卒　業　論　文

可逆プログラム言語 R-WHILE による
万能可逆チューリング機械の構成

2014SE006　青木 崚
2014SE059　増田 大輝
2014SE089　柴田　心太郎

指導教員　横山 哲郎

20yy 年 mm 月

南山大学 理工学部 ソフトウェア工学科

英語タイトル

2014SE006　AOKI Ryo
2014SE059　MASUDA Hiroki
2014SE089　SHIBATA Shintaro

Supervisor　YOKOYAMA Tetsuo

Month 20yy

Department of Software Engineering
Faculty of Engineering
Nanzan University

要約

Abstract

目次

第 1 章

序論

1.1 **背景**

1.2 **期待される効果**

1.3 **課題及び目的**

第 2 章

関連研究

2.1 可逆プログラミング言語

2.1.1 Janus

2.1.2 WHILE 言語

2.1.3 可逆セルオートマトン

第 3 章

可逆チューリング機械

この章ではチューリング機械とチューリング機械に制限を加えた可逆チューリング機械の定義，及び具体例を述べる．

3.1 チューリング機械

実行すべきことに曖昧さがなく明確に記述されていて，記述されたことを機械的に実行すれば出力が得られるものとする．このときそのように記述されたものを機械的な手順，あるいはアルゴリズムと呼ぶ [1]．また，アルゴリズムの特徴として実効性，有限性がある．実行性は，その手順を実際に行うことのできる性質のことであり，有限性は，実行する時間が有限であることを示す．この機械的な手順は直感的な概念であるが，イギリスの数学者 Alan Mathison Turing(1912–1954) は，この概念を厳密に定義することを試み，1936 年にチューリング機械と呼ばれる理論的計算モデルを導入した．チューリング機械は現在のコンピュータをもシミュレートできるとされている．そのため，任意のチューリング機械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．このとき，計算万能性とは計算可能な問題であれば全て計算可能であることをいう．

また，1943 年に Stephen C.Kleen(1909–1994) によって，機械的な手順で計算できるということはチューリング機械の枠組みの中で定式化できるという提唱 (チャーチ・チューリングのテーゼ) がなされた.[1]

3.1.1 チューリング機械の振る舞い (動作)

チューリング機械はマス目に分割された左右に無限長のテープをもち，有限制御部，およびテープ上の記号を読み書きするためのヘッドから構成されている [図 1]．テープには予め入力情報 (2 進数の 0 と 1 や多数のアルファベット) が格納されており，ヘッドが位置するマス目の記号を読み取る．そして, この記号と現在の有限制御部の状態 (内部状態) に依存して，このマス目の記号を書き換え，ヘッドを左か右に 1 コマ移動もしくは不動にし，内部状態を遷移させる．この一連の振る舞い (動作) を繰り返すことで計算を行う．

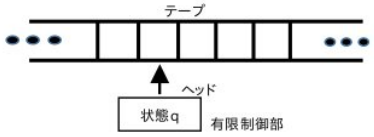

図 3.1: チューリング機械の概要

3.1.2 定義

ここでは，1 テープチューリング機械を $T=(Q,\Sigma,b,\delta,q_s,q_f)$ として定める．(以下,1 テープチューリング機械のことをチューリング機械と呼ぶ．) ただし Q は内部状態の空でない有限集合，Σ はテープ記号の空でない有限集合であり，$b(\in\Sigma)$ は空白記号でテープの有限個のマス目を除く残り全てのマス目に b が格納されていると仮定する．δ 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，$q_f(\subset Q)$ は最終状態の集合とする．

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，$(q,\langle s,s'\rangle,q')$ 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 d の方向に動かし, 状態を q' にすることを意味する．他の定義 [参考文献 (教科書)] においては，5 項組を用いているが，5 項組で設計を行った場合，ヘッダの移動とテープ記号の書き換えを 1 つの遷移規則で行うことができる為，少ない遷移規則の数は多くなってしまうが，遷移規則に対称性があるため，以降の章で説明する可逆チューリング機械を設計する際に前方決定的または後方決定的であるということの判断がしやすいと判断したため今回は 3 項組を用いる．

ここで，チューリング機械が計算を行っているある時点での様子を様相と呼ぶことにする．この時，チューリング機械 $T=(Q,\Sigma,b,\delta,q_s,q_f)$ 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 個以上並べたものの集合を表す．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．

チューリング機械 $T=(Q,\Sigma,b,\delta,q_s,q_f)$ の計算ステップは，$c\underset{T}{\Rightarrow}c'$ を満たすように様相 c を様相 c' に移すものとする．ただし，ここで，b つまり空白記号が無限に続くときを λ として

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

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

チューリング機械 $T=(Q,\Sigma,b,\delta,q_s,q_f)$ の意味を $[\![T]\!]=\{(r,r')|(q_s,(\lambda,b,r))\underset{T}{\Rightarrow}^*(q_f,(\lambda,b,r'))\}$ とする．

3.1.3 具体例

これまでに定義したチューリング機械に基づいて，簡単なチューリング機械を考えてみる．

例　与えられた 2 進数に 1 を加えるチューリング機械　$T_1=(Q_1,(b,0,1),b,\delta_1,q_s,q_f)$ を考える．ただし，$Q_1=\{q_s,q_1,q_2,q_3,q_4,q_f\}$ であり，δ_1 は以下の 3 項組の集合である．

$$
\begin{aligned}
\delta_1=\{&[q_s,\langle b,b\rangle,q_1],[q_1,\rightarrow,q_2],\\
&[q_2,\langle 1,0\rangle,q_1],\\
&[q_2,\langle 0,1\rangle,q_3],[q_3,\leftarrow,q_4],
\end{aligned}
$$

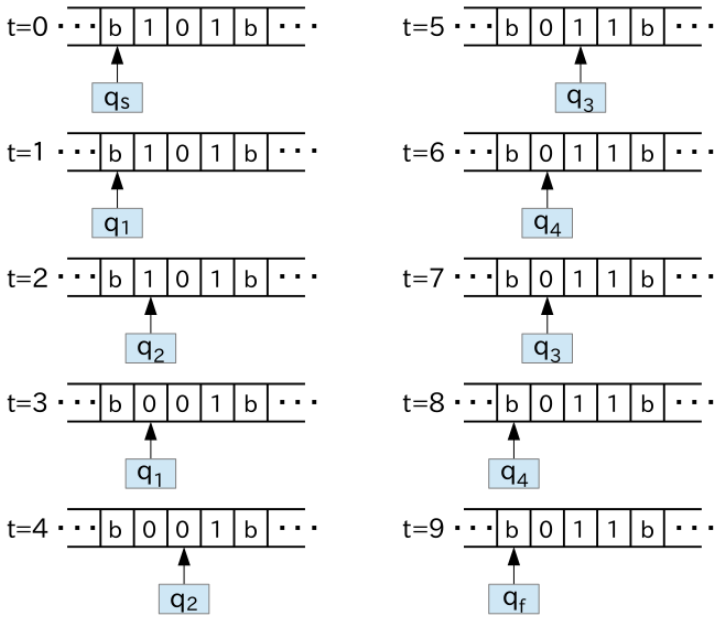

図 3.2: チューリング機械の動作例

$[q_2, \langle b, 1\rangle, q_3]$,
$[q_4, \langle 0, 0\rangle, q_3]$,
$[q_4, \langle 1, 1\rangle, q_3]$,
$[q_4, \langle b, b\rangle, q_f]\}$

T_1 は，非負整数 n の 2 進数表現が書かれたテープが与えられ，ヘッドを 2 進数表現の左隣のます目に置いて初期状態 q_s から動作を開始したとき，n を n に 1 を加えたものに書き換え，2 進数表現の左隣のます目までヘッドを移動し，最終状態 q_f で停止するチューリング機械である (テープに書かれる 2 進数表現は反転されたものとする)．このとき，T_1 は 2 進数表現の一桁目から読んでいく．1 を読んだとき，1 を 0 に書き換える．また，0 または b を読み込んだときにそれを 1 に書き換える．各状態における T_1 の動作を以下に説明する．また実際の動作の例は [図 2]…あらかじめ格納されている 2 進数を「101」とする．t は計算の進行 (遷移) を表す．

q_s：2 進数表現 n の左隣の b を読み，q_1 へ．
q_1：ヘッドを右に 1 コマ移動し，q_2 へ．
q_2：1 を読んだとき，0 に書き換え q_1 へ．b または 0 を読んだとき 1 に書き換え，q_3 へ．
q_3：ヘッドを左に 1 コマ移動し，q_4 へ．
q_4：0 または 1 を読んだ場合，q_3 へ，b を読んだ場合 q_f へ．

また，この遷移規則による様相を上記の記法を用いて表わす．

$$
\begin{array}{lll}
t=0 & (q_s,(\lambda,b,1)) & \underset{T_1}{\Rightarrow} \quad (q_1,(\lambda,b,1)) \\
t=1 & (q_1,(\lambda,b,1)) & \underset{T_1}{\Rightarrow} \quad (q_2,(b,1,0)) \\
t=2 & (q_2,(b,1,0)) & \underset{T_1}{\Rightarrow} \quad (q_1,(b,0,0)) \\
t=3 & (q_1,(b,0,0)) & \underset{T_1}{\Rightarrow} \quad (q_2,(0,0,1)) \\
t=4 & (q_2,(0,0,1)) & \underset{T_1}{\Rightarrow} \quad (q_3,(0,1,1)) \\
t=5 & (q_3,(0,1,1)) & \underset{T_1}{\Rightarrow} \quad (q_4,(b,0,1)) \\
t=6 & (q_4,(b,0,1)) & \underset{T_1}{\Rightarrow} \quad (q_3,(b,0,1)) \\
t=7 & (q_3,(b,0,1)) & \underset{T_1}{\Rightarrow} \quad (q_4,(\lambda,b,0)) \\
t=8 & (q_4,(\lambda,b,0)) & \underset{T_1}{\Rightarrow} \quad (q_3,(\lambda,b,0)) \\
t=9 & (q_3,(\lambda,b,0)) & \underset{T_1}{\Rightarrow} \quad (q_f,(\lambda,b,0))
\end{array}
$$

3.1.4 n テープチューリング機械

先程までは，1 テープのみを考えたが，テープとヘッドがそれぞれ n 個ある n テープチューリング機械も存在する．n テープチューリング機械 T' は $T' = (Q,(\Sigma_1,\Sigma_2,...,\Sigma_n),b,\delta,q_s,q_f)$ として定める．Q,b,q_s,q_f は 1 テープチューリング機械と同様であり，$\Sigma_1,\Sigma_2,...,\Sigma_n$ はそれぞれ n 本目のテープで利用されるテープ記号の空でない有限集合，δ は $((Q\backslash q_f)\times[[\Sigma_1\times\Sigma_2\times...\times\Sigma_n]\times[\Sigma_1\times\Sigma_2\times...\times\Sigma_n]]\times Q)\cup((Q\backslash q_f)\times[d_1\times d_2\times...\times d_n]\times Q)$ の部分集合である．ただし，d はヘッド移動の動作規則 $[\leftarrow,\downarrow,\rightarrow]$ である．

また，任意の n テープチューリング機械に対して，それと等価な 1 テープチューリング機械が存在する．[引用:計算理論と...]

3.2 可逆チューリング機械

可逆チューリング機械とは，内部状態とヘッドの先にある記号に対して，直後の動作が唯一である場合，遷移直後の内部状態とヘッドの記号が決まれば，どのような動作をしたかが決定できる性質をもつチューリング機械である．そのため，時間軸の逆方向に決定的なチューリング機械であると言える．

3.2.1 定義

チューリング機械 $T=(Q,\Sigma,b,\delta,q_s,q_f)$ において，任意の異なる遷移規則 $m_1=[p,\langle s,t\rangle,q]$，$m_2=[p',\langle s',t'\rangle,q']\in\delta$ に対して，$p=p'$ ならば,$s\neq s'$ を満たすとき T を局所的に前方決定的または，決定性チューリング機械と呼ぶ．また，T の任意の異なる遷移規則 $m_1=[p,\langle s,t\rangle,q]$，$m_2=[p',\langle s',t'\rangle,q']\in\delta$ に対して $q=q'$ ならば,$t\neq t'$ を満たすとき T を局所的に後方決定的または，可逆チューリング機械と呼ぶ.

以下，可逆チューリング機械とは決定性チューリング機械の条件と可逆チューリング機械の条件をもち最終状態からの遷移及び，初期状態への遷移がないものとする．

3.2.2 具体例

これまでに定義した可逆チューリング機械に基づいて，簡単な可逆チューリング機械を考えてみる．

例　与えられた 2 進数の 0 と 1 を反転させる可逆チューリング機械　$T_2 = (Q_2, (0,1), b, \delta_2, q_s, q_f)$ を考える．ただし，$Q_2 = \{q_s, q_1, q_2, q_3, q_4, q_f\}$ であり，δ_2 は以下の 3 項組の集合である．

$$
\begin{aligned}
\delta_2 = \{&[q_s, \langle b, b\rangle, q_1], [q_1, \rightarrow, q_2],\\
&[q_2, \langle 0, 0\rangle, q_1],\\
&[q_2, \langle 1, 1\rangle, q_1],\\
&[q_2, \langle b, b\rangle, q_3], [q_3, \leftarrow, q_4],\\
&[q_4, \langle 0, 1\rangle, q_3],\\
&[q_4, \langle 1, 0\rangle, q_3],\\
&[q_4, \langle b, b\rangle, q_f]\}
\end{aligned}
$$

T_2 は，2 進数表現が書かれたテープが与えられ，ヘッドを 2 進数表現の左隣のます目に置いて初期状態 q_s から動作を開始したとき，書かれている 2 進数表現を反転したものに書き換え，2 進数表現の左隣のます目までヘッドを移動し，最終状態 q_f で停止するチューリング機械である．このとき，T_2 は 2 進数表現の 1 つ右隣のます目までヘッド移動する．その後，ヘッドを初期の位置に移動させながら，1 を読み込んだとき 0 に書き換える．また，0 を読み込んだとき 1 に書き換える．各状態における T_2 の動作を以下で説明する．また実際の動作の例は…である．

q_s：2 進数表現 n の左隣の b を読み，q_1 へ．
q_1：ヘッドを右に 1 コマ移動し，q_2 へ．
q_2：b 以外を読んだとき，q_1 へ．また，b を読んだとき，q_3 へ．
q_3：ヘッドを左に 1 コマ移動し，q_4 へ．
q_4：0 を読んだとき，1 に書き換える．また，1 を読み込んだ場合 0 に書き換え，q_3 へ．そして，b を読んだとき
　q_f へ．

T_2 において，q_1 が 1 番目の項として現れている 3 項組は (i) $[q_2, \langle 0, 0\rangle, q_1]$,(ii) $[q_2, \langle 1, 1\rangle, q_1]$ または (iii) $[q_2, \langle b, b\rangle, q_3]$ の 3 つである．ある時刻において T_1 が状態 q_1 であり，現在読んでいるます目が 0 ならば (i)，1 ならば (ii) または b ならば (iii) が直後に実行されるということが一意に決まる．これと同様のことは状態 q_4 でも言えるため，T_2 は局所的に前方決定的である．また，q_1 が 3 番目の項として現れている 3 項組は (iv) $[q_s, \langle b, b\rangle, q_1]$，(v) $[q_2, \langle 0, 0\rangle, q_1]$ または (vi) $[q_2, \langle 0, 0\rangle, q_1]$ の 3 つである．ある時刻において T_1 が状態 q_1 であり，現在読んでいるます目が b ならば (iv)，0 ならば (v) または 1 ならば (vi) が直後に実行されるということが一意に決まる．これと同様のことは状態 q_3 でも言えるため，T_2 は局所的に後方決定的である．そして T_1 は最終状態からの遷移および初期状態への遷移はない．以上より T_2 は可逆チューリング機械である．

3.2.3 可逆シミュレーション

様々な理由により実物による可逆化が困難または不可能な場合に，実物から特徴的な要素を抽出してモデル化し，それとほぼ同じ条件や規則に従った別のシステムで模擬することを可逆シミュレーションという．

3.2.4 非可逆のチューリング機械の可逆にする例

ここでは非可逆なチューリング機械の遷移規則を工夫することで，可逆チューリング機械に作り変えること (可逆シミュレート) が出来るということを実際に例を用いて説明する．例には 3.3 節で説明した T_1 を用いる．T_1 の遷移規則 δ_1 は以下の 3 項組の集合であった．

$\delta_1 = \{[q_s, \langle b, b\rangle, q_1], [q_1, \rightarrow, q_2],$
$[q_2, \langle 1, 0\rangle, q_1],$
$[q_2, \langle 0, 1\rangle, q_3], [q_3, \leftarrow, q_4],$
$[q_2, \langle b, 1\rangle, q_3],$
$[q_4, \langle 0, 0\rangle, q_3],$
$[q_4, \langle 1, 1\rangle, q_3],$
$[q_4, \langle b, b\rangle, q_f]\}$

T_1 は次の理由により非可逆なチューリング機械である．q_3 が 3 番目の項として現れている 3 項組は (i) $[q_2, \langle 0, 1\rangle, q_3]$，(ii) $[q_2, \langle b, 1\rangle, q_3]$，(iii) $[q_4, \langle 0, 0\rangle, q_3]$ または (iv) $[q_4, \langle 1, 1\rangle, q_3]$ の 4 つである．そのうち (i),(ii) または (iv) の 3 つの遷移規則は書き換えた後の記号がどれも 1 である．つまり，ある時刻において T_1 が状態 q_3 にあり，現在読んでいるます目が 1 ならば，(i),(ii) または (iv) のどれが直前に実行されたかが一意に決まらない．すなわち，可逆チューリング機械の条件である局所的に後方決定的であるという条件を満たさないため，T_1は 非可逆なチューリング機械である．

この非可逆なチューリング機械 T_1 の遷移規則を可逆になるように設計した可逆チューリング機械 $T_{1-2} = (Q_3, (b, 0, 1), b, \delta_1, q_s, q_f)$ とする．ただし，$Q_3 = \{q_s, q_1, q_2, q_3, q_4, q_5, q_6{\cdot}q_7, q_8, q_9, q_{10}, q_f\}$ であこのとき，遷移規則 δ_{1-2} 以下のように表す．

$\delta_{1-2} = \{[q_s, \langle b, b\rangle, q_1], [q_1, \rightarrow, q_2],$
$[q_2, \langle 1, 0\rangle, q_1],$
$[q_2, \langle 0, 1\rangle, q_3], [q_3, \rightarrow, q_5],$
$[q_5, \langle 0, 0\rangle, q_7],$
$[q_5, \langle 1, 1\rangle, q_7],$
$[q_2, \langle b, 1\rangle, q_4], [q_4, \rightarrow, q_6],$
$[q_6, \langle b, b\rangle, q_7], [q_7, \leftarrow, q_8],$
$[q_8, \langle 1, 1\rangle, q_9], [q_9, \leftarrow, q_{10}],$
$[q_{10}, \langle 0, 0\rangle, q_9],$
$[q_{10}, \langle b, b\rangle, q_f]\}$

T_1 は 1 を加えた直後の状態から直前に 1 を加えることで 2 進数表現に桁上げが行われたのかを判断することが出来なかった．そこで T_{1-2} では 1 を加える動作の後にヘッドを 2 進数表現の 1 つ左まで移動するのではなく，さらに右に 1 つヘッドを移動し、そのます目に書かれる記号が空白記号であるかそうでないかを確認する動作を追加した．そうすることで 2 進数表現に 1 が加えられたとき，桁上げが行われていたかを判断することが出きるようになる．

章○では単純な可逆チューリング機械について説明したが，この例の様に非可逆なチューリング機械は単純に設計されていても，可逆なものに再度設計 (可逆シミュレーション) することで単純であった遷移規則が複雑になってしまうことがある．

また，この遷移規則による様相を 3.1 の記法を用いて表わす．

$$
\begin{array}{lll}
t=0 & (q_s,(\lambda,b,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_1,(\lambda,b,1)) \\
t=1 & (q_1,(\lambda,b,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_2,(b,1,0)) \\
t=2 & (q_2,(b,1,0)) & \underset{T_{1-2}}{\Longrightarrow} (q_1,(b,0,0)) \\
t=3 & (q_1,(b,0,0)) & \underset{T_{1-2}}{\Longrightarrow} (q_2,(0,0,1)) \\
t=4 & (q_2,(0,0,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_3,(0,1,1)) \\
t=5 & (q_3,(0,1,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_5,(1,1,b)) \\
t=6 & (q_5,(1,1,b)) & \underset{T_{1-2}}{\Longrightarrow} (q_7,(1,1,b)) \\
t=7 & (q_7,(1,1,b) & \underset{T_{1-2}}{\Longrightarrow} (q_8,(0,1,1)) \\
t=8 & (q_8,(0,1,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_9,(0,1,1)) \\
t=9 & (q_9,(0,1,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_{10},(b,0,1)) \\
t=10 & (q_{10},(b,0,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_9,(b,0,1)) \\
t=11 & (q_9,(b,0,1)) & \underset{T_{1-2}}{\Longrightarrow} (q_{10},(\lambda,b,0)) \\
t=12 & (q_{10},(\lambda,b,0)) & \underset{T_{1-2}}{\Longrightarrow} (q_f,(\lambda,b,0))
\end{array}
$$

3.2.5 3 テープ可逆チューリング機械

2.4 節では，1 テープの非可逆チューリング機械を 1 テープの可逆チューリング機械にする可逆シミュレーションを行ったが，本節では非可逆なチューリング機械を可逆なチューリング機械を構成するために，3 本のテープを持つチューリング機械を用いる．3 テープ可逆チューリング機械　T'' を 1.4 節の定義より $T''=(Q,(\Sigma_1,\Sigma_2,\Sigma_3),b,\delta,q_s,q_f)$ と定める．T'' の 1 本目のテープは T のテープをそのまま表現し，ヘッドの位置も同じである．　2 本目のテープは各ステップで T のどの 3 項組が実行されたのかという記述を記録する．3 本目のテープは 1 本目のテープの記述をコピーし，可逆化によって初期状態に戻る 1 本目のテープの代わりに，結果を記録する．Q,b,q_s,q_f は 1 テープチューリング機械と同様であり，$\Sigma_1,\Sigma_2,\Sigma_3$ は 3 本のテープそれぞれで使用される記号の有限集合である．

δ は 3 項組集合であるが，各項は,$[p,\langle s,t\rangle,q]$ または $[p,d,q]$ の形であった 1 テープのチューリング機械の遷移規則である s,t,d がそれぞれ 3 テープ分必要になるため，3 テープチューリング機械では $[p,\langle[s_1,s_2,s_3],[t_1,t_2,t_3]\rangle,q]$ または $[p,[d_1,d_2,d_3],q]$ の形をしている．そのため, 前者の 3 項組は T'' が状態 p で，3 つのヘッドの記号 s_1,s_2,s_3 を読んだ場合，記号 t_1,t_2,t_3 に書き換え，状態を q にすることを意味する．後者の 3 項組は T'' が状態 q の場合，ヘッドを d_1,d_2,d_3 の方向に動かし，状態 q にすることを意味する．

可逆性の定義も 1 テープの場合と同様である．すなわち任意の異なる 2 つの 3 項組 $n_1=[p,\langle[s_1,s_2,s_3],[t_1,t_2,t_3]\rangle,q]$,$n_2=[p',\langle[s'_1,s'_2,s'_3],[t'_1,t'_2,t'_3]\rangle,q']$ に対して $p'=q'$ ならば $[t_1,t_2,t_3]\neq[t'_1,t'_2,t'_3]$ という関係が成り立つ．

可逆化 (Bennett)

まず計算における可逆であることの利点を述べる．可逆計算において単純でわかりやすいことから，計算に要するエネルギーを減らすことが可能である点である．出力結果から入力の値を求めることができる性質を利用し，不必要な情報 (以下ゴミ情報と呼ぶ) を消去することで，計算に必要なエネルギーを最小にすることが出来る．3 テープ可逆チューリング機械の動作の過程は，大きく以下の 3 つに分割される．

1.「2 本目のテープに動作履歴を残しながら計算を実行する過程」
2.「計算結果を 3 本目のテープにコピーする過程」
3.「動作履歴を可逆的に消去しながら逆の動作を実行する過程」

このようにして動作を実行することで，最終的に 1 本目のテープに入力が残り，3 本目のテープ に答えが出力され，ゴミ情報は綺麗に消去された状態で停止することが出来る．

Bennett の可逆化の例

3-2-2 の記法を用いて，3-1 章の具体例で扱った 1 テープ非可逆チューリング機械を 3 テープ可逆チューリング機械でシミュレートする．

チューリング機械 T_1 をシミュレートする可逆チューリング機械 T_3 は，

T_3=$(\{q_s, q_1, q_2, q_3, q_4, q_f, c_b, c_1, c_2, c_3, c_4, c_5, p_s, p_1, p_2, p_3, p_4, p_f\}, \{b, 0, 1\}, \{0, 1, 2, 3, 4, 5, 6, 7\}, \{b, 0, 1\}, 0, \delta_3, q_s, \{p_s\})$

遷移規則 δ_3 は，

δ_3=$\{$ $[q_s, \langle[b, b, b], [b, 1, b]\rangle, q_1]$, $[q_1, [\rightarrow, \rightarrow, \downarrow], q_2]$,
$[q_2, \langle[1, b, b], [0, 2, b]\rangle, q_1]$, $[q_3, [\leftarrow, \rightarrow, \downarrow], q_4]$,
$[q_2, \langle[0, b, b], [1, 3, b]\rangle, q_3]$,
$[q_2, \langle[b, b, b], [1, 4, b]\rangle, q_3]$,
$[q_4, \langle[0, b, b], [0, 5, b]\rangle, q_3]$,
$[q_4, \langle[1, b, b], [1, 6, b]\rangle, q_3]$,
$[q_4, \langle[b, b, b], [b, 7, b]\rangle, q_f]$, $[q_f, [\downarrow, \rightarrow, \downarrow], c_0]$,
$[c_0, \langle[b, b, b], [b, b, b]\rangle, c_1]$, $[c_1, [\rightarrow, \downarrow, \rightarrow], c_2]$,
$[c_2, \langle[1, b, b], [1, b, 1]\rangle, c_1]$,
$[c_2, \langle[0, b, b], [0, b, 0]\rangle, c_1]$,
$[c_2, \langle[b, b, b], [b, b, b]\rangle, c_3]$,$[c_3, [\leftarrow, \downarrow, \leftarrow], c_4]$,
$[c_4, \langle[1, b, 1], [1, b, 1]\rangle, c_3]$,
$[c_4, \langle[0, b, 0], [0, b, 0]\rangle, c_3]$,
$[c_4, \langle[b, b, b], [b, b, b]\rangle, c_5]$,$[c_5, [\downarrow, \leftarrow, \downarrow], p_f]$,
$[p_f, \langle[b, 7, b], [b, b, b]\rangle, p_4]$,
$[p_3, \langle[1, 6, b], [1, b, b]\rangle, p_4]$,
$[p_3, \langle[0, 5, b], [0, b, b]\rangle, p_4]$,
$[p_3, \langle[1, 4, b], [b, b, b]\rangle, p_2]$,
$[p_3, \langle[1, 3, b], [0, b, b]\rangle, p_2]$,
$[p_1, \langle[0, 2, b], [1, b, b]\rangle, p_2]$, $[p_4, [\rightarrow, \leftarrow, \downarrow], p_3]$,
$[p_1, \langle[b, 1, b], [b, b, b]\rangle, p_s]$, $[p_2, [\leftarrow, \leftarrow, \downarrow], p_1]$ $\}$

以下は上で述べた 3 テープ可逆チューリング機械の動作に従いながら解説する．まず 1 つ目の過程であるが，δ_3 の最初の 9 個の 3 項組は δ_2 の 9 個の 3 項組に対応している．これらは 1 本目のテープ上で T_2 の動作を以前と同様にシミュレートするだけでなく，2 本目のテープに δ_2 の $1-7$ のどの 3 項組が使われかを示す $1-7$ の数字を書き込むことで，可逆チューリング機械の条件を満たしながら計算を実行することが出来る.2 つ目の過程では，T_3 は，T_2 の最終状態である q_f に遷移したとき，c_s に移し，その後 $c_i(i = 0, 1, ..., 5)$ を使って 1 本目のテープに記された答えを 3 本目のテープにコピーする．コピー後にゴミ情報を消去するため状態を p_f に遷移する．3 つ目の過程では，δ_3 の最後の 9 個の 3 項組は最初の 9 個の 3 項組の逆から，つまり q_f から q_s までの動作を行う．そのため p と q はそれぞれ対応していて，最終的には状態 p_s で停止する．

また，この遷移規則による様相を 3.1 の記法を用いて表わす.

$$
\begin{array}{lll}
t=0 & (q_s,((\lambda,b,1),(\lambda,b,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_1,((\lambda,b,1),(\lambda,1,\lambda),(\lambda,b,\lambda))) \\
t=1 & (q_1,((\lambda,b,1),(\lambda,1,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_2,((b,1,0),(1,b,\lambda),(\lambda,b,\lambda))) \\
t=2 & (q_2,((b,1,0),(1,b,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_1,((b,0,0),(1,2,\lambda),(\lambda,b,\lambda))) \\
t=3 & (q_1,((b,0,0),(1,2,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_2,((0,0,1),(2,b,\lambda),(\lambda,b,\lambda))) \\
t=4 & (q_2,((0,0,1),(2,b,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_3,((0,1,1),(2,3,\lambda),(\lambda,b,\lambda))) \\
t=5 & (q_3,((0,1,1),(2,3,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_4,((b,0,1),(3,b,\lambda),(\lambda,b,\lambda))) \\
t=6 & (q_4,((b,0,1),(3,b,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_3,((b,0,1),(3,5,\lambda),(\lambda,b,\lambda))) \\
t=7 & (q_3,((b,0,1),(3,5,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_4,((\lambda,b,0),(5,b,\lambda),(\lambda,b,\lambda))) \\
t=8 & (q_4,((\lambda,b,0),(5,b,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (q_f,((\lambda,b,0),(5,7,\lambda),(\lambda,b,\lambda))) \\
t=9 & (q_f,((\lambda,b,0),(5,7,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_0,((\lambda,b,0),(7,b,\lambda),(\lambda,b,\lambda))) \\
t=10 & (c_0,((\lambda,b,0),(7,b,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_1,((\lambda,b,0),(7,b,\lambda),(\lambda,b,\lambda))) \\
t=11 & (c_1,((\lambda,b,0),(7,b,\lambda),(\lambda,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_2,((b,0,1),(7,b,\lambda),(b,b,\lambda))) \\
t=12 & (c_2,((b,0,1),(7,b,\lambda),(b,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_1,((b,0,1),(7,b,\lambda),(b,0,\lambda))) \\
t=13 & (c_1,((b,0,1),(7,b,\lambda),(b,0,\lambda))) & \underset{T_3}{\Rightarrow} (c_2,((0,0,1),(7,b,\lambda),(0,b,\lambda))) \\
t=14 & (c_2,((0,0,1),(7,b,\lambda),(0,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_1,((0,1,1),(7,b,\lambda),(0,1,\lambda))) \\
t=15 & (c_1,((0,1,1),(7,b,\lambda),(0,1,\lambda))) & \underset{T_3}{\Rightarrow} (c_2,((1,1,b),(7,b,\lambda),(1,b,\lambda))) \\
t=16 & (c_2,((1,1,b),(7,b,\lambda),(1,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_1,((1,1,b),(7,b,\lambda),(1,1,\lambda))) \\
t=17 & (c_1,((1,1,b),(7,b,\lambda),(1,1,\lambda))) & \underset{T_3}{\Rightarrow} (c_2,((1,b,\lambda),(7,b,\lambda),(1,b,\lambda))) \\
t=18 & (c_2,((1,b,\lambda),(7,b,\lambda),(1,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_3,((1,b,\lambda),(7,b,\lambda),(1,b,\lambda))) \\
t=19 & (c_3,((1,b,\lambda),(7,b,\lambda),(1,b,\lambda))) & \underset{T_3}{\Rightarrow} (c_4,((1,1,b),(7,b,\lambda),(1,1,b))) \\
t=20 & (c_4,((1,1,b),(7,b,\lambda),(1,1,b))) & \underset{T_3}{\Rightarrow} (c_3,((1,1,b),(7,b,\lambda),(1,1,b))) \\
t=21 & (c_3,((1,1,b),(7,b,\lambda),(1,1,b))) & \underset{T_3}{\Rightarrow} (c_4,((0,1,1),(7,b,\lambda),(0,1,1))) \\
t=21 & (c_4,((0,1,1),(7,b,\lambda),(0,1,1))) & \underset{T_3}{\Rightarrow} (c_3,((0,1,1),(7,b,\lambda),(0,1,1))) \\
t=22 & (c_3,((0,1,1),(7,b,\lambda),(0,1,1))) & \underset{T_3}{\Rightarrow} (c_4,((b,0,1),(7,b,\lambda),(b,0,1))) \\
t=23 & (c_4,((b,0,1),(7,b,\lambda),(b,0,1))) & \underset{T_3}{\Rightarrow} (c_3,((b,0,1),(7,b,\lambda),(b,0,1))) \\
t=24 & (c_3,((b,0,1),(7,b,\lambda),(b,0,1))) & \underset{T_3}{\Rightarrow} (c_4,((\lambda,b,0),(7,b,\lambda),(\lambda,b,0))) \\
t=25 & (c_4,((\lambda,b,0),(7,b,\lambda),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (c_5,((\lambda,b,0),(7,b,\lambda),(\lambda,b,0))) \\
t=26 & (c_5,((\lambda,b,0),(7,b,\lambda),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_f,((\lambda,b,0),(5,7,b),(\lambda,b,0))) \\
t=27 & (p_f,((\lambda,b,0),(5,7,b),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_4,((\lambda,b,0),(5,b,\lambda),(\lambda,b,0))) \\
t=28 & (p_4,((\lambda,b,0),(5,b,\lambda),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_3,((b,0,1),(3,5,b),(\lambda,b,0))) \\
t=29 & (p_3,((b,0,1),(3,5,b),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_4,((b,0,1),(3,b,\lambda),(\lambda,b,0))) \\
t=30 & (p_4,((b,0,1),(3,b,\lambda),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_3,((0,1,1),(2,3,b),(\lambda,b,0))) \\
t=31 & (p_3,((0,1,1),(2,3,b),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_2,((0,0,1),(2,b,\lambda),(\lambda,b,0))) \\
t=32 & (p_2,((0,0,1),(2,b,\lambda),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_1,((b,0,0),(1,2,b),(\lambda,b,0))) \\
t=33 & (p_1,((b,0,0),(1,2,b),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_2,((b,1,0),(1,b,\lambda),(\lambda,b,0))) \\
t=34 & (p_2,((b,1,0),(1,b,\lambda),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_1,((\lambda,b,1),(\lambda,1,\lambda),(\lambda,b,0))) \\
t=35 & (p_1,((\lambda,b,1),(\lambda,1,\lambda),(\lambda,b,0))) & \underset{T_3}{\Rightarrow} (p_s,((\lambda,b,1),(\lambda,b,\lambda),(\lambda,b,0)))
\end{array}
$$

この方法を一般化することにより，次のことが言える．任意の 1 テープチューリング機械 T_a に対し，それをシミュレートする 3 テープの可逆チューリング機械 T'_a では，T_a に記号列 x を与えたとき，記号列 y を書き出

して最終状態で停止するならば，T'_a に x を与えたとき，ゴミ情報である動作履歴を残すことなく x と y だけを書き出し，最終状態で停止することができる．

第 4 章

R-WHILE 言語での実装

この章では命令型可逆プログラミング言語 R-WHILE（以下，単に R-WHILE と呼ぶことにする) を万能可逆チューリング機械に変換する規則を与えることで，R-WHILE が万能可逆チューリング機械を実装可能であることを示す．

4.1 R-WHILE について

ここでは R-WHILE[7] について説明する．

R-WHILE は，Jones の言語 WHILE を可逆化したものである．R-WHILE は非可逆なプログラムを記述することができない．そのため単射関数しか表すことはできない．この言語の特徴は木構造のデータをもっていることである．それにより単純な方法で身近なデータ構造をモデリングが可能である．木構造のデータは同じ可逆プログラミング言語である Janus ももっているが，R-WHILE は他の木構造のデータを持っている言語に比べてとても単純な構造である．R-WHILE の構文規則は図 4.1である．プログラム P はただ一つの入り口と出口の点 (read,write) をもち，命令 C がプログラムの本体である．

プログラムのデータ領域 D はアトム nil とすべての組 (d_1, d_2) を含む（$d_1, d_2 \in D$）最小の集合である．Var は変数名の無限集合である．本稿では $d, e, f, ... \in D$ とする．また，X,Y,Z,... $\in$ Vars とする．式は変数 X，定数 d，または複数の演算子からなる (先頭と素の残りを表す hd と tail，組を表す cons また等号を表す =?）からなる．パターンは式の部分集合であり，変数 X，定数 d またはパターンのペアを表す cons Q R からなる．本稿では以後ペアを表す cons E F または cons Q R をそれぞれ (E.F) または (Q.R) と表記する．パターンは線形的でなくてはならない．

次に命令 C について説明する．非可逆なプログラミング言語における代入は左辺の変数の値を上書きする．そして，代入後に再び値を取り戻すことはできない．そのため，可逆プログラミング言語に用いることは出来ない．R-WHILE では可逆的代入 X ^= E を用いる．X ^= E では，X の値が nil のとき X の値を E の値にする．また，X の値が E の値と等しいとき，X の値を nil にする．X ^= E では左辺の変数 X に右辺の式 E があらわれてはならない．なぜなら，X ^= X のとき，X がどのような値であっても X が nil の値になるため，単射ではなくなってしまうからである．可逆的置換 Q <= R は Q の変数を R の変数を使って更新する．例えば

```
{Q = nil,R = a}
     Q <= R;
{Q = a,R = nil}
```

となる．可逆的代入と比べ両辺に表されるのはパターンの Q と R である．可逆的条件文 if E then C else D fi F の制御フローの分岐はテスト E に依存する．もし真であれば命令 C が実行され，アサーション E は真でなくてはならない．また，もし偽であった場合命令 D が実行され，アサーション E は偽でなくてはならない．

```
E,F ::= X | d | cons E F | hd E | tl E | =? E F     式
Q,R ::= X | d | cons Q R                            パターン
C,D ::= X ^= E                                      命令
      | Q <= R
      | C; D
      | if E then C else D fi F
      | from E do C loop D until F
P   ::= X; C; write Y                               R-WHILE のプログラム
```

図 4.1: 言語 R-WHILE の構文規則

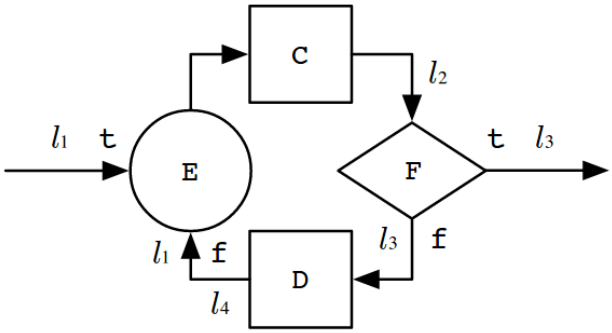

図 4.2: while loop のフローチャート

E と F の返す値が対応していない場合，条件文は定義されない．可逆的ループ from E do C loop D until F は図 4.2の様に描くことができる．ループを行うとき，アサーション E は真でなくてはならない（図 4.2の t）．そして，命令 C が実行される．実行後のテスト F が真であれば繰り返しは続行される．もし F が偽であった場合, 命令 D が実行される．また，アサーション E は偽でなくてはならない（図 4.2の f）．

R-WHILE にはプログラム逆変換機 $\mathcal{I}$ が定義されている (図 4.4)．それによって反転されたプログラムを再帰的降下によって得ることができる．例えば，図 4.5はリストを反転させる R-WHILE のプログラム reverse と逆変換機 $\mathcal{I}$ によって求められた reverse のプログラム $\mathcal{I}\,[\![\texttt{reverse}]\!]$ である．$\mathcal{I}\,[\![\texttt{reverse}]\!]$ は X と Y の位置が入れ替わっていることを除き，reverse と同じプログラムの構造をしていることがわかる．

4.1.1 可逆チューリング機械から R-WHILE への変換

この章では実際に可逆チューリング機械を R-WHILE のプログラムで模倣する．
図 4.6aにチューリング機械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．図 4.6aの main プログラムは入力としてチューリング機械のテープ上に書かれている記号列 R を読み込む．その後，計算を実行し，書き換えられた記号列 R' を出力するというものである．main プログラム内の Q はチューリング機械の内部状態を表している．また，T はチューリング機械のテープの状態を表している．そして，組 (Q,T) はチューリング機械の様相を表している．プログラム内のループでは，チューリング機械の内部状態が初期状態から最終状態に遷移するまでマクロ STEP を繰り返し実行する．
図 4.6b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

$$
\begin{array}{lcl}
\mathcal{E}[\![d]\!]\sigma & = & d \\
\mathcal{E}[\![\texttt{X}]\!]\sigma & = & \sigma(\texttt{X}) \\
\mathcal{E}[\![\texttt{hd E}]\!]\sigma & = & \texttt{e} \text{ if } \mathcal{E}[\![\texttt{E}]\!]\sigma = \texttt{(e.f)} \\
\mathcal{E}[\![\texttt{tl E}]\!]\sigma & = & \texttt{f} \text{ if } \mathcal{E}[\![\texttt{E}]\!]\sigma = \texttt{(e.f)}
\end{array}
\qquad
\begin{array}{lcl}
\mathcal{E}[\![\texttt{cons E F}]\!]\sigma & = & (\mathcal{E}[\![\texttt{E}]\!]\sigma, \mathcal{E}[\![\texttt{F}]\!]\sigma) \\
\mathcal{E}[\![\texttt{=? E F}]\!]\sigma & = & \begin{cases} \texttt{true} & \text{if } \mathcal{E}[\![\texttt{E}]\!]\sigma = \mathcal{E}[\![\texttt{F}]\!]\sigma \\ \texttt{false} & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{Q}[\![d]\!]\sigma & = & (d, \sigma) \\
\mathcal{Q}[\![\texttt{X}]\!](\sigma \uplus \{\texttt{X} \mapsto d\}) & = & (d, \sigma \uplus \{\texttt{X} \mapsto \texttt{nil}\}) \\
\mathcal{Q}[\![\texttt{cons Q R}]\!]\sigma & = & ((d_1 \,.\, d_2)), \sigma_2 \text{ where } (d_1, \sigma_1) = \mathcal{Q}[\![\texttt{Q}]\!]\sigma \ \wedge \ (d_2, \sigma_2) = \mathcal{Q}[\![\texttt{R}]\!]\sigma_1
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{C}[\![\texttt{X ^= E}]\!](\sigma \uplus \{\texttt{X} \mapsto d\}) & = & \sigma \uplus \{\texttt{X} \mapsto d \odot \mathcal{E}[\![\texttt{E}]\!]\sigma\} \\
\mathcal{C}[\![\texttt{Q <= R}]\!] & = & \mathcal{Q}[\![\texttt{Q}]\!]^{-1}(\mathcal{Q}[\![\texttt{R}]\!]\sigma) \\
\mathcal{C}[\![\texttt{C; D}]\!]\sigma & = & \mathcal{C}[\![\texttt{D}]\!](\mathcal{C}[\![\texttt{C}]\!]\sigma) \\
\mathcal{C}[\![\texttt{if E then C else D fi F}]\!] & = & \begin{cases} \sigma' & \text{if } \mathcal{E}[\![\texttt{E}]\!]\sigma = \texttt{true} \ \wedge \ \sigma' = \mathcal{C}[\![\texttt{C}]\!]\sigma \ \wedge \ \mathcal{E}[\![\texttt{F}]\!]\sigma' = \texttt{true} \\ \sigma' & \text{if } \mathcal{E}[\![\texttt{E}]\!]\sigma = \texttt{false} \ \wedge \ \sigma' = \mathcal{C}[\![\texttt{D}]\!]\sigma \ \wedge \ \mathcal{E}[\![\texttt{F}]\!]\sigma' = \texttt{false} \end{cases} \\
\mathcal{C}[\![\texttt{from E do C loop D until F}]\!] & = & \sigma' \quad \text{if} \mathcal{E}[\![\texttt{E}]\!] = \texttt{true} \wedge \sigma' = fix(F)(\sigma) \\
 & & \text{where } F(\varphi) \ = \ \{(\sigma, \sigma_1) \mid \sigma_1 = \mathcal{C}[\![\texttt{C}]\!]\sigma \ \wedge \ \mathcal{E}[\![\texttt{F}]\!]\sigma_1 = \texttt{true}\} \cup \\
 & & \qquad \{(\sigma, \sigma_3) \mid \sigma_1 = \mathcal{C}[\![\texttt{C}]\!]\sigma \ \wedge \ \mathcal{E}[\![\texttt{F}]\!]\sigma_1 = \texttt{false} \wedge \\
 & & \qquad\qquad \sigma_2 = \mathcal{C}[\![\texttt{D}]\!]\sigma_1 \ \wedge \ \mathcal{E}[\![\texttt{E}]\!]\sigma_2 = \texttt{false} \wedge \\
 & & \qquad\qquad \sigma_3 = \varphi(\sigma_2)\}
\end{array}
$$

$$
\mathcal{P}[\![\texttt{P}]\!]D = D' \quad \text{if P is } \texttt{read X; C; write Y} \ \wedge \ \mathcal{C}[\![\texttt{C}]\!](\sigma^{\texttt{P}}_{\texttt{X}}(D)) = \sigma^{\texttt{P}}_{\texttt{Y}}(D')
$$

図 4.3: R-WHILE の表示的意味

$$
\begin{array}{lcl}
\mathcal{I}[\![\texttt{X ^= E}]\!] & = & \texttt{X ^= E} \\
\mathcal{I}[\![\texttt{Q <= R}]\!] & = & \texttt{Q <= R} \\
\mathcal{I}[\![\texttt{C; D}]\!] & = & \mathcal{I}[\![\texttt{D}]\!]\texttt{;} \ \mathcal{I}[\![\texttt{C}]\!] \\
\mathcal{I}[\![\texttt{if E then C else D fi F}]\!] & = & \texttt{if F then } \mathcal{I}[\![\texttt{C}]\!] \texttt{ else } \mathcal{I}[\![\texttt{D}]\!] \texttt{ fi E} \\
\mathcal{I}[\![\texttt{from E do C loop D until F}]\!] & = & \texttt{from F do } \mathcal{I}[\![\texttt{C}]\!] \texttt{ loop } \mathcal{I}[\![\texttt{D}]\!] \texttt{ until E} \\
\mathcal{I}[\![\texttt{read X; C; write Y}]\!] & = & \texttt{read Y;} \mathcal{I}[\![\texttt{C}]\!]\texttt{; write X}
\end{array}
$$

図 4.4: R-WHILE の逆変換機 $\mathcal{I}$

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 4.7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．可逆チューリング機械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 4.6cMOVEL はヘッドを一つ左に動かすためのものである．チューリング機械のテープ (l, s, r) はスタック L と R を用いて (L S R) として表す．

```
read X;
 from (=? Y nil)
 loop (Z.X) <= X;
      Y<= (Z.Y)
 until(=? X nil);
write Y
```

(a) プログラム reverse

```
read Y;
 from (=? X nil)
 loop (Z.Y) <= Y;
      X <= (Z.X)
 until(=? Y nil);
write X
```

(b) プログラム $\mathcal{I}[\![\texttt{reverse}]\!]$

図 4.5: 逆変換器 $\mathcal{I}$ の例

```
read R;
  Q ^= q_s;
  T <= (nil b R);
  from (=? Q q_s) loop
    STEP(Q,T)
  until (=? Q q_f);
  (nil b R') <= T;
  Q ^= q_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
 [b,nil] => [nil,nil]
 [S,STK] => [nil,(S.STK)]
```

(d) マクロ PUSH

図 4.6: RTM を模倣する R-WHILE プログラム

$\mathcal{T}[\![\langle q_1, \langle s_1, s_2\rangle, q_2\rangle]\!] =$
    [$\overline{q_1}$,(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 4.7: 遷移規則から R-WHILE の書き換え規則への変換器 $\mathcal{T}$

第 5 章

結論

5.1 **結果**

5.2 **考察**

謝辞

付録

参考文献

[1] Stephen C.Kleene.:The Church-Turing Thesis(1997).

[2] Ryo Aoki,Shintaro Shibata,Tetsuo Yokoyama.: A universal reversible Turing machine program inreversible programming language R-WHILE(2016).

[3] 丸岡 章: “計算理論とオートマトン言語理論 コンピュータの原理を明かす “, pp.4–5,142–169, サイエンス社 (2005).

[4] 米田 政明, 広瀬 貞樹, 大里 延康, 大川 知: “オートマトン・言語理論の基礎 “, pp.60,85–103, 近代科学社 (2003).

[5] 森田 憲一: “可逆計算 ナチュラルコンピューティングシリーズ Vo5“, pp.13–22, 近代科学社 (2012).

[6] Axelsen, H. B. and Gluck, R.: What Do Reversible Programs Compute.

[7] Gluck, R. and Yokoyama, T.: A Linear-Time Self-Interpreter of a Reversible Imperative Lan guage.

[8] Yokoyama, T., Axelsen, H. B. and Gluck,R.: Towards a Reversible Functional Language.

[9] Jones, N. D.: Computability and Complexity: From a Programming Perspective, MIT Press (1997).