \in \Phi \mathfrak{N}$ ist und in $\mathfrak{P}$ eine Produktion $Qy \longrightarrow Qy'$ ex. oder falls $x=wyw'$, $y=wy'w'$ mit $w,w' \in \Phi^*$, $y,y' \in \Phi \mathfrak{N} \Phi$ ist und in $\mathfrak{P}$ eine Produktion $Q_1 y Q_2 \longrightarrow Q_1 y\ Q_2$ ex..

Mit $\overset{*}{\Rightarrow}$ bezeichnen wir die transitive reflexive Hülle von $\Rightarrow$.

Es läßt sich nun zeigen:

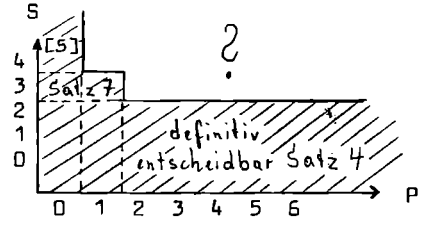
Satz 6: Zu $\tilde{\mathfrak{P}}_2$ läßt sich ein Stack-System $\Gamma_{\tilde{\mathfrak{P}}_2}$ konstruieren mit der Eigenschaft: Eine Prozedur q in $\tilde{\mathfrak{P}}_2$ ist genau dann formal erreichbar, falls es in $\Gamma_{\tilde{\mathfrak{P}}_2}$ ein Startwort s und ein durch q bestimmtes Wort $tw \in \Phi^* \mathfrak{N}$ gibt, so daß gilt $s \overset{*}{\Rightarrow} tw$.

Da letzteres jedoch entscheidbar ist [1], erhält man

Satz 7: Es ist entscheidbar, ob eine Prozedur in einem monadischen ALGOL 60-P-Programm mit höchstens einparametrigen Prozeduren und Prozedurschachtelungstiefe ≤ 3 formal erreichbar ist.

6. Schlußbemerkungen

Für monadische Programme haben wir also folgendes Diagramm



dessen definitiver Entscheidbarkeitsbereich über den früheren hinausragt.

Bei höherer Schachtelungstiefe als 3 im einparametrigen monadischen Fall wird die formale Erreichbarkeit von Prozeduren wohl auch noch entscheidbar sein. Bei höherer Parameterzahl und Schachtelungstiefe ≥ 3 wagen wir im Moment noch keine Prophezeiung.

Literatur:

[1] Ginsburg,S.,Greibach,S.A. und Harrison,M.A.:
Stack automata and compiling, J.ACM. 14,1 (Jan.67), 172-201

[2] Brainerd,W.: Tree generating regular systems,
Information and Control 14 (1969), 217-231

[3] Doner,J.: Tree acceptors and some of their applications,
Journal of Computer and System Sciences 4 (1979), 406-451

[4] Kaufholz,G.,Lippe,W.M.: A note on a paper of H. Langmaack
about procedure parameter transmission, Bericht A 74/06
des Fachbereichs Angewandte Mathematik und Informatik der
Universität des Saarlandes (1974)

[5] Langmaack,H.: On correct procedure parameter transmission in
higher programming languages, Acta Informatoca Vol.2,(1973)

[6] Langmaack,H.: On procedures as open subroutines I,II
Acta Informatica Vol.2, 1973 und Vol.3,(1974)

[7] Lippe,W.M.: Entscheidbarkeitsprobleme bei der Übersetzung
von Programmen mit einparametrigen Prozeduren,
Lecture Notes in Computer Scince Nr.7, Springer Verlag (1974)

[8] Rounds,W.C.: Mappings and grammars on trees,
Mathematical systems theorie, Vol. 4, Nr.3, 257-287