

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けられる.

- (1) ポインタ操作 $P \leftarrow P \cdot \text{up} (l_k);$
- (2) " $P \leftarrow P \cdot \text{down } i (l_k);$
- (3) 判定 $Q_m(P \cdot s, A_{m1}, \dots, A_{mr(m)}, l_j, l_k;$
- (4) 出力 $P \cdot s' \leftarrow F_t(P \cdot s_{t1}, \dots, P \cdot s_{tr'(t)}, A_{tr'(t)+1}, \dots, A_{tr(t)});$
- (5) レジスタ出力 $R_i \leftarrow F_t(P \cdot s_{t1}, \dots, P \cdot s_{tr'(t)}, A_{tr'(t)+1}, \dots, A_{tr(t)});$
- (6) 停止 $\text{HALT}(z); \quad z = 1, 2, \dots, K.$

ここで, l_j, l_k 等はプログラムラベルである. $A_1, \dots, A_n$ は入力レジスタでこれらへの出力は許されない. $R_1, \dots, R_m$ 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.

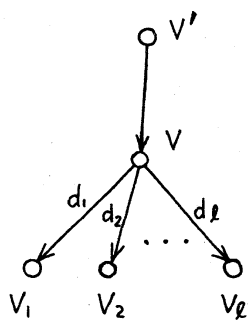


図 1.

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. $P \cdot up$ 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 $P \cdot down i$ 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 $P \cdot down i$ 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トア 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トア s でえらばれるフィールド $P \cdot s$ 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. また, 各フィールドは必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とおそれがある。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 $\mathcal{D}$ とする。 $\mathcal{D}$ のデータ構造は任意個の節点をもちうるがセレクト名はプログラム固有である。(4)(5)はフィールド $P \cdot s'$ とホカレジスタ R_i へのわりあて式である。ここで、 F_t , 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 $\mathcal{D}$ 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 $\mathcal{D}$ 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。

```

 $l_1$ :  $P \leftarrow P \cdot \text{down } 1 \ (l_5)$ ;
START:  $Q_1(P \cdot \text{username}, A_1) \ l_2, l_1$ ;
 $l_2$ :  $P \leftarrow P \cdot \text{down } 2 \ (l_4)$ ;
 $Q_1(P \cdot \text{filename}, A_2) \ l_3, l_2$ ;
 $l_3$ :  $R_1 \leftarrow F_1(P \cdot \text{status})$ ;
      HALT(3);
 $l_4$ : HALT(2);
 $l_5$ : 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ータス情報を 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): ステータス情報が R_1 に入っている. $\triangleleft$

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 $M = (K, \Sigma, \delta, q_0, F_1, \dots, F_k)$ 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)

$\delta: K \times \Sigma \rightarrow K \times \{U, D_1, \dots, D_m, S\}$

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 $F_i \cap F_j = \emptyset$.

* 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) がルートを上方向へ去るとき状態が F_i に入っているのは M は与えられた木をクラス i と判定するといひ, そのような木の集合を $L_i(M)$ とかく.

D2: 1ta は $M = (K, \Sigma, \delta, q_0, F_1, \dots, F_k)$ 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.

$$\delta: \underbrace{K \times K \times \dots \times K}_m \times \Sigma \rightarrow K$$

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 と判定するといひ, 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⁽⁶⁾⁽⁵⁾ は能力において等しいことがよく知られているが, 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, 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)」というよりよいことがいえる.

フィ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 $A_1, \dots, A_m$ の値は変わらないので述語 $Q_m(p.s, A_{m1}, \dots, A_{mr(m)})$ は実質的に $P.s$ 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 の各述語のとり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2人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、
 $P \leftarrow P \cdot \text{right}(li)^*$, $P \leftarrow P \cdot \text{left}(li)$, $P \leftarrow P \cdot \text{up}(li)^*$, $P \leftarrow P \cdot \text{down}(li)$
 $\text{HALT}(1)$, $\text{HALT}(2)$ 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 $\text{HALT}(1)$ でとまるかまたは $\text{HALT}(2)$ 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 フログラム形のクラス 1 $\mathcal{S}_1$

$\mathcal{S}_1$ のフログラム形は (1) ~ (6) の文を用いてかかる。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*。このクラスのフログラム形は次のように出力レジスタをもつ 2 方向本順序機械 $2tsm^{**}$ 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 つのフログラム形 S_1 と S_2 は解釈の仕方によらず、 $\mathcal{S}$ 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: 「 $\mathcal{S}_1$ のフ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 $2tsm$ M_1, M_2 に対し出力のつじつまが揃っているかどうかをしらべる $1ta$ $\bar{M}_1, \bar{M}_2$ 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。

* この条件を満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 $2ta$ 。

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、 $1ta \bar{M}_j$ 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 $2ta \bar{M}_j$ 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、 $\bar{M}_j$ が状態 F_i でとま、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、節点毎に出力板と置換すること、フィールドのかきかえ、1節点あるいは部分木の消去、各板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: §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、かきこみととりを何回でも自由にできるとする

と等価性は決定可能となる。

3 フロگرام形のクラス 2 δ_2

δ_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、各ブロックとポインタ P は高々有界回しかささないとする。等価の定義は δ_1 と同じ。

P3: 「 δ_2 のフロگرام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 回しかふかくなならないのでこの分類はやはり有界である。その他は δ_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⁽⁴⁾して、 $1/a$ に帰着する。◀

R4: チューリング機械の 1 次エ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 $R(n)$ であるといえるとする。(3)の結果に対して、

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、

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

なら, T は lta で認識される. $\square$

4 フログラム形のクラス $\mathcal{R}_3$

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 $\mathcal{R}_1$ と同様.

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

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

(9) マークの検知 $D_{M_j}, l_i, l_k;$

(10) サブルーチンコール $\text{CALL } P_j;$

(11) $\text{RETURN}(z); z = 1, \dots, K.$

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

各フログラム形は有界個のサブルーチン P_j ($j=1, \dots, J$) 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. (11)(12) 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して仮定する。等価性は $\equiv_1$ と同様に定義する。

P5: 「 $\equiv_3$ 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 $\mathcal{A}$ 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 2 $\mathcal{A}$ の能力は $1\mathcal{A}$ 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 $M_1, \dots, M_T$ を許しても、マーク M_j が動くときは必ず $M_{j+1}, \dots, M_T$ も同じコマにあって、かっ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 $1fa$ と変わらないことが示せる。上の $\equiv_3$ はこのオートマトンに対応したものであるが、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究 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.