カウンター機械 (レジスター機械)
Counter Machine (Register Machine)

y.*

2018 年 6 月 6 日

最終更新日: 2018 年 6 月 6 日

概要

レジスター機械 (register machine) とは，自然数を格納できる無限個のレジスターに対し単純な命令の列 (プログラム) を順次実行して計算を進める計算モデルである．レジスター機械の中でも**カウンター機械** (counter machine) は最も単純なモデルであり，レジスターに 1 を足す・1 を引くという機能 (と条件分岐) しか持たない．しかしながら，カウンター機械は Turing 機械をシミュレートできるほどに強力な計算能力を持つことが知られている．本稿ではカウンター機械が Turing 機械と同等の計算能力を持つことを証明する．

Keywords: カウンター機械 (counter machine), レジスター機械 (register machine), プログラム機械 (program machine), Minsky 機械 (Minsky machine).

1　カウンター機械

カウンター機械 (counter machine) は以下のものから構成される．

- 自然数を格納できる可算無限個の**レジスター** (register) $\mathcal{R}_0, \mathcal{R}_1, \ldots$．それぞれのレジスター $\mathcal{R}_i$ は計算の各時点で一つの自然数 r_i を保持している．
- **プログラム** (program) を格納しておくプログラム格納所．プログラムは後述する**命令** (instruction) の有限列であり，プログラム中のそれぞれの命令には 0 から始まる行番号が振られている．プログラムは計算したい部分関数ごとに入れ換えられる．
- プログラム中の現在実行している行番号を保持する**プログラムカウンター** (program counter) (あるいは**インストラクションポインター** (instruction pointer) とも呼ぶ)．

カウンター機械の概念図を図 1 に示す．

* http://iso.2022.jp/

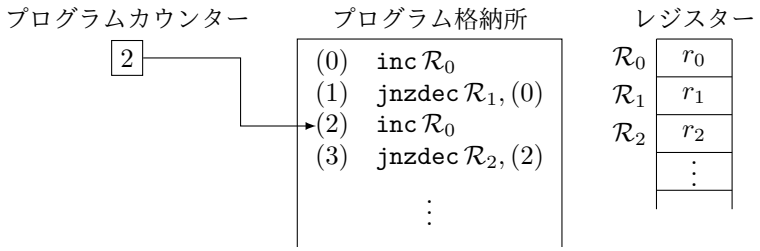

図 1　カウンター機械の構成要素

ここではカウンター機械のプログラムで使用可能な命令として以下に示す inc, jnzdec の 2 つのみを用いる．(この定義は新井 [3] を参考にした．)

- $\texttt{inc}\,\mathcal{R}_i$: [*1]$\mathcal{R}_i$ に保持されている自然数 r_i の値を 1 だけ増加させ，プログラムカウンターの値を 1 だけ増加させる．
- $\texttt{jnzdec}\,\mathcal{R}_i, (n)$: [*2]$r_i = 0$ ならばプログラムカウンターの値を n にする．$r_i \neq 0$ ならば r_i の値を 1 だけ減少させ，プログラムカウンターの値を 1 だけ増加させる．

このバージョンのカウンター機械は**プログラム機械** (program machine) とも呼ばれる．

プログラムに使用可能な命令セットを変更することによりカウンター機械の様々なバリエーションを作ることができる．計算可能性の概念が**頑健性** (robustness) を持つことから，命令セットの多少の変更では計算能力は変化しない．このことは 4 節でも見る．

カウンター機械 (プログラム機械) はレジスター機械の中でも最も単純な命令セットを持つので，例えば現在の CPU にあるような間接アドレス指定の機能はない．つまり，“$\texttt{inc}\,\mathcal{R}_{r_i}$” のような命令は使用できない．[*3]

N 行からなるカウンター機械のプログラム P に対し，次のようにして計算される n 変数部分関数 $F_n^P(x_1, \ldots, x_n)$ が定まる．

1. レジスター $\mathcal{R}_1, \ldots, \mathcal{R}_n$ にそれぞれ $x_1, \ldots, x_n$ を格納する．
2. プログラムカウンターの値を 0 に設定する．
3. プログラムカウンターに保持されている行番号の命令を実行する．これをプログラムカウンターの値がプログラムの行数 N 以上になるまで繰り返す．プログラムカウンターの値が N 以上になった時点で計算を終了する．
4. 計算終了時点で $\mathcal{R}_0$ に格納されている値 r_0 を出力値 $F_n^P(x_1, \ldots, x_n)$ とする．

以下にカウンター機械の簡単なプログラムの例を示す．

例 1.1.　次のプログラム P を考える．

$$\begin{array}{ll} (0) & \texttt{inc}\,\mathcal{R}_0 \\ (1) & \texttt{inc}\,\mathcal{R}_0 \\ (2) & \texttt{jnzdec}\,\mathcal{R}_1, (0) \\ (3) & \texttt{inc}\,\mathcal{R}_0 \\ (4) & \texttt{jnzdec}\,\mathcal{R}_2, (3) \end{array} \tag{1}$$

[*1] inc は increment の略．
[*2] jnz は jump if not zero の略，dec は decrement の略．
[*3] もちろん，間接アドレス指定の機能を持たせても計算能力が向上するわけではなく，Turing 機械と等価なままである．

P は関数 $F_2^P(x_1, x_2) = 2(x_1+1)+(x_2+1) = 2x_1+x_2+3$ を計算するカウンター機械のプログラムである．実際，P を $r_1 = 2, r_2 = 1$ として実行すると表 1 のように動作する．

表 1　入力が $r_1 = 2, r_2 = 1$ のときの P の計算の履歴

行番号	0	1	2	0	1	2	0	1	2	3	4	3	4	5
$\mathcal{R}_0$	0	1	2	2	3	4	4	5	6	6	7	7	8	8
$\mathcal{R}_1$	2	2	2	1	1	1	0	1	1	0	0	0	0	0
$\mathcal{R}_2$	1	1	1	1	1	1	1	1	1	1	1	0	1	0

これ以降，前に作ったプログラムを新しいプログラムの中に埋め込んで再利用するということを多く行う．そのような状況で `jnzdec` で行番号を正確に指定するのは煩わしいので，以降は不要な行番号は省略し，また必要に応じて行番号の代わりに $L, L_0, L_1, \ldots$ などのラベルを用いることにする．この約束のもとでは，例えばプログラム (1) は

$$\begin{array}{ll} (L_0) & \texttt{inc}\,\mathcal{R}_0 \\ & \texttt{inc}\,\mathcal{R}_0 \\ & \texttt{jnzdec}\,\mathcal{R}_1, (L_0) \\ (L_1) & \texttt{inc}\,\mathcal{R}_0 \\ & \texttt{jnzdec}\,\mathcal{R}_2, (L_1) \end{array}$$

のように書き直せる．

最後に，プログラムを書くときに便利な略記法をいくつか定義しておく．

補題 1.2.　以下のような動作をするプログラムの断片を書くことができる．

- goto (n): プログラムカウンターの値を n にする．
- clear $\mathcal{R}_i$: r_i を 0 にする．
- move $\mathcal{R}_i$ to $\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_n}$: $r_{j_1} = \cdots = r_{j_n} = 0$ のとき，r_i の値を $r_{j_1}, \ldots, r_{j_n}$ に移し，r_i を 0 にする．
- copy $\mathcal{R}_i$ to $\mathcal{R}_j$ using $\mathcal{R}_u$: $r_j = r_u = 0$ のとき，r_i の値を r_j にコピーし，r_i と $r_u = 0$ の値はそのまま保持する．
- nop: 何もせず，プログラムカウンターの値だけを増加させる．[*4]

証明. 次のようにすればよい．

$$\text{goto}\,(n) = \left[\begin{array}{ll} & \texttt{inc}\,\mathcal{R}_0 \\ & \texttt{jnzdec}\,\mathcal{R}_0, (n) \end{array}\right],$$

$$\text{clear}\ \mathcal{R}_i = \left[\begin{array}{lll} (L) & \texttt{jnzdec}\,\mathcal{R}_j, (L) \end{array}\right],$$

$$\text{move}\ \mathcal{R}_i \text{ to } \mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_n} = \left[\begin{array}{ll} (L_0) & \texttt{jnzdec}\,\mathcal{R}_i, (L_1) \\ & \text{goto}\,(L_2) \\ (L_1) & \texttt{inc}\,\mathcal{R}_{j_1} \\ & \cdots \\ & \texttt{inc}\,\mathcal{R}_{j_n} \\ & \text{goto}\,(L_0) \\ (L_2) & \end{array}\right],$$

$$\text{copy}\ \mathcal{R}_i \text{ to } \mathcal{R}_j \text{ using } \mathcal{R}_u = \left[\begin{array}{ll} & \text{move}\ \mathcal{R}_i \text{ to } \mathcal{R}_j, \mathcal{R}_u \\ & \text{move}\ \mathcal{R}_u \text{ to } \mathcal{R}_i \end{array}\right],$$

[*4] nop は no operation の略.

$$\mathrm{nop} = \left[\begin{array}{l} \\ (L) \end{array}\begin{array}{l} \texttt{inc}\,\mathcal{R}_0 \\ \texttt{jnzdec}\,\mathcal{R}_0, (L) \\ \end{array}\right]. \qquad \square$$

2 再帰的関数はカウンター機械で計算可能

本節では，Turing 機械で計算可能な任意の部分関数がカウンター機械によっても計算可能であることを示す．ただし，直接 Turing 機械をシミュレートするのではなく，再帰的関数がカウンター機械で計算可能であることを示す．再帰的関数の定義については「再帰的関数」[2] を参照のこと．

定理 2.1. 任意の再帰的関数 $f\colon \mathbb{N}^n \rightharpoonup \mathbb{N}$ はカウンター機械によって計算可能である．

証明. より強く，次のことを再帰的関数の定義に沿った帰納法で証明する．

> 任意の再帰的関数 $f\colon \mathbb{N}^n \rightharpoonup \mathbb{N}$ に対して次が成り立つ: 「互いに相異なる $n+2$ 個の自然数 $i_1, \ldots, i_n, j, u \in \mathbb{N}$ (ただし $\max\{i_1, \ldots, i_n, j\} < u$) に対して，次のようなカウンター機械のプログラム $P = P(f; i_1, \ldots, i_n; j; u)$ を作ることができる: 「$f(r_{i_1}, \ldots, r_{i_n})\!\downarrow = y$ のとき，$r_j, r_u, r_{u+1}, \ldots$ が全て 0 の状態で P を実行すると，$r_j = y$ となり，その他のレジスターの値は変化させない (ただし，$f(r_{i_1}, \ldots, r_{i_n})\!\uparrow$ のときは P も停止しない).」」

つまり，P は f を計算するときの途中計算を $\mathcal{R}_u, \mathcal{R}_{u+1}, \ldots$ にメモしておいて，計算が終わったら途中計算を全て消去するようなプログラムである．

$f = \texttt{zero}$ のとき $P(\texttt{zero}; ; j; u)$ を次のように定義すればよい．

$$\mathrm{nop}$$

$f = \texttt{proj}^n_k$ のとき $P(\texttt{proj}^n_k; i_1, \ldots, i_n; j; u)$ を次のように定義すればよい．

$$\text{copy } \mathcal{R}_{i_k} \text{ to } \mathcal{R}_j \text{ using } \mathcal{R}_u$$

$f = \texttt{succ}$ のとき $P(\texttt{succ}; i; j; u)$ を次のように定義すればよい．

$$\begin{array}{l} \text{copy } \mathcal{R}_i \text{ to } \mathcal{R}_j \text{ using } \mathcal{R}_u \\ \texttt{inc}\,\mathcal{R}_j \end{array}$$

$f = g \circ (h_1, \ldots, h_m)$ のとき 帰納法の仮定から $g, h_1, \ldots, h_m$ を計算するプログラム $P(g; i_1, \ldots, i_m; j; u)$ と $P(h_1; i_1, \ldots, i_n; j; u), \ldots, P(h_m; i_1, \ldots, i_n; j; u)$ がある．このとき $P(f; i_1, \ldots, i_n; j; u)$ を次のように定義すればよい．

$$\begin{array}{l} P(h_1; i_1, \ldots, i_n; u; u+m) \\ P(h_2; i_1, \ldots, i_n; u+1; u+m) \\ \ldots \\ P(h_m; i_1, \ldots, i_n; u+m-1; u+m) \\ P(g; u, u+1, \ldots, u+m-1; j; u+m) \\ \text{clear } \mathcal{R}_u \\ \ldots \\ \text{clear } \mathcal{R}_{u+m-1} \end{array}$$

$f(0, \vec{y}) = g(\vec{y}), f(x+1, \vec{y}) = h(x, \vec{y}, f(x, \vec{y}))$ のとき 帰納法の仮定から g, h を計算するプログラム $P(g; i_1, \ldots, i_n; j; u), P(h; i_0, \ldots, i_{n+1}; j; u)$ がある．このとき $P(f; i_0, \ldots, i_n; j; u)$ を次のように定義

すればよい．

$$
\begin{array}{ll}
 & \text{copy } \mathcal{R}_{i_0} \text{ to } \mathcal{R}_u \text{ using } \mathcal{R}_{u+3} \\
 & P(g; i_1, \ldots, i_n; j; u+3) \\
(L_0) & \texttt{jnzdec}\, \mathcal{R}_u, (L_1) \\
 & \text{goto}\, (L_2) \\
(L_1) & P(h; u+1, i_1, \ldots, i_n, j; u+2; u+3) \\
 & \text{copy } \mathcal{R}_{u+2} \text{ to } \mathcal{R}_j \text{ using } \mathcal{R}_{u+3} \\
 & \texttt{inc}\, \mathcal{R}_{u+1} \\
 & \text{goto}\, (L_0) \\
(L_2) & \text{clear } \mathcal{R}_u \\
 & \text{clear } \mathcal{R}_{u+1} \\
 & \text{clear } \mathcal{R}_{u+2}
\end{array}
$$

$f(\vec{x}) = \mu y[g(\vec{x}, y) = 0]$ のとき　帰納法の仮定から g を計算するプログラム $P(g; i_1, \ldots, i_{n+1}; j; u)$ がある．

このとき $P(f; i_1, \ldots, i_n; j; u)$ を次のように定義すればよい．

$$
\begin{array}{ll}
 & \text{goto}\, (L_1) \\
(L_0) & \texttt{inc}\, \mathcal{R}_j \\
(L_1) & P(g; i_1, \ldots, i_n, j; u; u+1) \\
 & \texttt{jnzdec}\, \mathcal{R}_u, (L_0) \\
 & \text{clear } \mathcal{R}_u
\end{array}
$$

以上より，$f\colon \mathbb{N}^n \rightharpoonup \mathbb{N}$ はプログラム $P(f; 1, \ldots, n; 0; n+1)$ によって計算される．　□

3　カウンター機械で計算可能ならば Turing 機械で計算可能

本節では前節の逆，つまりカウンター機械で計算可能な部分関数は Turing 機械によっても計算可能であることを示す．Church-Turing の提唱から Turing 機械で計算可能であると言ってもよいのだが，念の為証明の概略を与えておく．

定理 3.1.　カウンター機械で計算可能な任意の部分関数 $f\colon \mathbb{N}^n \to \mathbb{N}$ は Turing 機械で計算可能である．

証明の概略．f を計算するカウンター機械のプログラムを P とする．P は命令の有限の長さのリストなので，P で言及されるレジスターの個数も有限個である．そこで $k := \max\{\, i \mid \mathcal{R}_i \text{ が } P \text{ に現れる} \,\} \cup \{n\}$ とおく．P の動作を k テープ Turing 機械 M で模倣すればよい．[*5]つまり，一つのレジスターにつき一本のテープを用いる．

M はまず，1 本目のテープ ($\mathcal{R}_0$) に含まれる 1 を $\mathcal{R}_1, \ldots, \mathcal{R}_n$ に振り分ける:

$$
\begin{array}{ll}
\mathcal{R}_0 & 1^{x_1+1}\text{␣}1^{x_2+1}\text{␣}\cdots\text{␣}1^{x_n+1}\text{␣}\cdots \\
\mathcal{R}_1 & \text{␣␣}\cdots \\
\cdots & \\
\mathcal{R}_n & \text{␣␣}\cdots \\
\mathcal{R}_{n+1} & \text{␣␣}\cdots \\
\cdots & \\
\mathcal{R}_k & \text{␣␣}\cdots
\end{array}
\quad\longrightarrow\quad
\begin{array}{ll}
\mathcal{R}_0 & \text{␣␣}\cdots \\
\mathcal{R}_1 & 1^{x_1}\text{␣}\cdots \\
\cdots & \\
\mathcal{R}_n & 1^{x_n}\text{␣}\cdots \\
\mathcal{R}_{n+1} & \text{␣␣}\cdots \\
\cdots & \\
\mathcal{R}_k & \text{␣␣}\cdots .
\end{array}
$$

[*5] ここで使用する Turing 機械の定義は基本的には「Turing 機械の変種」[1] に沿うものとする．ただし，k テープ Turing 機械が計算する n 変数部分関数 $f(x_1, \ldots, x_n)$ は次のように定める: 1 本目のテープの内容が $1^{x_1+1}\text{␣}1^{x_2+1}\text{␣}\cdots\text{␣}1^{x_n+1}\text{␣}\cdots$，他の全てのテープは空白記号␣で埋まった状態で計算を始め，状態が q_{halt} になったら全てのテープに含まれる空白記号␣以外の記号の個数を出力値とする．

その後，P の命令に従ってテープ上の 1 の個数を増減させる．このとき，カウンター機械のプログラムカウンターの値 (P の行数以下) は M の状態として保持しておく．

最後に，P の動作のシミュレーションが終了した後，M は $\mathcal{R}_0$ 以外の全てのテープの内容を消去してから停止する． □

4 カウンター機械の変種

本節では，カウンター機械の変種の一つとして **Minsky 機械** (Minsky machine) を取り上げる．Minsky 機械は各命令の実行後にプログラムカウンターを自由に設定できる点以外はプログラム機械と同じである．

定義 4.1 (Minsky 機械). 以下の命令セットを持つカウンター機械を Minsky 機械と呼ぶ．

- $\mathtt{inc}\,\mathcal{R}_i, (n)$: $\mathcal{R}_i$ に保持されている自然数 r_i の値を 1 だけ増加させ，プログラムカウンターの値を n にする．
- $\mathtt{jnzdec}\,\mathcal{R}_i, (m), (n)$: $r_i \neq 0$ ならば r_i の値を 1 だけ減少させ，プログラムカウンターの値を m にする．$r_i = 0$ ならばプログラムカウンターの値を n にする．

補題 4.2. プログラム機械と Minsky 機械は等価である．

証明. Minsky 機械をプログラム機械でシミュレートするには

$$\mathtt{inc}\,\mathcal{R}_i, (L) = \begin{bmatrix} \mathtt{inc}\,\mathcal{R}_i \\ \mathrm{goto}\,(L) \end{bmatrix},$$
$$\mathtt{jnzdec}\,\mathcal{R}_i, (L_0), (L_1) = \begin{bmatrix} \mathtt{jnzdec}\,\mathcal{R}_i, (L_0) \\ \mathrm{goto}\,(L_1) \end{bmatrix}$$

とすればよいし，逆にプログラム機械で Minsky 機械をシミュレートするには

$$\mathtt{inc}\,\mathcal{R}_i = \begin{bmatrix} & \mathtt{inc}\,\mathcal{R}_i, (L) \\ (L) & \end{bmatrix},$$
$$\mathtt{jnzdec}\,\mathcal{R}_i, (L) = \begin{bmatrix} & \mathtt{jnzdec}\,\mathcal{R}_i, (L), (L_0) \\ (L_0) & \end{bmatrix}$$

とすればよい． □

参考文献

[1] y., Turing 機械の変種 (2018), http://iso.2022.jp/math/undecidable-problems/files/variants-of-turing-machine.pdf.

[2] y., 再帰的関数 (2018), http://iso.2022.jp/math/undecidable-problems/files/recursive-function.pdf.

[3] 新井敏康, 数学基礎論, 岩波書店, 2011.

変更履歴

2018/06/06　公開