数理解析研究所講究録
第179巻 1973年 141-152

樹状のデータ構造をもつプログラム形

都倉信樹・嵩忠雄
(大阪大学基礎工学部)

1 諸定義

本文で扱うプログラム形は次のような文を用いてかかれる.

(1) ポインタ操作　$P \leftarrow P \cdot up\ (l_k)$;

(2) 〃　$P \leftarrow P \cdot down\ i\ (l_k)$;

(3) 判定　$Q_m(P \cdot s, A_{m1}, \cdots, A_{mr(m)}), l_j, l_k$;

(4) 出力　$P \cdot s' \leftarrow F_t(P \cdot s_{t1}, \cdots, P \cdot s_{tr'(t)}, A_{tr'(t)+1}, \cdots, A_{tr(t)})$;

(5) レジスタ出力　$R_i \leftarrow F_t(P \cdot s_{t1}, \cdots, P \cdot s_{tr'(t)}, A_{tr'(t)+1}, \cdots, A_{tr(t)})$;

(6) 停止　$HALT(z)$;　$z = 1, 2, \cdots, K$.

ここで, l_j, l_k 等はプログラムラベルである. $A_1, \cdots, A_n$ は入力レジスタでこれらへの出力は許されない. $R_1, \cdots, R_m$ は出力レジスタでその内容は参照されない. P は樹状構造上を走査するポインタである. ここでルート(そこへ入射する枝をもたない)を除いて, どの節点もすべて1つの枝が入射するようなアサイクリックな有向グラフを木とよぶ.* 出る枝をもたない節点を端点とよぶ. ポインタ P は木の上を枝に沿って上, 下2方向に動き各節点を訪れる. 図1のような木の

* 通常 rooted directed tree という. ここでは単に 木 とよぶ.

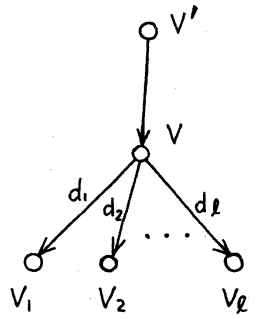

図1.

一部を考える．Pが節点Vをさしているとき操作(1)によってPはV'をさし，操作(2)によってPはV_iをさすようにそれぞれ枝に沿って移動する．P·up はPがルートをさしているとき NIL であり，Pが端点をさしているときP·down i は NIL であり，またPが図の節点Vをさしている場合，枝d_iが存在しないならP·down i はやはり NIL である．(1)(2)の操作で右辺が NIL ならポインタは移動させずℓ_Kへとびこす．各節点にはいくつかのデータがおかれていると考え，それらはフィールドセレクタsとポインタPをくみあわせてアクセスされる．たとえば，(3)の判定ではPのさす節点のセレクタsでえらばれるフィールドP·sと入力レジスタの内容に関するr(m)+1項述語P_mの真，偽によってℓ_j，ℓ_Kへそれぞれとびこす．いくつかのフィールドをまとめたものをブロックとよぶが，各節点はそれぞれ1つのブロックをあらわす．ブロックはデータフィールドの他に up，down 1，… などのリンクフィールドをもつと考えれば(1)，(2)はリンクのわりあて式とみてよい．ただし，データフィールドとリンクフィールドは本文では独立なものと考える．また，各フィールドは必らずしも有界な大きさとはかぎらないとする．これはたとえば任意に大きい数の

2進表現や任意に長い文字列などを1つのデータとして扱う場合に相当する．フィールドの有界性に依存して等価性が決定可能といえたところでその判定手続きは現実的なものでなくなるおそれがある．上のようにブロックを樹状に接続したデータ構造を樹状データ構造（以下単にデータ構造）とよびそのクラスを$\mathfrak{D}$とする．$\mathfrak{D}$のデータ構造は任意個の節点をもちうるがセレクタ名はプログラム固有である．(4)(5)はフィールド$P\cdot s'$と出力レジスタR_iへのわりあて式である．ここで，F_k，Q_mは単に関数，述語の記号であり，これらと各フィールドの定義域に適当な解釈を与えることによって，あるプログラム形から1つのプログラムが定まる．そのとき$\mathfrak{D}$の1つのデータ構造Dにも対応する解釈を与えプログラムにデータとして与えると1つの実行が定まる．以下では2つのプログラム形S_1，S_2が$\mathfrak{D}$のいかなるデータ構造に対しても，いかなる解釈に対しても〝同じふるまい〟をするとき強等価という．ただし，同じふるまいの意味は各モデル毎に定める．

ここで，プログラム形の簡単な例をあげる．

```
     l1:   P ← P·down 1 (l5);
START:     Q1(P·username, A1) l2, l1;
     l2:   P ← P·down 2 (l4);
           Q1(P·filename, A2) l3, l2;
     l3:   R1 ← F1(P·status);
           HALT(3);
     l4:   HALT(2);
     l5:   HALT(1);
```

ここで次の解釈を与えるとファイルディレクトリ中の A_1 で指定されたユーザの，A_2 で指定されたファイルのスティタス情報を R_1 へ返すようなプログラムとなる．

$Q_1(X, Y)$ ： $X = Y$

$F_1(X)$ ： $F_1(X) = X$

そのとき，

HALT(1)： 指定されたユーザは登録されていない．

HALT(2)： 指定されたファイルなし．

HALT(3)： スティタス情報が R_1 に入っている． ◁

以下の議論のため若干準備をする．

A1： プログラムはポインタをルートにおいて実行を開始し，つねにルートから上へ出るときにのみ停止するとする．*

A2： いくつかの証明は2方向木オートマトン 2ta，1方向木オートマトン 1ta に帰着させる．定義をのべる．

D1： 2ta は $M = (K, \Sigma, \delta, q_0, F_1, \cdots, F_k)$ で表わされるシステムである．ただし，

K： 空でない有限集合（状態）

Σ： 入力記号（入力木の節点のラベル）

δ： $K \times \Sigma \longrightarrow K \times \{U, D_1, \cdots, D_m, S\}$

q_0： K の1つの元（初期状態）

F_i： K の部分集合．ただし，$i \neq j$ なら $F_i \cap F_j = \phi$．

* ポインタがどこにあっても，$P \leftarrow P \cdot up$ をくりかえせば，つねにルートに達するのでこの仮定は一般性を失わない．

U, D_i は上方向，下方向への移動，Sは移動しないことをそれぞれ表わす．ポインタ（ヘッド）がルートを上方へ去るとき状態がF_iに入っていればMは与えられた木をクラスiと判定するといい，そのような木の集合を$L_i(M)$とかく．

D2： 1ta は $M=(K, \Sigma, \delta, q_0, F_1, \cdots, F_k)$ で表わされるシステムである．K, Σ, q_0, F_i's は 2ta と同様．

$$\delta: \underbrace{K \times K \times \cdots \times K}_{m} \times \Sigma \longrightarrow K$$

はじめ全端点にq_0をわりあて，次のように各節点にδを適用し状態を決めていく．節点Vの状態はVの直接下位の節点の状態が決まっていれば，その節点のラベルと合せてδによって決められる．mは各節点より出る枝数の最大値である．ルートにわりあてられた状態がF_iに属するならMは与えられた木をクラスiと判定するといい，そのような木の集合を$L_i(M)$とかく．

2方向有限オートマトン 2fa と1方向有限オートマトン 1fa は能力において等しいことがよく知られているが[6][5]，同様にして，$(\forall i)\ L_i(M)=L_i(N)$ のとき M と N が等価として，

P1;「任意の 2ta M に対して，Mに等価な 1ta を作りうる」*

A3： ポインタPは2方向に動くから同じ節点を2回以上訪れ，異なるフィールドについて判定することも，同じフィ

*「1ta は 後の P3 でのべる scan limited TM（どの cs の長さも一定値以下）でシミュレートでき，それは 1ta でシミュレートできる．(P4)」というよりつよいことがいえる．

ールドを異なる述語で判定することもある．いま，データフィールドは必らずしも有界なデータとは限らないとしているが，これを次のように各フィールドの領域を有界個の類に分割する．等価を判定すべき2つのプログラム形 S_1, S_2 中で用いられている判定の述語の全体をQとする．プログラムの実行中パラメータ $A_1, \cdots, A_m$ の値は変らないので述語 $Q_m(P\cdot s, A_{m1}, \cdots, A_{mr(m)})$ は実質的に $P\cdot s$ のみの単項述語とみなし，Qを単項述語のみと考える．各フィールドの内容は2つのプログラム形に関してはQの各述語を真とするか偽とするかということ以上の効果はない．そこで各フィールドの内容をQの各述語のとる値によって分類する記号（その全体をC）でおきかえて，各述語はこの有界個の記号についての判定でおきかえる．また，NILはそれぞれ新しい節点を追加し端記号#を記入する．このようにして(1)～(3)，(6)の命令のみを用いているプログラム形の動作は2dtでシミュレートできる．

R1：　木構造でなくたとえば2次元配列Aに対して，$P \leftarrow P\cdot right(l_i^*)$, $P \leftarrow P\cdot left(l_i)$, $P \leftarrow P\cdot up(l_i^*)$, $P \leftarrow P\cdot down(l_i)$ HALT(1), HALT(2) という命令のみを用いてかいたプログラム形をA上同じ点から動かしたとき，ともに止らないか，止るならともにHALT(1)でとまるかまたはHALT(2)でとまるというように等価を定義すると決定不能である．**

* この指定はなくてもよい．** 2カウンタ機械をシミュレートできることによる．

2 プログラム形のクラス1 $\mathcal{S}_1$

$\mathcal{S}_1$のプログラム形は(1)～(6)の文を用いてかかれる．このクラスでは(4)の出力文の出力フィールドと，(3)～(5)の文で参照されるフィールドは互いに素であると仮定する*．このクラスのプログラム形は次のように出力レジスタをもつ2方向木順序機械 2dsm** でシミュレートできる．参照されるフィールドはA3にのべたように有界個の分類記号にそれぞれおきかえ，出力文の実行は出力フィールドへ，出力の関数形を記号（列）としてかきこむものとする．レジスタ出力も同様．$\mathcal{S}_1$の2つのプログラム形S_1とS_2は解釈の仕方によらず，$\mathcal{D}$の各データ構造に対しルートから出発して一方がHALT(i)で停止すれば（停止条件はA1），他方もHALT(i)で停止し，そのときのデータ構造が同一でかつ各出力レジスタの内容が同一なら等価であるという．

P2:「$\mathcal{S}_1$のプログラム形の等価性は決定可能である．」

証明の概要．2faを1faへ変換する方法(5)(6)と類似の方法でS_1, S_2をシミュレートする2dsm M_1, M_2に対し出力のつじつまがあっているかどうかをしらべる1da $\overline{M}_1$, $\overline{M}_2$を構成し，その等価性の判定問題に帰着する．構成法は省略するが，各出力レジスタに対応する出力フィールドを各節点に追加し，レジスタへの出力はそれらフィールドへの出力でおきかえる．

* この条件をみたすかどうかは判定できる．** 完全順序機械csmに対応するもので，状態遷移の他に出力も指定されているような2da．

レジスタへの出力は木全体で最後の出力だけが有効ゆえ，レジスタに対応するフィールドは高々1つだけが空でないような木のみをえらび出し，その節点でのレジスタへの出力が最後のものであり，かつ，その内容が正しいことを確認するような $1ta$ を作りうる．これと同時に(4)の出力文による出力のつじつまがあうかどうかもしらべる．すなわち，$1ta\ M_j$ は次の条件をみたすようにつくる．$L_i(\bar{M}_j)$ が，各レジスタ R_ℓ に対応するフィールドには高々1カ所にだけその R_ℓ に対する最後の出力位置にその関数形がかかれており，その位置で $2ta\ M_j$ が R_ℓ に出力し，レジスタに対応するフィールドを除けば，M_j が状態 F_i でとまったとき残されたデータ構造であるようなものをちょうど含む．

R2：ここではもっとも単純なモデルについてのべたが，同じ手法で木の構造のある程度変化するようなプログラム形の等価性が決定できることが示せる．すなわち，節点毎に出力枝を置換すること，フィールドのかきかえ，1節点あるいは部分木の消去，各枝当り有界個の節点の挿入（ただし，挿入された節点のフィールドは参照せず出力のみ）などの操作を導入できる．

R3：S_1 では参照されるフィールドと出力フィールドを分離したが，かきこみよみとりを何回でも自由にできるとする

と等価性は決定不能となる.

3 プログラム形のクラス2 $\mathcal{S}_2$

$\mathcal{S}_1$ の条件をゆるめて，出力したフィールドの内容を参照してもよいとする．しかし，R3を考慮して，各ブロックをポインタPは高々有界回しかさせないとする．等価の定義は $\mathcal{S}_1$ と同じ．

P3:「$\mathcal{S}_2$ のプログラム形の等価性は決定可能である．」

この場合，出力したものを参照する可能性があるので，各フィールドは出力関数の合成関数についての判定結果についても分類する必要がある．このとき仮定から，関数は高々K回しかふかくならないのでこの分類はやはり有界である．その他は $\mathcal{S}_1$ と同様．木の各枝で crossing sequence (cs) を考えると，仮定からどのcsの長さも有界である．オフラインチューリング機械の場合と同様にして[4]，1taに帰着する．◁

R4: チューリング機械の1次元テープのかわりに樹状の〝テープ〟をもつ tree tape Turing machine (tTM) を考える．tTM Mが任意の木テープを与えられたとき，その節点数をnとするとき，上の意味でのcsの最大長が $R(n)$ でおさえられているとする．(3)の結果に対応して，

P4:「木テープの集合Tが tTM によって，$R(n)$-認識可能であって，

$$\lim_{n\to\infty} \frac{R(n)}{\log\log n} = 0$$

なら，T は lta で認識される．」

4 プログラム形のクラス3 $\mathcal{S}_3$

本文ではポインタは1つとしているが，データ構造上に目印としてマークをつけることを許し，次のような文を導入する．それ以外は $\mathcal{S}_1$ と同様．

(7) マークの移動　$P \leftarrow P\cdot up \text{ with } M_j\ (l_k)$;

(8)　〃　　$P \leftarrow P\cdot down^i \text{ with } M_j\ (l_k)$;

(9) マークの検知　D_{M_j}, l_i, l_k ;

(10) サブルーチンコール　CALL P_j ;

(11)　　RETURN(z) ; $z = 1, \cdots, K$.

(12) リターンパラメータの判定　GO TO $l_{i_1}, \cdots, l_{i_K}$;

各プログラム形は有界個のサブルーチン P_j $(j=1,\cdots,J)$ からなり，各 P_j は1つのマーク M_j をもち(7)(8)によりポインタとともにマークを移動することができる．(1)(2)を用いればマークをおいてポインタのみをうごかせる．マークの所在は(9)によってたしかめ，現在ポインタがさしている位置にマーク M_j があれば l_i へ，なければ l_k へそれぞれとびこす．サブルーチン P_j が(10)でコールされると，現在ポインタのさしている位置に新たにマーク M_j がおかれる．P_j の中ではこのマークを移動できる．P_j で RETURN を実行すると，

マーク M_j は消滅し，ポインタはマーク M_{j-1} の位置にもどる．このときのパラメータは，(12)の GOTO 文の行先を定める．RETURN を実行するとプログラムの制御は CALL 文の次の文へうつる．P_j 内ではポインタは M_j より上へ出ることはないと仮定する．等価性は $\mathcal{S}_1$ と同様に定義する．

P5:「$\mathcal{S}_3$ のプログラム形の等価性は決定可能である」

これはマーク付 ta の問題に帰着する．

R5: 1次元テープの場合，1マークの2方向有限オートマトンの能力は $1fa$ と同じであることが知られている．[1][5] 同様に1マークの $2ta$ の能力は $1ta$ とかわらないことが示せる．2つ以上のマークをもつ $2fa$ は $1fa$ の能力をこえ，等価性も決定不能となる．これは2つのポインタ（ヘッド）をもつ有限オートマトンをシミュレートできるからである．

しかし，複数個のマーク $M_1, \cdots, M_J$ を許しても，マーク M_j が動くときは必ず $M_{j+1}, \cdots, M_J$ も同じコマにあって，かつ一緒にすべて次のコマへ移るというようにマークの動き方に制限をつけた $2fa$ を考えると，このクラスのオートマトンの能力は $1fa$ と変らないことが示せる．上の $\mathcal{S}_3$ はこのオートマトンに対応したものではあるが，上述のポインタに対する制限が付け加えられている．

文 献

[1] Blum, M. and Hewitt, C. Automata on a 2-dimensional tape. Proc. 8th Ann. Symp. on Switching and Automata Theory, 155-160. 1967.

[2] Doner, J. Tree Acceptors and Some of Their Applications. JCSS 4 406-451. 1970.

[3] Hartmanis, J. Computational Complexity of One-Tape Turing Machine Computations. JACM 15 (2) 325-339, 1968.

[4] Hennie, F.C. One-Tape, Off-Line Turing Machine Computation. Inf. and Contr. 8 553-578. 1965.

[5] 池野信一, あるオートマトンの性質. 信学会オートマトン研資 1967-11.

[6] Shepherdson, J. C. The reduction of two-way automata to one-way automata. IBM J. 3 198-200 1959.

[7] Thatcher, J. W. and Wright, J. B. Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic. Math. Syst. Theory 2 57-81 1968.