1J-04

可逆プログラミング言語R-WHILEによる万能可逆チューリング機械の構成

青木　峻 † 柴田 心太郎 † 横山 哲郎 ††
† 南山大学情報理工学部ソフトウェア工学科　†† 南山大学理工学部ソフトウェア工学科

1　はじめに

計算できるという概念をチューリング機械（TM）で計算できるということにしようという主張は広く認められている．TMは計算の効率を問題としなければ現在のコンピュータをも模倣できるとされている強力な計算モデルであり，計算可能性理論を議論する際に重要である．任意のTMをシュミレートできる計算システムは計算万能性をもつといわれる．プログラミング言語の計算モデルが計算万能性をもつことを示すことは重要である．

可逆計算とは，計算過程において，どの状態においても直前と直後のにとり得る状態を高々1つもつものである．可逆チューリング機械（RTM）が計算できるのは単射な計算可能関数であることが知られている[1]．可逆プログラミング言語とは，そのプログラムの実行過程が必ず可逆になるような言語設計がなされているプログラミング言語である．可逆プログラミング言語が可逆チューリング言語を模倣できること，すなわちその計算モデルが可逆的計算万能性をもつことを示すことは重要である．

可逆プログラミング言語R-WHILEは，Jonesの言語WHILE[2]を可逆化したものであり，任意の命令Cに対して逆命令$\mathcal{I}[\![\mathtt{C}]\!]$があるという特徴をもち，木構造のデータをもつ．筆者の知る範囲では，R-WHILEが可逆的計算万能性をもつという報告はない．

2　可逆チューリング機械

本章では，TMとTMに制限を加えたRTMの定義を述べる．本稿では，簡単のため1テープのTMのみを考える．また，文献[3]の表記を用いる．

2.1　チューリング機械

TMを$T=(Q,\Sigma,b,\delta,q_s,q_f)$として定める．ただし$Q$は内部状態の空でない有限集合，$\Sigma$はテープ記号の空でない有限集合，$b\,(\in S)$は空白記号，$\delta$は$(Q\times[\Sigma\times\Sigma]\times Q)\cup(Q\times\{\leftarrow,\downarrow,\rightarrow\}\times Q)$の部分集合，$q_s\,(\in Q)$は初期状態，$q_f\,(\in Q)$は最終状態とする．

δは遷移規則の集合である．矢印（$\leftarrow,\downarrow,\rightarrow$）はそれぞれヘッドの移動（左，不動，右）を表す．遷移規則は3項組であり$(q,(s,s'),q')$または(q,d,q')である$(q,q'\in Q,s,s'\in\Sigma,d\in\{\leftarrow,\downarrow,\rightarrow\})$．前者の3項組は$T$が状態$q$で記号$s$を読んだ場合，記号$s'$に書き換え，状態を$q'$にすることを意味する．また，後者の3項組は$T$が状態$q$の場合ヘッドを$d$の方向に動かし，状態を$q'$にすることを意味する．

TM　T　$=$　$(Q,\Sigma,b,\delta,q_s,q_f)$　の様相とは，組$(q,(l,s,r))\in Q\times((\Sigma\backslash\{b\})^*\times\Sigma\times(\Sigma\backslash\{b\})^*)$である．ここで$V^*$は$V$中の記号を0個以上並べたものを表す．ただし，$q$は内部状態，$s$はヘッドの下にある記号，$l$はヘッドの左側のテープを表す記号列，$r$はヘッドの右側のテープを表す記号列を表す．

TM $T=(Q,\Sigma,b,\delta,q_s,q_f)$の計算ステップは，$c\underset{T}{\Rrightarrow}c'$を満たすように様相$c$を様相$c'$に移す．ただし，ここで，$\lambda$を空語として

$$
\begin{aligned}
&(q,(l,s,r)) \underset{T}{\Rrightarrow} (q',(l,s',r)) && \text{if} \quad (q,(s,s'),q')\in\delta\\
&(q,(\lambda,s,r)) \underset{T}{\Rrightarrow} (q',(\lambda,b,sr)) && \text{if} \quad (q,\leftarrow,q')\in\delta\\
&(q,(ls',s,r)) \underset{T}{\Rrightarrow} (q',(l,s',sr)) && \text{if} \quad (q,\leftarrow,q')\in\delta\\
&(q,(ls,b,\lambda)) \underset{T}{\Rrightarrow} (q',(l,s,\lambda)) && \text{if} \quad (q,\leftarrow,q')\in\delta\\
&(q,(l,s,r)) \underset{T}{\Rrightarrow} (q',(l,s,r)) && \text{if} \quad (q,\downarrow,q')\in\delta\\
&(q,(l,s,\lambda)) \underset{T}{\Rrightarrow} (q',(ls,b,\lambda)) && \text{if} \quad (q,\rightarrow,q')\in\delta\\
&(q,(l,s,s'r)) \underset{T}{\Rrightarrow} (q',(ls,s',r)) && \text{if} \quad (q,\rightarrow,q')\in\delta\\
&(q,(\lambda,b,sr)) \underset{T}{\Rrightarrow} (q',(\lambda,s,r)) && \text{if} \quad (q,\rightarrow,q')\in\delta
\end{aligned}
$$

である．$\underset{T}{\Rrightarrow}$の反射推移閉包を$\underset{T}{\Rrightarrow}^*$と記す．

TM $T=(Q,\Sigma,b,\delta,q_s,q_f)$の意味を$[\![T]\!]=\{(r,r')\mid(q_0,(\lambda,b,r))\underset{T}{\Rrightarrow}^*(q_f,(\lambda,b,r'))\}$とする．

2.2　可逆チューリング機械

TM　T　は任意の異なる遷移規則$(q_1,a_1,q_1'),(q_2,a_2,q_2')\in\delta$に対して，$q_1=q_2$ならば$a_1=(s_1,s_1')$, $a_2=(s_2,s_2')$およびに$s_1\neq s_2$であるならば局所的に前方決定的であるという．またTM Tは任意の異なる遷移規則$(q_1,a_1,q_1'),(q_2,a_2,q_2')\in\delta$に対して$q_1'=q_2'$ならば$a_1=(s_1,s_1')$, $a_2=(s_2,s_2')$および$s_1'\neq s_2'$であるならば局所的に後方決定的であ

A universal reversible Turing machine program in reversible programming language R-WHILE
† Ryo AOKI
† Shintaro SHIBATA
†† Tetsuo YOKOYAMA
Department of Software Engineering, Faculty of Information Sciences and Engineering, Nanzan University (†)
Department of Software Engineering, Faculty of Science and Engineering, Nanzan University (††)

```
read R;
  Q ^= q̄_s;
  T <= (nil b̄ R);
  from (=? Q q̄_s) loop
    STEP(Q,T)
  until (=? Q q̄_f);
  (nil b̄ R') <= T;
  Q ^= q̄_f;
write R'
```

(a) main プログラム

```
macro STEP(Q,T) ≡
rewrite [Q,T] by
 T⟦t⟧*
```

(b) マクロ STEP

```
macro MOVEL(T) ≡
(L S R) <= T;
PUSH(S,R);
POP(S,L);
T <= (L S R)
```

(c) マクロ MOVEL

```
macro PUSH(S,STK) ≡
rewrite [S,STK] by
 [b̄,nil] => [nil,nil]
 [S,STK] => [nil,(S.STK)]
```

(d) マクロ PUSH

図 1: RTM を模倣する R-WHILE プログラム

るという．

TM $T = (Q, \Sigma, b, \delta, q_s, q_f)$ は，局所的に前方決定的かつ後方決定的であり，最終状態からの遷移および初期状態への遷移がないとき，可逆と呼ぶ．

3 RTM から R-WHILE への変換

図 1a に TM プログラムからの変換で得られた R-WHILE プログラムを示す．組 (Q,T) が様相を表す．ループでは，初期様相が最終様相に遷移するまでマクロ STEP(Q,T) を繰り返し実行する．

図 1b で定義されるマクロ STEP(Q,T) では，様相 (Q,T) を書き換え規則により書き換える．$\mathcal{T}[\![t]\!]^*$ は生成された書き換え規則の列である．

それぞれの書き換え規則は，図 2の変換器 $\mathcal{T}$ によって遷移規則 t $(\in \delta)$ から生成される．$\bar{q}$ は，状態 q に対応する R-WHILE のアトムである．RTM の遷移規則列を変換した場合，異なる書き換え規則は矢印=>の左側のパターンと右側で返却される値がそれぞれ重なることはない．

図 1cの MOVEL はヘッドを 1 つ左に動かすものである．テープ (l, s, r) は，スタック L と R を用いて (L S R) で表す．ヘッドを 1 つ左に動かすのは，スタック R からヘッダの下にある記号 S を PUSH(S,R) によりプッシュし，スタック L から 1 つ要素を POP(S,L) によってポップしてヘッダの下を表す記号 S とすることで模倣する．ヘッドを右に 1 つ動かす MOVER は，MOVEL を逆変換したもの $\mathcal{I}[\![\texttt{MOVEL(X)}]\!]$ である．

$\mathcal{T}[\![\langle q_1, \langle s_1, s_2\rangle, q_2\rangle]\!] =$
　[$\overline{q_1}$,(L $\overline{s_1}$ R)] => [$\overline{q_2}$,(L $\overline{s_2}$ R)]
$\mathcal{T}[\![\langle q_1, \leftarrow, q_2\rangle]\!] =$
　[$\overline{q_1}$,T] => {MOVEL(T); Q ^=$\overline{q_1}$; Q ^=$\overline{q_2}$}
$\mathcal{T}[\![\langle q_1, \rightarrow, q_2\rangle]\!] =$
　[$\overline{q_1}$,T] => {MOVER(T); Q ^=$\overline{q_1}$; Q ^=$\overline{q_2}$}
$\mathcal{T}[\![\langle q_1, \downarrow, q_2\rangle]\!] =$ [$\overline{q_1}$,T] => [$\overline{q_2}$,T]

図 2: 遷移規則から R-WHILE の書き換え規則への変換

要素 S をスタック STK にプッシュするマクロ PUSH(S,STK) を図 1dのように定義する．POP(S,STK) ではスタックが空の場合，空白記号がポップされる．この操作によってスタックのボトムが非空白記号である状態（無限のテープの中で有限の記号列を表す）を保つことができる為，任意の回数行うことができる．プッシュの逆操作であるポップ POP(S,STK) は $\mathcal{I}[\![\texttt{PUSH(S,STK)}]\!]$ とする．

4 まとめ

本稿の 3 章において任意の RTM から R-WHILE プログラムへの変換 $\mathcal{T}$ を R-WHILE の書き換え規則によって定義した．従って言語 R-WHILE によって任意の RTM プログラムを書けるということである．以上より R-WHILE の計算モデルは可逆的にチューリング完全であると言うことを示した．すなわち可逆プログラミング言語 R-WHILE によって万能可逆チューリング機械が作れることが示された．

謝辞 本研究の一部は JSPS 科研費 25730049 の助成による．

参考文献

[1] Axelsen, H. B. and Glück, R.: What Do Reversible Programs Compute?, *Foundations of Software Science and Computational Structures. Proceedings* (Hofmann, M., ed.), LNCS, Vol. 6604, Springer-Verlag, pp. 42–56 (2011).

[2] Jones, N. D.: *Computability and Complexity: From a Programming Perspective*, MIT Press (1997). Revised version, available from http://www.diku.dk/~neil/Comp2book.html.

[3] Yokoyama, T., Axelsen, H. B. and Glück, R.: Towards a Reversible Functional Language, *Reversible Computation. Proceedings* (De Vos, A. and Wille, R., eds.), LNCS, Vol. 7165, Springer-Verlag, pp. 14–29 (2012).