可逆計算 講義ノート (各回のものをまとめたもの．準備中)

横山 哲郎

2017 年 1 月 5 日

目次

1 はじめに

スマートフォンは，デスクトップパソコンよりも少ないエネルギーで計算を行っている．朝に充電して家を出発して夕方帰宅するまで一度も充電されなくても，電池に蓄えられたエネルギーによって計算が行われる場合も多い．スマートフォンに使われるプロセッサは，デスクトップパソコンに使われるよりも消費電力が少ないものが使われている．このような技術的な改良が行われているにも関わらず，スマートフォンユーザは，電池の持ちに関して強い不満を持っている．むしろ，稼働時間の長さがそのスマートフォンの商品の価値を評価する重要な指標の 1 つになっている．今後も，より少ないエネルギーで必要とされる計算を行えるように，改良が行われていくことであろう．

では理論的にはどこまで行き着くことができるのであろうか？すなわち，計算を行うのにどれくらいのエネルギーが必要なのだろうか？この疑問に答えるには，どのような計算を行うかやプロセッサのアーキテクチャ，さらにはトランジスタの特性などを解析する必要があるであろう．しかし，さらにこうした改良が行き着く先はどこであるのだろうか．すなわち，**ある計算を行うのに必要な最小のエネルギー量は？**

非量子的な情報を消去するなどの非可逆な演算は，必ず熱の発生を伴うと広く信じられていた．この原理は，IBM の研究者の名前にちなんでランダウアの原理と呼ばれる [43]．この原理によると，1 ビットの情報が消去されると $kT\ln 2$ ($\approx 3\times 10^{-21}$ J) の熱が散逸[*1] する．ただしここで k はボルツマン定数，T は室温を表す．ちなみに，この原理は，1949 年にフォン・ノイマンによって行われたイリノイ大学での講義でも考察されている [60, p.66]．古典力学系におけるランダウアの原理は実験的にも示されている [12]．一方，情報をエネルギーに変換できることも実験的に示されている [71][*2]．

消失する情報の量はビット数で表すことができる．情報理論では，情報量（エントロピー）は次のように定義される．システムの可能な状態を i，システムが状態 i にある確率を p_i とする．このときシステムの情報

[*1] 散逸: エネルギーが熱エネルギーに不可逆的に変化すること．
[*2] 量子計算における情報の消去は量子もつれによって熱力学の第二法則に違反することなく冷却（負の発熱）を引き起こすことがある [17]．

量は

$$H = -\sum_i p_i \log_2 p_i \tag{1.1}$$

である．ここで単位はビットである．例えば，2 状態が等しい確率で起こるシステムでは，情報量は $H = -(0.5\log_2(0.5) + 0.5\log_2(0.5)) = 1$ bit である．また，ある 1 つの状態が常に起こる 2 状態のシステムでは，情報量は $H = -(1\log_2(1) + 0\log_2(0)) = 0$ bit である[*3]．2 状態が等しい確率で起こるシステムがある 1 つの状態が常に起こるように変化すると 1 ビットの情報が消失する．

一方で，可逆な演算には，必要な最小のエネルギーというものが存在しない．つまり，可逆な演算のみを行っているならば，コンピュータ中の 1 ステップあたりのエネルギー消費はいくらでも小さくすることができる．初期の可逆計算の研究動機のひとつは，エネルギー消費の少ない計算機を実現することであった．

- 運動の第 2 法則 $m\frac{\mathrm{d}^2\vec{x}}{\mathrm{d}t^2} = \vec{F}$ は，時間に対して可逆である（時間の逆方向にも同じ物理法則が成り立っている）．実際，$m\frac{\mathrm{d}^2\vec{x}}{\mathrm{d}(-t)^2} = m\frac{\mathrm{d}^2\vec{x}}{\mathrm{d}t^2}$ である．また，量子力学が支配するような微視的な世界では物理法則は可逆的である．
- 化学反応は，原理的には，可逆である．
- まだ充分にはわかっていない問題がたくさんある．
- 2009 年から可逆計算の国際会議が毎年開かれている[*4]．
- 可逆計算は量子計算の特別なケースと見なすことができる．
- DNA 重合反応は 1 ステップあたり 20–100kT のエネルギー消費が必要とされる [8]．
- 22 nm ノードの CMOS 論理セルのスイッチに必要な消費エネルギーは $10^5 kT$ に近づいている [77, 孫引き]

1.1 ノート

日本語では，次のような幅広い読者に向けた可逆計算の解説記事がある．

- 『計算の物理的な限界はあるか?』[10]
  - 計算物理と可逆計算について概略の解説記事．原著は Scientific American という一般向けの科学雑誌に，日本語版はサイエンス（日経サイエンスの 1990 年 9 月以前の誌名）に掲載された．
  - キーワード：フレドキンゲート[*5]，ビリヤード・ボール・コンピュータ，量子可逆コンピュータ，可逆チューリング機械（仮想的な酵素型，ブラウン運動型時計じかけ），RNA ポリメラーゼ，計算への物理的制限
- 『可逆コンピューティング—ビリヤードボールでコンピュータが作れるか？—』[89]
  - キーワード：ランダウアの原理，フレドキンゲート，トフォリゲート，万能可逆論理ゲート，ロータリー素子，ビリヤードボールモデル（BBM），可逆チューリング機械
- 『情報は物理過程だ』[44]
  - キーワード：マクスウェルの悪魔，時間変調ポテンシャル井戸スキーム，(計算の物理的限界)
- 『マクスウェルの悪魔現る』[11]

[*3] ただし，$0 \cdot \log_2 0 = 0$ とする．ちなみに，$\lim_{x\to 0} x\log x = 0$ である．
[*4] http://www.reversible-computation.org/
[*5] 記事中では，フレトキンゲートと訳されている．

  - 物理法則と情報に関する一般向けのエッセイ
  - 測定で得た情報量を考慮すれば，やはり熱力学の第二法則は破られていなかった．
  - 「マクスウェルの悪魔を作った東京大学の沙川貴大と上田正仁は 2010 年，ジャルジンスキー等式に悪魔が測定で得た情報量の項を付け加え，一般的な場合に拡張した。」
- 『多世界から生まれた計算機』[88]
  - 物理法則と計算に関する一般向けエッセイ，量子チューリング機械の誕生秘話も
  - 「情報とは，我々がこの世界について知っていることだけを意味するのではない。この世界を作っていると思われる」ホイーラー
  - 量子コンピュータは「多数の並行宇宙を使って計算する計算機」ドイチュ，エヴェレット解釈
  - 「計算の基本原理は物理ですよ」ベネット
- 『多数の宇宙で計算する 常識を揺るがすコンピュータ』[87]
  - 量子計算機の誕生秘話を語った一般向けエッセイ。ドイチュとエカートにも長期に渡って協力を受けて執筆したらしい。
  - キーワード: エヴァレット解釈（多世界解釈，並行宇宙論），おみくじ世界観，ポジティビズム（実証主義），リアリズム（実在主義），DJ(ドイチェ・ジョザ) のアルゴリズム，ショアのアルゴリズム
  - 「情報は物理的」「計算は物理過程」「熱を出さない計算機を作れるか？」IBM 研究者ロルフ・ランダウアー
  - 「けっきょく、自然は古典力学で動いていないから、古典的な計算では何をやってもうまくいかないのだ。やれやれ！　自然をシミュレートしたいなら、量子力学的にやった方が良い。」ノーベル賞学者リチャード・ファインマン
- 『シンギュラリティは近い―人類が生命を超越するとき』[38] には，以下のようにあるが,,,.
  - 「今後数十年間で、コンピューティングはリバーシブル・コンピューティングへと移行する … その結果、リバーシブル・コンピューティングはノンリバーシブル・コンピューティングに比べて、エネルギー需要を 10 億分の 1 に削減する可能性がある。…したがって、ナノテクノロジーが発展を極めた段階では、各ビットスイッチに必要なエネルギーは 1 兆分の 1 に削減されることになる。」

英語では，次のような幅広い読者に向けた可逆計算の解説記事などがある．文献 [7] は，可逆計算における先駆的な研究として知られている．

- “What Computing Is All About” の 11.2 節 “Reversible Computations” 以降 [74]
- ノーベル物理学賞の受賞者であるリチャード・P・ファインマンによる紹介『可逆計算と計算の熱力学』[22, 23]
- 可逆計算の分野を計算する動機，可逆論理回路，可逆論理回路のプロトタイプなど [30]
- デンマークのコペンハーゲン大学，ベルギーのゲント大学，補聴器の会社である Oticon による MicroPower プロジェクトの紹介 [5]
- 可逆論理回路 [62]

2　数学の復習

定義 1 (単射)**.** 写像 $f : X \rightharpoonup Y$ が**単射**とは，$a, b \in X$, $a \neq b$ なら $f(a) \neq f(b)$ であることである．

対偶をとると，写像 $f : X \rightharpoonup Y$ が単射とは，$a, b \in X$, $f(a) = f(b)$ なら $a = b$ であることである．単射な写像を **1 対 1** の写像とも呼ぶ．

定義 2 (半群)**.** 集合 G の任意の 2 元 a, b に対して a, b の積と呼ばれる G の元がただ 1 つ定まり（この元を ab と表す）次の性質を持つとき**半群**という [90, p.178,p.477].

1. 積に対して結合法則が成り立つ．すなわち，G の任意の元 a, b, c に対して $(ab)c = a(bc)$ が成り立つ．

自然数全体は加法に関して半群である．

定義 3 (群)**.** 集合 G の任意の 2 元 a, b に対して a, b の積と呼ばれる G の元がただ 1 つ定まり（この元を ab と表す）次の性質を持つとき**群**という [90, p.178].

1. 積に対して結合法則が成り立つ．すなわち，G の任意の元 a, b, c に対して $(ab)c = a(bc)$ が成り立つ．
2. 単位元が存在する．すなわち G のすべての元 a に対して $ae = ea = a$ となる元 e が存在する．e を**単位元**という．
3. 任意の元 a に対して逆元が存在する．G の元 a に対して $ab = ba = e$ となる G の元 b が存在するとき b を a の**逆元**といい，a^{-1} と表す．

定義 4 (自明群)**.** ただ 1 つの元からなる群を自明群と呼ぶ．

定義 5 (同値関係)**.** 集合 X が与えられたとき，次の性質を満たす X の 2 項関係 $\sim$ を同値関係といい，$x \sim y$ であるとき，x と y は同値であるという．

1. (反射律) すべての x に対して，$x \sim x$.
2. (対称律) $x \sim y$ であるとき，$y \sim x$.
3. (推移律) $x \sim y$, $y \sim z$ であるとき，$x \sim z$.

3 可逆チューリング機械

3.1 チューリング機械

1 テープ・チューリング機械（TM）は，以下の特徴をもつ：

- プッシュダウン・オートマトンと同様に有限状態による制御を行う．
- ひとつのテープをもつ．
  - テープはセルの列からなる．セルは両方向に無限個ある．
  - 各セルは必ず任意のテープ記号をひとつもつ．
  - 各テープ記号は有限個のセルにのみ含まれる．ただし，空白記号 b は無限個のセルに含まれても良い．
- ひとつのテープ・ヘッドをもつ．
  - テープ・ヘッドが必ずひとつのセルを示す．
  - テープ・ヘッドに示されたセルを読み書きすることができる．
  - テープ・ヘッドに示されたセルがテープ・ヘッドの移動に影響する．

- 状態およびテープ・ヘッドに示されたセルを変更していく．

定義 6 (1 テープ・チューリング機械)**.** **1 テープ・チューリング機械** T は，6 つ組

$$T = (Q, S, q_0, F, b, \delta) \tag{3.1}$$

によって定められる．ただし，

- Q: 制御部の状態の有限集合
- S: テープ記号の有限集合
- q_0 ($\in Q$): 初期状態
- F ($\subset Q$): 最終状態の有限集合
- b ($\in S$): 空白記号
- δ: $((Q\backslash F) \times (S \times S) \times Q) \cup ((Q\backslash F) \times \{\leftarrow, \downarrow, \rightarrow\} \times Q)$ の部分集合[*6]

である．

δ は遷移規則を表している．記号 $\leftarrow, \downarrow, \rightarrow$ は，ヘッドの移動方向を表し，それぞれ左への移動，不動，右への移動を意味する．遷移規則は 3 項組であり $(q, (s, s'), q')$ または (q, d, q') の形をしている（$q, q' \in Q, s, s' \in S, d \in \{\leftarrow, \downarrow, \rightarrow\}$）．前者は，状態 q で記号 s を呼んだ場合に記号 s' を上書きし，状態を q' にするという動作を表す．後者は，状態が q になった場合にヘッドを d の方向に動かし，状態を q' にするという動作を表す．遷移規則 δ に課された制約「$\backslash F$」は，最終状態から遷移することがないことを意味している．

定義 7 (様相)**.** 1 テープ・チューリング機械 $T = (Q, S, q_0, F, b, \delta)$ の様相とは，組 $(q, (l, s, r)) \in Q \times ((S\backslash\{b\})^* \times S \times (S\backslash\{b\})^*)$ である．

ここで，V^* は V の上の語（V 中の記号を 0 個以上並べたもの）の集合を表す．空語は ϵ と記す．様相 $(q, (l, s, r))$ において，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である．

定義 8 (計算ステップ)**.** 1 テープ・チューリング機械 $T = (Q, S, q_0, F, b, \delta)$ 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．ただし，ここで

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

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

[*6] チューリング機械の遷移規則は，5 項組みで定義されることが多い．しかし，可逆チューリング機械では，対称的な遷移規則を用いて前方実行と後方実行を統一的に理解しやすくなるので，本稿では 3 項組を用いることにする．なお，文献 [84, 56] を始めとする多くの文献で 4 項組が使われているので注意して欲しい．

定義 9 (1 テープ・チューリング機械の意味（その 1）)**.** 1 テープ・チューリング機械 $T = (Q, S, q_0, F, b, \delta)$ の意味 $\mathcal{T}_1$ とは,

$$\mathcal{T}_1[\![T]\!] = \{(r, c') \mid (q_0, (\epsilon, b, r)) \underset{T}{\Rightarrow}^* (q, c') \wedge q \in F\} \tag{3.2}$$

である.

例 1 (単進数のパリティのチェック)**.** 1 テープ・チューリング機械

$$T_{\text{parity}} = (\{q_0, q_{\text{even}}, q_{\text{odd}}, q_f\}, \{b, 0, 1\}, q_0, \{q_f\}, b, \delta_{\text{parity}}) \tag{3.3}$$

を考える．ただし，遷移規則 δ_{parity} は

$$\begin{aligned}\delta_{\text{parity}} = \{&(q_0, \rightarrow, q_{\text{even}}),\\ &(q_{\text{even}}, (0, b), q'_{\text{even}}), (q_{\text{even}}, (1, b), q'_{\text{odd}}), (q_{\text{even}}, (b, 0), q_f),\\ &(q'_{\text{even}}, \rightarrow, q_{\text{even}}),\\ &(q_{\text{odd}}, (0, b), q'_{\text{odd}}), (q_{\text{odd}}, (1, b), q'_{\text{even}}), (q_{\text{odd}}, (b, 1), q_f),\\ &(q'_{\text{odd}}, \rightarrow, q_{\text{odd}})\}\end{aligned}$$

である.

$\mathcal{T}_1[\![T_{\text{parity}}]\!](101) = (\epsilon, 0, \epsilon)$ である．なぜなら,

$$\begin{aligned}(q_0, (\epsilon, b, 101)) &\underset{T_{\text{parity}}}{\Longrightarrow} (q_{\text{even}}, (\epsilon, 1, 01))\\ &\underset{T_{\text{parity}}}{\Longrightarrow} (q'_{\text{odd}}, (\epsilon, b, 01))\\ &\underset{T_{\text{parity}}}{\Longrightarrow} (q_{\text{odd}}, (\epsilon, 0, 1))\\ &\underset{T_{\text{parity}}}{\Longrightarrow} (q'_{\text{odd}}, (\epsilon, b, 1))\\ &\underset{T_{\text{parity}}}{\Longrightarrow} (q_{\text{odd}}, (\epsilon, 1, \epsilon))\\ &\underset{T_{\text{parity}}}{\Longrightarrow} (q'_{\text{even}}, (\epsilon, b, \epsilon))\\ &\underset{T_{\text{parity}}}{\Longrightarrow} (q_{\text{even}}, (\epsilon, b, \epsilon))\\ &\underset{T_{\text{parity}}}{\Longrightarrow} (q_f, (\epsilon, 0, \epsilon))\end{aligned}$$

だからである.

例 2. 1 テープ・チューリング機械 $T_{\text{clear}} = (\{q_0, q_1, q_2, q_3, q_4, q_f\}, \{b, 0, 1\}, q_0, \{q_f\}, b, \delta_{\text{clear}})$ を考える．ただし，遷移規則 δ_{clear} は

$$\begin{aligned}\delta_{\text{clear}} = \{&(q_0, \rightarrow, q_1),\\ &(q_1, (0, b), q_2), (q_1, (1, b), q_3),\\ &(q_2, \rightarrow, q_1),\\ &(q_3, \rightarrow, q_4),\\ &(q_4, (0, b), q_3), (q_4, (1, b), q_3), (q_4, (b, b), q_f)\}\end{aligned}$$

である.

$\mathcal{T}_1[\![T_{\mathrm{clear}}]\!](011) = (\epsilon, b, \epsilon)$ である．なぜなら，

$$\begin{aligned}(q_0,(\epsilon,b,011)) &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_1,(\epsilon,0,11))\\ &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_2,(\epsilon,b,11))\\ &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_1,(\epsilon,1,1))\\ &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_3,(\epsilon,b,1))\\ &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_4,(\epsilon,1,\epsilon))\\ &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_3,(\epsilon,b,\epsilon))\\ &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_4,(\epsilon,b,\epsilon))\\ &\underset{T_{\mathrm{clear}}}{\Longrightarrow} (q_f,(\epsilon,b,\epsilon))\end{aligned}$$

だからである．一方，$\mathcal{T}_1[\![T_{\mathrm{clear}}]\!](000)$ は定義されない．すなわち，$\mathcal{T}_1[\![T_{\mathrm{clear}}]\!](000) = \perp$ である．T_{clear} は 1 が含まれているときのみ停止して空な記号列を返す．

問題 1. T_{clear} が停止する入力を正規表現で表せ．

例 3. $T_2 = (\{q_0\}, \{b\}, q_0, \{\}, b, \{\})$ は，チューリング機械である．
$[\![T_2]\!] = \{\}$ である．これは，どのような入力に対して定義されないことを意味する．

例 4 (0 と 1 の反転（非可逆版）)**.** $T_3 = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_f\}, \{b, 0, 1\}, q_0, \{q_f\}, b, \delta_1)$ は，チューリング機械である．ただし，遷移規則 δ_3 は

$$\begin{aligned}\delta_3 = \{&(q_0, \rightarrow, q_1),\\ &(q_1, (0,1), q_2), (q_1, (1,0), q_2), (q_1, (b,b), q_f),\\ &(q_2, \rightarrow, q_1)\}\end{aligned}$$

である．

3.2 可逆チューリング機械

記号列を別の記号列に変換する単射関数をチューリング機械で表すことを考える．
出力に状態を含むことができないので，受理状態を一意にしたい．

補題 1 (唯一の最終状態をもつ 1 テープ・チューリング機械)**.** 任意の 1 テープ・チューリング機械 $T = (Q, S, q_0, F, b, \delta)$ に対して，ある Q', δ' が存在して最終状態の有限集合 F $(\subseteq Q)$ を $q_f \notin Q$ である唯一の要素からなる集合 $\{q_f\}$ とする同じ意味をもつ 1 テープ・チューリング機械 $T' = (Q', S, q_0, \{q_f\}, b, \delta')$ が構成できる．

証明.（記号を書き換えたりヘッドを移動することなく）F の要素から q_f に遷移する規則を追加することで，T から T' を構成できる．すなわち，

$$\delta' = \delta \cup \bigcup_{q \in F} \{(q, \downarrow, q_f)\} \tag{3.4}$$

とおくと，

$$T' = (Q \cup \{q_f\}, S, q_0, \{q_f\}, \delta') \tag{3.5}$$

が構成できる．ここで，Q が q_f を含まないので，δ' は遷移規則の条件を満たしている． □

定義 10 (1 テープ・チューリング機械の意味（その 2）)**.** 1 テープ・チューリング機械 $T = (Q, S, q_0, \{q_f\}, b, \delta)$ の意味とは,

$$[\![T]\!] = \{(r, r') \in ((S\backslash\{b\})^* \times (S\backslash\{b\})^*) \mid (q_0, (\epsilon, b, r)) \underset{T}{\Rightarrow}^* (q_f, (\epsilon, b, r'))\} \tag{3.6}$$

である.

定義 11. 1 テープ・チューリング機械 $T = (Q, S, q_0, F, b, \delta)$ 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, **局所的に前方決定的**という. 1 テープ・チューリング機械 $T = (Q, S, q_0, F, b, \delta)$ 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**という.

補題 2 ([3])**.** チューリング機械 T が局所的に前方決定的で局所的に後方決定的である場合, $\cdot \underset{T}{\Rightarrow}^n \cdot$ は様相の間の単射関数である.

証明. $(q_1, (l_1, s_1, r_1)) \underset{T}{\Rightarrow} (q_2, (l_2, s_2, r_2))$ かつ $(q_1, (l_1, s_1, r_1)) \underset{T}{\Rightarrow} (q_3, (l_3, s_3, r_3))$ であったとする. それぞれの遷移は遷移規則 (q_1, a_1, q_2) と (q_1, a_2, q_3) によってそれぞれ起こったはずである. T が局所的に前方決定的であるので, この 2 つの遷移規則は同一である. したがって, $\cdot \underset{T}{\Rightarrow}^* \cdot$ は単射である.

同様にして, T が局所的に後方決定的であることから, $\cdot \underset{T}{\Rightarrow}^* \cdot$ は関数であることが示せる. □

1 テープ・チューリング機械は, 遷移規則 δ を $(Q \times (S \times S)^k \times Q) \cup (Q \times \{\leftarrow, \downarrow, \rightarrow\}^k \times Q)$ の部分集合に置き換えることで, k テープに拡張できる. 例えば, 2 テープ・チューリング機械の遷移規則は, $(Q \times (S \times S) \times (S \times S) \times Q) \cup (Q \times \{\leftarrow, \downarrow, \rightarrow\} \times \{\leftarrow, \downarrow, \rightarrow\} \times Q)$ の部分集合となる.

定義 12. 局所的に前方決定的かつ後方決定的で, 最終状態がひとつ（$Q = \{q_f\}$）で, 遷移規則の集合 $\delta : ((Q\backslash\{q_f\}) \times (S \times S) \times (Q\backslash\{q_0\})) \cup ((Q\backslash\{q_f\}) \times \{\leftarrow, \downarrow, \rightarrow\} \times (Q\backslash\{q_0\}))$ をもつチューリング機械 $T = (Q, S, q_0, \{q_f\}, b, \delta)$ は**可逆**という.

可逆チューリング機械を略して RTM(Reversible Turing Machine) と呼ぶ.

例 5. チューリング機械 $T = (\{q_0, q_1, q_2, q_f\}, \{b, 0, 1\}, q_0, \{q_f\}, b, \delta)$ を考える. ただし, $\delta = \{(q_0, \rightarrow, q_1), (q_1, 0, 0, q_2), (q_1, 1, b, q_0), (q_2, \leftarrow, q_f)\}$ である. T は, 局所的に前方決定的で局所的に後方決定的であるが, 可逆でない. $[\![T]\!]$ は単射でない. たとえば, $[\![T]\!](0) = 0$, $[\![T]\!](10) = 0$ である.

定理 13 ([3])**.** T が RTM ならば, $[\![T]\!]$ は単射関数である.

証明. 単射関数の合成は単射関数であることから, $\cdot \underset{T}{\Rightarrow} \cdot$ は単射である. 数学的帰納法より任意の自然数 n に対して $(q_0, (\epsilon, b, \cdot)) \underset{T}{\Rightarrow}^n (q_f, (\epsilon, b, \cdot))$ は単射関数である. δ の型の制限のために, q_f から遷移することはなく, q_0 に遷移することはないので, $(q_0, (\epsilon, b, \cdot)) \underset{T}{\Rightarrow}^* (q_f, (\epsilon, b, \cdot))$ は単射関数である. □

例 6 (0 と 1 の反転（可逆版）)**.** $T_{\text{flip}} = (\{q_0, q_1, q_2, q_3, q_4, q_f\}, \{b, 0, 1\}, q_0, \{q_f\}, b, \delta_{\text{flip}})$ は, 可逆チューリン

グ機械である．ただし，遷移規則 δ_3 は

$$\begin{aligned}\delta_{\text{flip}} = \{&(q_0, \rightarrow, q_1),\\ &(q_1, (0,1), q_2), (q_1, (1,0), q_2), (q_1, (b,b), q_3),\\ &(q_2, \rightarrow, q_1),\\ &(q_3, \leftarrow, q_4),\\ &(q_4, (0,0), q_3), (q_4, (1,1), q_3), (q_4, (b,b), q_f)\}\end{aligned}$$

である．

(TODO: 以下をちゃんと書く．)

定理 14 ([7, Bennett 1973])**.** 任意の 1 テープ・チューリング機械 S に対して，次のような 3 テープ可逆チューリング機械 R がある．I と P が空白記号を含まないような，S のアルファベット上の文字列ならば，S は I で停止するときまたそのときに限って R が $(I; B; B)$ で停止し，$S : I \to P$ であるときまたそのときに限って $R : (I; B; B) \to (I; B; P)$ である．

3.3 チューリング機械の別定義

以下は，文献 [32] によるチューリング機械の定義である．

定義 15 (1 テープ・チューリング機械の定義（その 2）)**.** **1 テープ・チューリング機械** M は，7 つ組

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \tag{3.7}$$

によって定められる．ただし，

- Q: 有限制御部の状態の有限集合
- Σ: 入力記号の集合
- Γ: テープ記号の有限集合（$\Sigma \subseteq \Gamma$）
- δ: 遷移関数 $(q, X) \mapsto (p, Y, D)$
  - $q \in Q$
  - $X \in \Gamma$
  - $p \in Q$: 次の状態
  - $Y \in \Gamma$:
  - $D \in \{L, R\}$: ヘッドが動く方向
- q_0 $(\in Q)$: 初期状態
- B $(\in \Gamma \backslash \Sigma)$: 空白記号．初期セルは，有限個を除いて空白記号をもつ．
- F $(\subseteq Q)$: 最終状態[*7] の有限集合

である．

定義 16 (時点表示（ID: instantaneous Description）)**.** 文字列

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \tag{3.8}$$

[*7] 最終状態は受理状態と呼ばれることもある．

で時点表示を表す．ただし，

1. q はチューリング機械の状態である．
2. 左から i 番目の記号をテープ・ヘッドが読み込んでいる．
3. $X_1X_2\cdots X_{i-1}X_iX_{i+1}\cdots X_n$ は，空白以外をもつすべてのセルを並べたテープの部分である．ただし，ヘッドが空白以外のセルの並びの左端よりも左にあったり，もしくは右端よりも右にあったりした場合は，接頭辞もしくは接尾辞が空白になり，i は 1 もしくは n にそれぞれなる．

である．

定義 17 (遷移)**.** $\delta(q, X_i) = (p, Y, L)$ である場合，

$$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash_M X_1X_2\cdots X_{i-2}pX_{i-1}YX_{i+1}\cdots X_n \tag{3.9}$$

である．ただし，$i = 1$ の場合，

$$qX_1X_2\cdots X_n \vdash_M pBYX_2\cdots X_n \tag{3.10}$$

であり，$i = n$ かつ $Y = B$ の場合，

$$X_1X_2\cdots X_{n-1}qX_n \vdash_M X_1X_2\cdots X_{n-2}pX_n \tag{3.11}$$

である．

$\delta(q, X_i) = (p, Y, R)$ である場合，

$$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash_M X_1X_2\cdots X_ipX_{i+1}\cdots X_n \tag{3.12}$$

である．ただし，$i = n$ の場合，

$$X_1X_2\cdots qX_n \vdash_M X_1X_2\cdots X_{n-1}YpB \tag{3.13}$$

であり，$i = 1$ かつ $Y = B$ の場合，

$$qX_1X_2\cdots X_n \vdash_M pX_2\cdots X_n \tag{3.14}$$

である．

3.4 ノート

- チューリング機械の日本語による解説 [23, 3-4 節].
- チューリング機械と可逆チューリング機械の日本語による解説 [10, p.109–114].
- チューリング機械は，1936 年にチューリングによって考案された [72].
- 可逆チューリング機械は，1963 年に Lecerf によって考察がされ [25]，また独立して 1973 年に Bennett によって定式化された [7].
- 1 テープ 3 記号可逆チューリング機械をクリーンに可逆模倣する 3 テープ RTM を，可逆埋込み（reversibilization）を用いずに構成する方法が知られている [2]．この 3 テープ RTM は，RTM 万能であり，模倣にはプログラム依存な漸近的オーバヘッドのみが必要である．
- 任意の TM を 1 テープ 2 記号 RTM で（非クリーンに）模倣できる [58].

- RTM が計算できるのは，すべての単射な計算可能関数である [3].
- RTM の日本語での定義は，[84][56, p.10, 定義 2.1] を参考.
- 記号の数と状態数が少ない RTM は [57] を参考.

TBA:

- 例：単進数を 2 倍
- 時間複雑度，空間複雑度

■用語の注意　表 1のような用語の違いに注意すること.

表 1　用語の違い

本稿	前方決定的	後方決定的	可逆的
Bennett[7]	deterministic	reversible	deterministic and reversible
森田 [86]	決定的	可逆的	決定的かつ可逆的

4　可逆決定性有限オートマトン

定義 18. 有限オートマトン（FA: Finite Automaton）とは，5 項組 $M = (Q, \Sigma, \delta, S, F)$ である。ただし、

- Q: 内部状態の有限集合
- Σ: 入力記号の有限集合
- S: 初期状態の集合
- $F \subseteq S$: 受理状態の集合
- $\delta : Q \times \Sigma \times Q$: 辺の集合

である。辺 (q, s, q') は $q \xrightarrow{a} q'$ とも記す．M の経路は，連続する辺の有限列である:

$$p = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ \cdots \ q_{n-1} \xrightarrow{a_n} q_n \tag{4.1}$$

定義 19. DFA $M = (S, \Sigma, \delta, s_0, F)$ は，任意の $x \in \Sigma$ について δ_x が単射であるとき，可逆という。ただし，$\delta_x(s) = \delta(s, x)$ とする。

4.1　ノート

- サーベイ [55]
- 日本語の入門 [85]
- Pin [67] (preliminary version [66]): reversible deterministic finite-state acceptors (DFAs) は DFA より表現力が低い.
- Reversible pushdown automata [40, 41]
- Queue automata [42] （手に入れていない）
- 関係ありそうな論文 [39]

5 可逆セルオートマトン

定理 20 ([86, 5.4 節]). 任意に与えられた k 次元 CA $A = (\mathbb{Z}^k, Q, N, f)$ に対して，A をシミュレートする $k+1$ 次元可逆 CA $A' = (\mathbb{Z}^{k+1}, Q', N', f')$ が存在する．

これは，可逆埋込みのアプローチ．

5.1 ノート

- オートマトンの創始 [60, 要チェック]
- 入門 [86, 5 章]

6 可逆プログラミング言語

- Janus[49, 82, 78], R, PISA, Bob [68], SyReC, ψ-Lisp, Theseus, SRL, RL, $\mathcal{MOQA}$, RFUN [81, 4], RCFUN [54], ...
- 可逆流れ図言語 [79]

6.1 Janus

Janus は，カルフォルニア工科大学の授業で作られた [49]．その後，ずいぶんたって形式化された [82, 78]．Janus 解釈系は，MicroPower プロジェクトのウェブページ上で実行することができる[*8]．

- Janus の部分計算 [52, 53]
- 部分計算の入門書 [33]
- 部分計算の参考ページ
  http://www.diku.dk/OLD/undervisning/2003e/235/literature.html

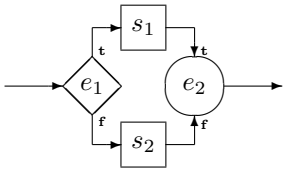
6.2 SRL

■LOOP($\mathbb{N}$)　まずは，非可逆の LOOP($\mathbb{N}$) を定義する．構文は

$$
\begin{array}{rcll}
r & ::= & r_0 \mid r_1 \mid \cdots & \text{Registers} \\
P & ::= & \texttt{INC}\ r & \text{Increment} \\
 & \mid & \texttt{DEC}\ r & \text{Decrement} \\
 & \mid & \texttt{FOR}\ r\ \{P\} & \text{Loop} \\
 & \mid & P_1; P_2 & \text{Composition}
\end{array}
$$

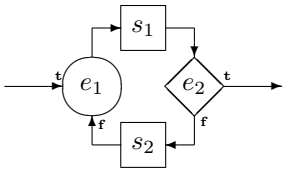
である．

[*8] http://topps.diku.dk/pirc/?id=janusP



`if` e_1
`then` s_1
`else` s_2
`fi` e_2

(a) 条件



`from` e_1
`do` s_1
`loop` s_2
`until` e_2

(b) ループ

図 1　可逆構造化制御流

次のような表示的意味論を与える:

$$
\begin{array}{ll}
[\![\cdot]\!] & : \mathrm{LOOP}(\mathbb{N}) \to \mathbb{N}^k \to \mathbb{N} \\
[\![P]\!](v_1, \ldots, v_k) & = \mathcal{P}[\![P]\!](\sigma_0[r_1 \mapsto v_1, \ldots, r_k \mapsto v_k])(r_0)
\end{array}
$$

$$
\begin{array}{ll}
\mathcal{P}[\![\cdot]\!] & : \mathrm{LOOP}(\mathbb{N}) \to (\mathrm{Registers} \to \mathbb{N}) \to \mathrm{Registers} \to \mathbb{N} \\
\mathcal{P}[\![\texttt{INC}\ r]\!]\sigma & = \sigma[r \mapsto \sigma(r) + 1] \\
\mathcal{P}[\![\texttt{DEC}\ r]\!]\sigma & = \sigma[r \mapsto \sigma(r) \dot{-} 1] \\
\mathcal{P}[\![\texttt{FOR}\ r\ \{P\}]\!]\sigma & = \mathcal{P}[\![P]\!]^{\sigma(r)}\sigma \qquad \text{if } \sigma(r) \geqslant 0 \\
\mathcal{P}[\![P_1; P_2]\!]\sigma & = \mathcal{P}[\![P_2]\!](\mathcal{P}[\![P_1]\!]\sigma)
\end{array}
$$

ここで,

$$
m \dot{-} n = \begin{cases} m - n & \text{if } m \geqslant n \\ 0 & \text{if } m < n \end{cases}
$$

である．入力は register $r_1, \ldots, r_k$ に格納され，出力は register r_0 から行われる．

LOOP($\mathbb{N}$) は以下のような特徴をもつ.

- LOOP($\mathbb{N}$) は全域的である.
- LOOP($\mathbb{N}$) プログラムで表現される関数のクラスは，原始帰納関数のクラスに一致する.
- LOOP($\mathbb{N}$) プログラムは単射ではない．もし，命令 `DEC` r の実行後の register r の値が 0 であるとき，直前の r の値が 0 および 1 のいずれであるか一意に定まらない．これは，$1 \dot{-} 1 = 0 \dot{-} 1 = 0$ であるためである.

■SRL($\mathbb{Z}$)　構文は LOOP($\mathbb{N}$) と同じである．ループ `FOR` r $\{P\}$ において，ループカウンタ r は P 中ではループカウンタとしてのみ出現する.

構文領域

$prog \in \mathrm{Progs}$	$s \in \mathrm{Stms}$	$d \in \mathrm{Vdecs}$	$\odot \in \mathrm{ModOps}$
$p \in \mathrm{Procs}$	$e \in \mathrm{Exps}$	$t \in \mathrm{Types}$	$\otimes \in \mathrm{Ops}$
$q \in \mathrm{PIds}$	$x \in \mathrm{Vars}$	$c \in \mathrm{Cons}$	

文法

$prog$	::= $p_{main}\ p^*$	Janus プログラム
d	::= $x \mid x\texttt{[}c\texttt{]}$	スカラと配列
t	::= `int` $\mid$ `stack`	データ型
p_{main}	::= `procedure main () (int` $d \mid$ `stack` $x)^*\ s$	main プロシージャ
p	::= `procedure` q`(`$t\ x$`,...,`$t\ x$`)` s	プロシージャ定義
s	::= $x\ \odot$`=` $e \mid x\texttt{[}e\texttt{]}\ \odot$`=` e	代入
	`if` e `then` s `else` s `fi` $e \mid$	条件
	`from` e `do` s `loop` s `until` $e \mid$	ループ
	`push(`x`,`x`)` $\mid$ `pop(`x`,`x`)` $\mid$	スタック操作
	`local` $t\ x$ `=` $e\ \ s\ \ $ `delocal` $t\ x$ `=` $e \mid$	局所変数ブロック
	`call` q`(`x`,...,`x`)` $\mid$ `uncall` q`(`x`,...,`x`)` $\mid$	プロシージャ呼び出し
	`skip` $\mid s\ s$	文の列
e	::= $c \mid x \mid x\texttt{[}e\texttt{]} \mid e \otimes e \mid$ `empty(`x`)` $\mid$ `top(`x`)` $\mid$ `nil`	式
c	::= `-2147483648` $\mid \cdots \mid$ `0` $\mid$ `1` $\mid \cdots \mid$ `2147483647`	整数定数 (-2^{31} から $2^{31}-1$)
$\odot$	::= `+` $\mid$ `-` $\mid$ `^`	演算子
$\otimes$	::= $\odot \mid$ `*` $\mid$ `/` $\mid$ `%` $\mid$ `&` $\mid$ `|` $\mid$ `&&` $\mid$ `||` $\mid$ `<` $\mid$ `>` $\mid$ `=` $\mid$ `!=` $\mid$ `<=` $\mid$ `>=`	演算子

図 2　Janus の構文

$v \in \mathrm{Vals}$	$=$	$\mathbb{Z}_{32} \cup \mathrm{Stack}_{\mathbb{Z}}$	値
$l \in \mathrm{Lvals}$	$=$	$\{\ \texttt{a}, \texttt{b}, \ldots, \texttt{a[0]}, \texttt{a[1]}, \ldots, \texttt{b[0]}, \ldots\ \}$	左辺値
$\sigma \in \mathrm{Stores}$	$=$	$\mathrm{Lvals} \rightharpoonup \mathrm{Vals}$	ストア
$\Gamma \in \mathrm{Pmaps}$	$=$	$\mathrm{PIds} \rightharpoonup \mathrm{Procs}$	プロシージャ環境

図 3　意味値

次のような表示的意味論を与える:

$$
\begin{aligned}
\mathcal{P}[\![\cdot]\!] &: \mathrm{SRL}(\mathbb{Z}) \to (\mathrm{Registers} \to \mathbb{Z}) \to \mathrm{Registers} \to \mathbb{Z} \\
\mathcal{P}[\![\texttt{INC}\ r]\!]\sigma &= \sigma[r \mapsto \sigma(r) + 1] \\
\mathcal{P}[\![\texttt{DEC}\ r]\!]\sigma &= \sigma[r \mapsto \sigma(r) - 1] \\
\mathcal{P}[\![\texttt{FOR}\ r\ \{P\}]\!]\sigma &= \begin{cases} \mathcal{P}[\![P]\!]^{\sigma(r)}\sigma & \text{if } \sigma(r) \geqslant 0 \\ \mathcal{P}[\![\mathcal{I}[\![P]\!]]\!]^{-\sigma(r)}\sigma & \text{if } \sigma(r) < 0 \end{cases} \\
\mathcal{P}[\![P_1; P_2]\!]\sigma &= \mathcal{P}[\![P_2]\!](\mathcal{P}[\![P_1]\!]\sigma)
\end{aligned}
$$

ここで，$\mathcal{I}[\![\cdot]\!]$ は，以下のような逆変換器である:

$$
\begin{aligned}
\mathcal{I}[\![\cdot]\!] &: \mathrm{SRL}(\mathbb{Z}) \to \mathrm{SRL}(\mathbb{Z}) \\
\mathcal{I}[\![\texttt{INC}\ r]\!] &= \texttt{DEC}\ r \\
\mathcal{I}[\![\texttt{DEC}\ r]\!] &= \texttt{INC}\ r \\
\mathcal{I}[\![\texttt{FOR}\ r\ \{P\}]\!] &= \texttt{FOR}\ r\ \{\mathcal{I}[\![P]\!]\} \\
\mathcal{I}[\![P_1; P_2]\!] &= \mathcal{I}[\![P_2]\!]; \mathcal{I}[\![P_1]\!]
\end{aligned}
$$

- ネストしたループを持たない $\mathrm{SRL}(\mathbb{N})$ プログラムによって実装される関数のクラスは，行列式 $+1$ をもつ行列 M によって表される線形変換 $f(\vec{x}) = M\vec{x} + C$ のクラスにちょうど一致する．

式の評価

$$\frac{}{\sigma \vdash_{expr} c \Rightarrow [\![c]\!]}\ \textsc{Con} \qquad \frac{}{\sigma \vdash_{expr} \texttt{nil} \Rightarrow nil}\ \textsc{Nil} \qquad \frac{}{\sigma \vdash_{expr} x \Rightarrow \sigma(x)}\ \textsc{Var} \qquad \frac{\sigma \vdash_{expr} e \Rightarrow v}{\sigma \vdash_{expr} x\texttt{[}e\texttt{]} \Rightarrow \sigma(x\texttt{[}v\texttt{]})}\ \textsc{Arr}$$

$$\frac{\sigma \vdash_{expr} e_1 \Rightarrow v_1 \quad \sigma \vdash_{expr} e_2 \Rightarrow v_2 \quad [\![\otimes]\!](v_1, v_2) = v}{\sigma \vdash_{expr} e_1 \otimes e_2 \Rightarrow v}\ \textsc{BinOp} \qquad \frac{}{\sigma[x \mapsto v_{hd} :: v_{tl}] \vdash_{expr} \texttt{top(}x\texttt{)} \Rightarrow v_{hd}}\ \textsc{Top}$$

$$\frac{}{\sigma[x \mapsto nil] \vdash_{expr} \texttt{empty(}x\texttt{)} \Rightarrow 1}\ \textsc{EmptyTrue} \qquad \frac{}{\sigma[x \mapsto v_{hd} :: v_{tl}] \vdash_{expr} \texttt{empty(}x\texttt{)} \Rightarrow 0}\ \textsc{EmptyFalse}$$

文の実行

$$\frac{\sigma \vdash_{expr} e \Rightarrow v \quad v_2 = [\![\odot]\!](v_1, v)}{\sigma[x \mapsto v_1] \vdash_{stmt} x \odot\texttt{=}\ e \Rightarrow \sigma[x \mapsto v_2]}\ \textsc{AssVar} \qquad \frac{\sigma \vdash_{expr} e_l \Rightarrow v_l \quad \sigma \vdash_{expr} e \Rightarrow v \quad v_2 = [\![\odot]\!](v_1, v)}{\sigma[x[v_l] \mapsto v_1] \vdash_{stmt} x\texttt{[}e_l\texttt{]} \odot\texttt{=}\ e \Rightarrow \sigma[x\texttt{[}v_l\texttt{]} \mapsto v_2]}\ \textsc{AssArr}$$

$$\frac{\sigma \vdash_{expr} e_1 \not\Rightarrow 0 \ \sigma \vdash_{stmt} s_1 \Rightarrow \sigma' \ \sigma' \vdash_{expr} e_2 \not\Rightarrow 0}{\sigma \vdash_{stmt} \texttt{if}\ e_1\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{fi}\ e_2 \Rightarrow \sigma'}\ \textsc{IfTrue} \qquad \frac{\sigma \vdash_{expr} e_1 \Rightarrow 0 \ \sigma \vdash_{stmt} s_2 \Rightarrow \sigma' \ \sigma' \vdash_{expr} e_2 \Rightarrow 0}{\sigma \vdash_{stmt} \texttt{if}\ e_1\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{fi}\ e_2 \Rightarrow \sigma'}\ \textsc{IfFalse}$$

$$\frac{\sigma \vdash_{expr} e_1 \not\Rightarrow 0 \quad \sigma \vdash_{stmt} s_1 \Rightarrow \sigma' \quad \sigma' \vdash_{loop} (e_1, s_1, s_2, e_2) \Rightarrow \sigma''}{\sigma \vdash_{stmt} \texttt{from}\ e_1\ \texttt{do}\ s_1\ \texttt{loop}\ s_2\ \texttt{until}\ e_2 \Rightarrow \sigma''}\ \textsc{LoopMain} \qquad \frac{\sigma \vdash_{expr} e_2 \not\Rightarrow 0}{\sigma \vdash_{loop} (e_1, s_1, s_2, e_2) \Rightarrow \sigma}\ \textsc{LoopBase}$$

$$\frac{\sigma \vdash_{expr} e_2 \Rightarrow 0 \quad \sigma \vdash_{stmt} s_2 \Rightarrow \sigma' \quad \sigma' \vdash_{expr} e_1 \Rightarrow 0 \quad \sigma' \vdash_{stmt} s_1 \Rightarrow \sigma'' \quad \sigma'' \vdash_{loop} (e_1, s_1, s_2, e_2) \Rightarrow \sigma'''}{\sigma \vdash_{loop} (e_1, s_1, s_2, e_2) \Rightarrow \sigma'''}\ \textsc{LoopRec}$$

$$\frac{}{\sigma[x \mapsto v_{hd}, x_s \mapsto v_{tl}] \vdash_{stmt} \texttt{push(}x\texttt{,}x_s\texttt{)} \Rightarrow \sigma[x \mapsto 0, x_s \mapsto v_{hd} :: v_{tl}]}\ \textsc{Push} \qquad \frac{\sigma' \vdash_{stmt} \texttt{push(}x\texttt{,}x_s\texttt{)} \Rightarrow \sigma}{\sigma \vdash_{stmt} \texttt{pop(}x\texttt{,}x_s\texttt{)} \Rightarrow \sigma'}\ \textsc{Pop}$$

$$\frac{\Gamma(q) \ = \ \texttt{procedure}\ q\texttt{(}t_1\ y_1\texttt{,}\ \ldots\texttt{,}t_n\ y_n\texttt{)}\ s \quad \sigma \vdash_{stmt} s[x_1/y_1, \ldots, x_n/y_n] \Rightarrow \sigma'}{\sigma \vdash_{stmt} \texttt{call}\ q\texttt{(}x_1\texttt{,}\ \ldots\texttt{,}x_n\texttt{)} \Rightarrow \sigma'}\ \textsc{Call} \qquad \frac{\sigma' \vdash_{stmt} \texttt{call}\ q\texttt{(}x_1\texttt{,}\ \ldots\texttt{,}x_n\texttt{)} \Rightarrow \sigma}{\sigma \vdash_{stmt} \texttt{uncall}\ q\texttt{(}x_1\texttt{,}\ \ldots\texttt{,}x_n\texttt{)} \Rightarrow \sigma'}\ \textsc{Uncall} \qquad \frac{}{\sigma \vdash_{stmt} \texttt{skip} \Rightarrow \sigma}\ \textsc{Skip}$$

$$\frac{\sigma \vdash_{stmt} s_1 \Rightarrow \sigma' \quad \sigma' \vdash_{stmt} s_2 \Rightarrow \sigma''}{\sigma \vdash_{stmt} s_1\ s_2 \Rightarrow \sigma''}\ \textsc{Seq} \qquad \frac{\begin{array}{c}\sigma \vdash_{expr} e \Rightarrow v \quad \sigma' \vdash_{expr} e' \Rightarrow v' \quad x_{new} \notin \sigma \cup \sigma' \\ \sigma[x_{new} \mapsto v] \vdash_{stmt} s[x_{new}/x] \Rightarrow \sigma'[x_{new} \mapsto v']\end{array}}{\sigma \vdash_{stmt} \texttt{local}\ t\ x\texttt{=}e\ \ s\ \ \texttt{delocal}\ t\ x\texttt{=}e' \Rightarrow \sigma'}\ \textsc{LocMem}$$

プログラムの実行

$$\frac{p_{main} \ = \ \texttt{procedure main()}\ t_1\ d_1 \cdots t_n\ d_n\ s \quad \Gamma = gen(p_1 \cdots p_k) \quad \{d_1 \mapsto init_{t_1}, \ldots, d_n \mapsto init_{t_n}\} \vdash^{\Gamma}_{stmt} s \Rightarrow \sigma}{\vdash_{prog} p_{main}\ p_1 \cdots p_k \Rightarrow \sigma}\ \textsc{Main}$$

図 4　Janus プログラムの操作的意味論

■ESRL($\mathbb{Z}$)　構文は

r	::=	$r_0 \mid r_1 \mid \cdots$	Registers
P	::=	INC r	Increment
	$\mid$	DEC r	Decrement
	$\mid$	$r \leftarrow -r$	Negate
	$\mid$	FOR r $\{P\}$	Loop
	$\mid$	$P_1; P_2$	Composition

である.

```
procedure main()
 ... RTM, tape and constants decl. and init. ...
 from  q=QS
 do    call inst(q,left,s,right,q1,s1,s2,q2,pc)
       pc += 1
       if pc=PC_MAX then
          pc ^= PC_MAX
       fi pc=0
 until q=QF

procedure inst(int q,stack left,int s,stack right,
               int q1,int s1,int s2,int q2,int pc)
 if q=q1[pc] then
   if s=s1[pc] then                  // Symbol rule:
     q += q2[pc]-q1[pc]              // set q to q2[pc]
     s += s2[pc]-s1[pc]              // set s to s2[pc]
   else
     if s1[pc]=SLASH then            // Shift rule:
       q += q2[pc]-q1[pc]            // set q to q2[pc]
       if s2[pc]=RIGHT then
         call   pushtape(s,left)     // push s on left
         uncall pushtape(s,right)    // pop right to s
       else
         if s2[pc]=LEFT then
           call   pushtape(s,right) // push s on right
           uncall pushtape(s,left)  // pop left to s
         fi s2[pc]=LEFT
       fi s2[pc]=RIGHT
     fi s1[pc]=SLASH
   fi s=s2[pc]
 fi q=q2[pc]

procedure pushtape(int s,stack stk)
 if empty(stk) && (s=BLANK) then
    s ^= BLANK  // zero-clear s
 else
    push(s,stk)
 fi empty(stk)
```

図 5　可逆チューリング機械解釈系

次のような表示的意味論を与える:

$$
\begin{array}{lcl}
\mathcal{P}[\![\cdot]\!] & : & \mathrm{ESRL}(\mathbb{Z}) \to (\mathrm{Registers} \to \mathbb{Z}) \to \mathrm{Registers} \to \mathbb{Z} \\
\mathcal{P}[\![\texttt{INC}\ r]\!]\sigma & = & \sigma[r \mapsto \sigma(r)+1] \\
\mathcal{P}[\![\texttt{DEC}\ r]\!]\sigma & = & \sigma[r \mapsto \sigma(r)-1] \\
\mathcal{P}[\![r \leftarrow -r]\!]\sigma & = & \sigma[r \mapsto -\sigma(r)] \\
\mathcal{P}[\![\texttt{FOR}\ r\ \{P\}]\!]\sigma & = & \begin{cases} \mathcal{P}[\![P]\!]^{\sigma(r)}\sigma & \text{if } \sigma(r) \geqslant 0 \\ \mathcal{P}[\![\mathcal{I}[\![P]\!]]\!]^{-\sigma(r)}\sigma & \text{if } \sigma(r) < 0 \end{cases} \\
\mathcal{P}[\![P_1; P_2]\!]\sigma & = & \mathcal{P}[\![P_2]\!](\mathcal{P}[\![P_1]\!]\sigma)
\end{array}
$$

ここで，$\mathcal{I}[\![\cdot]\!]$ は，以下のような逆変換器である:

$$
\begin{array}{lcl}
\mathcal{I}[\![\cdot]\!] & : & \mathrm{ESRL}(\mathbb{Z}) \to \mathrm{ESRL}(\mathbb{Z}) \\
\mathcal{I}[\![\texttt{INC}\ r]\!] & = & \texttt{DEC}\ r \\
\mathcal{I}[\![\texttt{DEC}\ r]\!] & = & \texttt{INC}\ r \\
\mathcal{I}[\![r \leftarrow -r]\!] & = & r \leftarrow -r \\
\mathcal{I}[\![\texttt{FOR}\ r\ \{P\}]\!] & = & \texttt{FOR}\ r\ \{\mathcal{I}[\![P]\!]\} \\
\mathcal{I}[\![P_1; P_2]\!] & = & \mathcal{I}[\![P_2]\!]; \mathcal{I}[\![P_1]\!]
\end{array}
$$

- ネストしたループを持たない ESRL($\mathbb{N}$) プログラムによって実装される関数のクラスは，行列式 ± 1 をもつ行列 M によって表される線形変換 $f(\vec{x}) = M\vec{x} + C$ のクラスにちょうど一致する．

6.3 翻訳

- 命令型可逆言語のクリーンな翻訳 [1]: 高水準可逆言語 Janus から可逆アセンブリ言語 PISA への翻訳器を実現した．

7 可逆回路

- 入門 “Circuits Built from Reversible Gates”[74, Section 11.3], 1993 年の時点では，1 ビットの演算に $10^9 kT$ のオーダの熱が散逸していたようである．
- Not ゲート，ファイマンゲート，フレドキンゲート [26]，トフォリゲート [69][70, 要チェック]
- 万能ゲート: フレドキンゲート
- 論理演算 1 つに $10^4 kT$ のオーダのエネルギーが必要 [77]

8 可逆プロセス代数

8.1 CCS

The Calculus of Communicating Systems

$$
\begin{array}{lrll}
\text{Actions:} & \alpha & ::= a \mid \bar{a} \mid \cdots & \text{Action on a channel} \\
& & \mid \tau & \text{Silent action} \\
\\
\text{Processes:} & P & ::= 0 & \text{End of process} \\
& & \mid \sum \alpha_i.P_i & \text{Guarded choice} \\
& & \mid (P \mid P) & \text{Fork} \\
& & \mid (a)P & \text{Restriction}
\end{array}
$$

Additive structured congruence:

$$P + 0 \equiv P \tag{8.1}$$

$$P_1 + P_2 \equiv P_2 + P_1 \tag{8.2}$$

$$(P_1 + P_2) + P_3 \equiv P_1 + (P_2 + P_3) \tag{8.3}$$

8.2 RCCS

本節では，可逆版の CCS である Reversible CCS (RCCS) [15] を紹介する．RCCS では，平行計算におけるバックトラックを扱うことができる．

8.2.1 構文

*9

$$
\begin{array}{lll}
\text{Memories:} & m ::= \langle\rangle & \text{Empty memory} \\
& \mid \langle 1\rangle \cdot m & \text{Left-Fork} \\
& \mid \langle 2\rangle \cdot m & \text{Right-Fork} \\
& \mid \langle *, \alpha, P\rangle \cdot m & \text{Semi-Synch} \\
& \mid \langle m, \alpha, P\rangle \cdot m & \text{Synch} \\
\\
\text{Monitored Processes:} & R ::= m \rhd P & \text{Threads} \\
& \mid (R \mid R) & \text{Product} \\
& \mid (a)R & \text{Restriction}
\end{array}
$$

Additional congruence:

$$m \rhd (P \mid Q) \equiv (\langle 1\rangle \cdot m \rhd P \mid \langle 2\rangle \cdot m \rhd Q) \tag{8.4}$$

$$m \rhd (a)P \equiv (a)m \rhd P \tag{8.5}$$

8.2.2 Structural congruence

2 つのプロセスは，式 8.4，式 8.5，および additive congruence identity で互いに移りあうとき structurally equivalent という．

定義 21. Coherence $\frown$ は，次の 2 条件を満たす最小の対称関係である．

1. $\forall i, j.\ i \neq j \Rightarrow \langle i\rangle \cdot m \frown \langle j\rangle \cdot m$
2. $\forall m_1, m_2.\ m \frown m' \Rightarrow m_1 \cdot m \frown m_2 \cdot m'$

定義 22. Monitored process は，その memories が互いに coherent ならば，coherent という．

補題 3. Structural congruence によって，coherence は保存される．

8.2.3 推移規則

$$R \xrightarrow{\mu:\zeta} S \tag{8.6}$$

ここで，R, S は monitored process, ζ は directed action, μ は identifier である．

$$
\begin{array}{lll}
\zeta & ::= & \alpha \mid \alpha_* \qquad \text{Directed Actions} \\
\mu & ::= & m \mid m, m \qquad\quad \text{Identifiers}
\end{array}
$$

*9 文献 [37] の記法の方が簡潔かもしれない。

8.2.4 Irreversible Actions

- $\mathcal{M}_R$: R 中に出現するメモリの集合
- $R \to^\star S$ となる S ならびに $m < m'$ となる $m \in \mathcal{M}_R$ および $m' \in \mathcal{M}_S$ が存在することを，R から始まるトレースが変更できるという．
- R から始まるトレースが変更できないメモリは，R 中で**ロックされた**という．

定義 23. L_R をロックされたすべてのメモリからなる $\mathcal{M}_R$ の部分集合とする．L_R は以下の 4 つを満たす．

1. $\langle\circ\rangle \cdot m \in \mathcal{M}_R \Rightarrow \langle\circ\rangle \cdot m \in \mathcal{L}_R$
2. $m \in \mathcal{L}_R, m' < m \Rightarrow m' \in \mathcal{L}_R$
3. $\langle m,\alpha,P\rangle \cdot m' \in \mathcal{L}_R \Rightarrow \langle m',\bar{\alpha},Q\rangle \cdot m \in \mathcal{L}_R$
4. $\langle i\rangle \cdot m \in \mathcal{L}_R \Rightarrow \langle j\rangle \cdot m \in \mathcal{L}_R$

例 7. Monitored process

$$R = \langle\rangle \rhd (x)(\bar{x}.0 \mid x.\underline{a}.P \mid x.\underline{b}.Q)$$

を考える．ここで，x は可逆な action，a, b を非可逆な action とする．以下の証明木：

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle 1\rangle \rhd \bar{x}.0 + 0 \xrightarrow{\langle 1\rangle:\bar{x}} \langle *,\bar{x},0\rangle\langle 1\rangle \rhd 0}\,\text{act}}{\langle 1\rangle \rhd \bar{x}.0 \equiv \langle 1\rangle \rhd \bar{x}.0 + 0 \xrightarrow{\langle 1\rangle:\bar{x}} \langle *,\bar{x},0\rangle\langle 1\rangle \rhd 0 \equiv \langle *,\bar{x},0\rangle\langle 1\rangle \rhd 0}= \quad \dfrac{\dfrac{}{\langle 2\rangle \rhd x.\underline{a}.P + 0 \xrightarrow{\langle 2\rangle:x} \langle *,x,0\rangle\langle 2\rangle \rhd \underline{a}.P}\,\text{act}}{\langle 2\rangle \rhd x.\underline{a}.P \equiv \langle 2\rangle \rhd x.\underline{a}.P + 0 \xrightarrow{\langle 2\rangle:x} \langle *,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \equiv \langle *,x,0\rangle\langle 2\rangle \rhd \underline{a}.P}=}{\langle 1\rangle \rhd \bar{x}.0 \mid \langle 2\rangle \rhd x.\underline{a}.P \xrightarrow{\langle 1\rangle,\langle 2\rangle:\tau} \langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P}\,\text{syn}}{\langle 1\rangle \rhd \bar{x}.0 \mid \langle 2\rangle \rhd x.\underline{a}.P \mid \langle 3\rangle \rhd x.\underline{b}.Q \xrightarrow{\langle 1\rangle,\langle 2\rangle:\tau} \langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \mid \langle 3\rangle \rhd x.\underline{b}.Q}\,\text{par-l}}{(x)(\langle 1\rangle \rhd \bar{x}.0 \mid \langle 2\rangle \rhd x.\underline{a}.P \mid \langle 3\rangle \rhd x.\underline{b}.Q) \xrightarrow{\langle 1\rangle,\langle 2\rangle:\tau} (x)(\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \mid \langle 3\rangle \rhd x.\underline{b}.Q)}\,\text{res}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \xrightarrow{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle:\underline{a}} \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd P}\,\text{commit}}{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \equiv \langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P + 0 \xrightarrow{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle:\underline{a}} \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd P \equiv \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd P}= \text{par-r}}{\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \xrightarrow{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle:\underline{a}} \langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd P}\,\text{par-l}}{\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \mid \langle 3\rangle \rhd x.\underline{b}.Q \xrightarrow{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle:\underline{a}} \langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd P \mid \langle 3\rangle \rhd x.\underline{b}.Q}\,\text{par-l}}{(x)(\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd \underline{a}.P \mid \langle 3\rangle \rhd x.\underline{b}.Q) \xrightarrow{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle:\underline{a}} (x)(\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd P \mid \langle 3\rangle \rhd x.\underline{b}.Q)}\,\text{res}$$

より，

$$R \;\to^\star\; (x)(\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \rhd P \mid \langle 3\rangle \rhd x.\underline{b}.Q) \quad =: R_a$$

が得られる．同様にして，

$$R \;\to^\star\; (x)(\langle\langle 3\rangle,\bar{x},0\rangle\langle 1\rangle \rhd 0 \mid \langle 2\rangle \rhd x.\underline{a}.P \mid \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 3\rangle \rhd x.\underline{b}.Q) \quad =: R_b$$

が得られる．R_a 中に出現するメモリの集合は

$$\mathcal{M}_{R_a} = \{\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle, \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle, \langle 3\rangle\}$$

であるので，1 より $\langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \in \mathcal{L}_{R_a}$ である．よって，2 より，$\{\langle\langle 1\rangle,x,0\rangle\langle 2\rangle, \langle 2\rangle\} \subseteq \mathcal{L}_{R_a}$ である．$\langle\langle 1\rangle,x,0\rangle\langle 2\rangle \in \mathcal{L}_{R_a}$ であるので，3 より，$\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle \in \mathcal{L}_{R_a}$ である．$\langle 2\rangle \in \mathcal{L}_{R_a}$ であるので，4 より，$\{\langle 1\rangle, \langle 3\rangle\} \subseteq \mathcal{L}_{R_a}$ である．これ以外にロックされたメモリは存在しない．まとめると，

$$\mathcal{L}_{R_a} = \{\langle\langle 2\rangle,\bar{x},0\rangle\langle 1\rangle, \langle 1\rangle, \langle\circ\rangle\langle\langle 1\rangle,x,0\rangle\langle 2\rangle, \langle\langle 1\rangle,x,0\rangle\langle 2\rangle, \langle 2\rangle, \langle 3\rangle\}$$

となる．

8.3 CCSK

$$\text{CCSK Processes:}\quad X, Y ::= P \qquad \text{CCS process}$$
$$\mid \alpha[i].X \mid X + Y \mid (X \mid Y) \mid X\backslash A$$

8.4 ノート

文献 [16]

- 分子生物学の現象をプロセス代数でモデル化した
- 4 章にあるように、ある程度の表現力がある
- 筆者の知る範囲では、CCS-R のような CCS は考えられてこなかった
- $a|b$ は，a, b の実行順序に依らず，両方実行して初期状態に戻ってこれるようになっている．

9 可逆コンビネータ論理

9.1 古典的コンビネータ論理

定義 24 (コンビネータ論理の項)**.** K と S を含む (有限/無限の) 集合および変数の無限集合の上のコンビネータ論理の項である **CL** 項は，次のように帰納的に定義される:

1. すべての変数と定数は **CL** 項である．
2. もし X と Y が **CL** 項ならば，(XY) は **CL** 項である．

例 8.

CL 項の例 $\mathsf{K}, \mathsf{S}, \mathsf{SKK}, x, \mathsf{S}xy$
CL 項でない例 $\lambda x.\ \mathsf{K}$ (λ 抽象は **CL** 項に含まれない)

構文上の等価性を記号 $\equiv$ で表す．

定義 25 (**CL** の簡約)**.** **CL** 項上の簡約関係 $\to$ は次の規則で定義される:

1. $\mathsf{K}XY \to X$
2. $\mathsf{W}XY \to XYY$
3. $\mathsf{C}XYZ \to XZY$
4. $\mathsf{B}XYZ \to X(YZ)$
5. $X \to X'$ implies $XY \to X'Y$
6. $X \to X'$ implies $YX \to YX'$

$\twoheadrightarrow$ は，$\to$ の反射推移閉包を意味する．$=$ は $\twoheadrightarrow$ を拡張した最小等価関係を意味する．外延性の規則

$$\forall x \notin FV(PP').\ Px = P'x \Rightarrow P = P' \tag{9.1}$$

を拡張した **CF** を **CF+ext** とする．

B コンビネータを用いて演算子

$$X \cdot Y = \mathsf{B}XY \tag{9.2}$$

を定義する．

$$\begin{aligned}(X \cdot Y)Z &= \mathsf{B}XYZ \quad \because \cdot \text{の定義} \\ &= X(YZ) \quad \because \mathsf{B} \text{の定義}\end{aligned}$$

$\cdot$ は **CL** 項の積である．したがって，**CL** 項全体は $\cdot$ に関して半群である．さらに，I コンビネータは，単位元である．

さらに任意の元に逆元が存在すれば **CL** 項全体が群になるところであるが，残念ながら逆元が存在しないような元が存在する．たとえば，K には逆元が存在しない．このことは背理法から言える．$\mathsf{I} = \mathsf{K} \cdot X$ となる X が存在したとする．このとき，任意の Y に対して $Y = \mathsf{I}Y = (\mathsf{K} \cdot X)Y = \mathsf{K}XY = X$ となる．しかし，これは Y が $Y \neq X$ であるとき成立しない．よって K コンビネータに逆元は存在しない．

CL 項全体の部分集合ならば，$\cdot$ に関して群であるようなものがあるだろうか．I は，自身の逆元である:

$$\mathsf{I} \cdot \mathsf{I} = \mathsf{BII} = \mathsf{I} \tag{9.3}$$

したがって，{I} は (自明) 群である．

また，

$$(\mathsf{C} \cdot \mathsf{C})XYZ = \mathsf{BCC}XYZ = \mathsf{C}(\mathsf{C}X)YZ = \mathsf{C}XZY = XYZ \tag{9.4}$$

であるので，外延性から C の逆元は C である．直観的には，C は，引数を入れ替えるコンビネータであり，2 回引数を入れ替えることで引数の順序を元に戻している．

9.2 可逆コンビネータ論理

定義 26 (**rCL** 項)**.** **CL** 項 M と履歴項 H からなる組 $\langle M \mid H \rangle$ を **rCL** 項と呼ぶ．ここで，履歴項 H は，

$$\begin{aligned}H &::= \epsilon \mid S : H \\ S &::= T\mathsf{K}_n^m \mid \mathsf{W}_n^m \mid \mathsf{B}_n^m \mid \mathsf{C}_n^m \mid \overline{S}\end{aligned}$$

である．ここで，T は古典的 **CL** 項，n, m は自然数である．

$H \equiv S_1 : S_2 : \cdots : S_n$ ならば，$\overline{H}$ で $\overline{S_n} : \overline{S_{n-1}} : \overline{1}$ を表すことにする．$\overline{\overline{H}}$ と H は同一視する．すなわち，$\overline{\overline{H}} = H$ とする．$\mathcal{H}$ は，この等式を法とする履歴項の集合を表す．積 : は，結合的であり，単位元 ϵ をもつとする．また，H の逆元は $\overline{H}$，すなわち $H : \overline{H} = \overline{H} : H = \epsilon$ とする．履歴項の全体は積 : に関して群をなす．

$$T\mathsf{K}_n^m + l \equiv T\mathsf{K}_{n+l}^m \tag{9.5}$$

$$S + l \equiv S \qquad \text{if } S \not\equiv T\mathsf{K}_n^m \tag{9.6}$$

$$H + l \equiv S_1 + l : S_2 + l : \cdots : S_k + l \tag{9.7}$$

$$len(X) = \begin{cases} 1 & X \text{ is a constant or variable} \\ n + m & \text{if } X = (YZ) \text{ with } len(Y) = n \text{ and } len(Z) = m \end{cases} \tag{9.8}$$

定義 27 (**rCL** における簡約)**.** 可逆簡約関係 $\rightarrowtail$ は次のように定義される:

前方規則

1. $\langle \mathsf{K}XY \mid \rangle \rightarrowtail \langle X \mid Y\mathsf{K}_0^{len(X)} \rangle$
2. $\langle \mathsf{W}XY \mid \rangle \rightarrowtail \langle XYY \mid \mathsf{W}_0^{len(X)} \rangle$
3. $\langle \mathsf{C}XY \mid \rangle \rightarrowtail \langle XZY \mid \mathsf{C}_0^{len(X)} \rangle$
4. $\langle \mathsf{B}XYZ \mid \rangle \rightarrowtail \langle X(YZ) \mid \mathsf{B}_0^{len(X)} \rangle$

後方規則

1. $\langle X \mid \rangle \rightarrowtail \langle \mathsf{K}XY \mid \overline{Y\mathsf{K}}_0^{len(X)} \rangle$
2. $\langle XYY \mid \rangle \rightarrowtail \langle \mathsf{W}XY \mid \overline{\mathsf{W}}_0^{len(X)} \rangle$
3. $\langle XZY \mid \rangle \rightarrowtail \langle \mathsf{C}XYZ \mid \mathsf{C}_0^{len(X)} \rangle$
4. $\langle X(YZ) \mid \rangle \rightarrowtail \langle \mathsf{B}XYZ \mid \mathsf{B}_0^{len(X)} \rangle$

構造規則

1. $\langle X \mid \rangle \rightarrowtail \langle X' \mid H' \rangle$ ならば $\langle XY \mid \rangle \rightarrowtail \langle X'Y \mid H' \rangle$
2. $\langle X \mid \rangle \rightarrowtail \langle X' \mid H' \rangle$ ならば $\langle YX \mid \rangle \rightarrowtail \langle YX' \mid H' + len(Y) \rangle$
3. $\langle X \mid \rangle \rightarrowtail \langle X' \mid H' \rangle$ ならば $\langle X \mid H \rangle \rightarrowtail \langle X' \mid H : H' \rangle$

関係 $\rightarrowtail$ は関数ではない．すなわち，**rCL** における簡約は，非決定的である．**後方簡約**を

$$\langle P_2 \mid H_2 \rangle \leftarrowtail \langle P_1 \mid H_1 \rangle \text{ iff } \langle P_1 \mid H_1 \rangle \rightarrowtail \langle P_2 \mid H_2 \rangle \tag{9.9}$$

とする．後方簡約は決定的である．これは，定義 27中の規則の右辺の履歴項が互いに重なり合わないことからいえる．

命題 1. 関係 $\leftarrowtail$ は部分関数である．

命題 2. $T = \langle M \mid H'$ と $T' = \langle N \mid H''$ を **rCL** 項とする．もし $T \twoheadrightarrow T'$ ならば，$H \in \mathcal{H}$ が存在して $H'' = H' : H$ である．

定理 28. すべての $M \in \mathbf{CL}$ に対して，もし $M \twoheadrightarrow N$ なら，すべての $H \in \mathcal{H}$ に対して，$H' \in \mathcal{H}$ が存在して，$\langle M \mid H \rangle \twoheadrightarrow \langle N|H' \rangle$ である．

逆は成り立たない．

補題 4. **CL** 項 X と $H \in \mathcal{H}$ が与えられたとする．

$$\langle X \mid \rangle \twoheadrightarrow \langle X' \mid H \rangle \text{ iff } \langle X' \mid \rangle \twoheadrightarrow \langle X \mid \overline{H} \rangle$$

命題 3. M を逆元が存在する **CL** 項とする．履歴項 $H \in \mathcal{H}$ と

$$\langle MN_1 \mid \rangle \twoheadrightarrow \langle N_2 \mid H \rangle$$

となる **CL** 項 N_1 と N_2 を仮定する．このとき，$H' \in \mathcal{H}$ が存在して

$$\langle M^{-1}N_2 \mid \rangle \twoheadrightarrow \langle N_1 \mid H' \rangle$$

である．

命題 4. すべての $H \in \mathcal{H}$ に対して，$\pi(H)$ は **rCL** 上の置換である．

命題 5. 射

$$\begin{aligned}\delta &: \mathcal{G}(\mathbf{rCL}, \mathcal{H}, \pi) \to \mathcal{G}(\mathbf{rCL}, \rightarrowtail\!\!\!\!\!\rightarrow) \\ \delta(\langle T, H, T' \rangle) &= \langle T, T' \rangle\end{aligned}$$

は，亜群同型射である．

9.3 ノート

文献 [18] では，前年に発表した操作的意味論を反映した表示的意味論の開発を目的として可逆版のコンビネータ論理を研究したとしている．

疑問: 逆元をもつような任意の M に対して，$\langle M^{-1}MN \mid \rangle \to^* \langle N \mid \rangle$ となる評価戦略 $\to^*$ はあるか．M^{-1} の形は様々だからなさそうな気がする．木構造でもてばいくらか履歴を同一視することはできそう．W も **CL** 項が一致したときに入れ替えを許せばできないか．

10 可逆模倣

- Bennett 1973 の可逆模倣 [7]: $T' = \mathrm{O}(T)$, $S' = \mathrm{O}(S + T)$
- Bennett 1989 の可逆模倣その 1 [9]
  - Bennett による解析: $T' = \mathrm{O}(T^{1+\epsilon})$, $S' = \mathrm{O}(S \log T)$
  - Levine と Sherman による解析: $T' = \Theta(T^{1+\epsilon}/S^{\epsilon})$, $S' = \Theta(S(1 + \ln(T/S)))$
- Bennett 1989 の可逆模倣その 2 [9]
  - Bennett による解析: $T' = \mathrm{O}(T)$, $S' = \mathrm{O}(ST^{\epsilon})$
- Bennett の可逆模倣の空間最適性 [47, 48]
- LMT アルゴリズム (Lange, McKenzie, and Tapp)[45]
  - 線形空間アルゴリズム

11 可逆化

- Probabilistic guarded-command language (pGCL) [83]
- RCC [13, 63, 65, 64]
- SEMCD machine [34]
- Fun to Inv [59]

12 可逆アルゴリズム

- 可逆アルゴリズムは，非単射関数のゴミ出力のある可逆シミュレーションと単射関数のクリーン可逆シミュレーションに分けられる．
- 整列法は，基本的なアルゴリズムのひとつであり非単射である．
- Lutz と Derby は，結果は正しくはないが，可逆バブルソートをプログラミングしようとした先駆者であった [49]．

- 長さ n の配列に対する整列結果に $[0 : n! - 1]$ の範囲の非負整数であるランクを出力に付与することでゴミ出力量が最適な可逆挿入整列法や可逆マージソートが提案された [80].
- ランクを付与することで単射になったクイックソートを $\mathcal{MOQA}$ 言語において（非可逆に）実現した [21].
- 文献 [64] の 15 章には，整列法の可逆シミュレーションが入力，置換，もしくはすべての制御流情報を記憶することで実現できることが議論されている.
- コンパクトで情報の追加と読み取りが効率的にできるようなゴミ情報の中間表現を用いて効率の良い挿入整列法，バブルソート，選択整列法などの比較ソートが実現でき，恒等置換のトリックを使って比較ソートのアルゴリズムに一般的に適用できる 2 パスの可逆シミュレーションがある [6].

13 量子計算

TBA

14 関連分野

- 撞球計算機（ビリヤード・ボール・コンピュータ）
  - Toffoli と Fredkin によって提案された [26].
  - 入門 [74, Sect. 11.4]
- DNA 計算機 [8]
  RNA ポリメラーゼは，1 秒間に約 30 個のヌクレオチドの速さで DNA 鎖を複製している．ヌクレオチドひとつあたり約 $20kT$ の熱を散逸している．誤りは 10000 個のヌクレオチドに対して 1 つ以下である．[74, p. 243]
- ロータリー素子 [86, 3 章，4 章]
- PISA [75], (RAM: Random Access Machine)
- 模倣
  - BBM で可逆論理ゲート [86, 3.4 節]
  - BBM でロータリー素子 [86, 3.6 節]
  - ロータリー素子で RTM[86, 4.2 節]
  - 単純な 2 次元可逆 CA で可逆論理回路 [86, 5.5 節]
- 可逆デバッグ
  - GDB and Reverse Debugging http://www.gnu.org/software/gdb/news/reversible.html
  - Commercial reversible debugger? http://chrononsystems.com/
  - OCaml: reverse execution http://caml.inria.fr/pub/docs/manual-ocaml/debugger.html
  - GDB: reverse debugging http://www.gnu.org/software/gdb/news/reversible.html

14.1 逆解釈

1956 年には，McCarthy がチューリング機械で与えられた関数の逆関数を計算する方法を考案していた [50]．生成と検査のアプローチ．

- T_m: m 番目のチューリング機械
- $f_m(n)$: T_m で計算される部分関数．n に対して T_m が停止しなかった場合は値が定義されない: $f_m(n) = \perp$ if T_m does not terminate.
- $f_m(g(m,r)) = r$ となるような関数 $g(m,r)$ を計算するチューリング機械を求める問題を考えた．
- $g(m,r)$ が存在しないときに，停止して 1 を表示するチューリング機械は存在しない．したがって，$g(m,r)$ の存在は決定不能である．
- $g(m,r)$ が存在するならば，$g(m,r)$ を計算するチューリング機械を構成することができる．
- 解法 1: $n = 0, 1, \ldots$ に対して，$f_m(n) = r$ となる n が見つかるまで $f_m(n)$ を順に計算していく．$f_m(n_1) = \perp$ となる n_1 があると別の n の候補を試すことができない．
- 解法 2: 以下の $f_m^k(n)$ を用いる．

$$f_m^k(n) = \begin{cases} (1, f_m(n)) & \text{if 計算が } k \text{ ステップ以内に停止} \\ (0,0) & \text{otherwise} \end{cases}$$

- $l(m,r,T)$: チューリング機械 T によって $g(m,r)$ が計算されるステップ数．
- 証明の列挙についてもアイデアが提示されている．

14.2 プログラム逆変換

1978 年に，Dijkstra はプログラム逆変換を論じている [19]．文献 [29] も分かりやすい．
入門書:

- “What Computing Is All About” の 11 章”Program Inversion”[74]．例として，木から前順走査と間順走査を生成する問題の逆を考えている．この問題は [36, 35, 73, 14] でも取り上げられている
  - 再帰による効率的な解法 [73]
  - 繰り返しによる解法 [14]
  - 前順走査から（非決定的に）木を構成 [28]
- [29]

関連文献

- LR 構文解析の応用 [27]
- 末尾再帰 [61]: [51] に似たアプローチ

14.2.1 ダイクストラの置換から符号への変換問題

プログラム逆変換の研究は，1978 年のダイクストラによる置換から符号への変換問題にまで遡ることができる [19]．問題は，整数列 $0, 1, \ldots, n-1$ の置換 $p[0 : n-1]$ から k 番目の要素 $y[k]$ が $p[0 : k-1]$ の中の

$p[k]$ よりも小さい要素数であるような配列 $y[0:n-1]$ を作成するというものである．すなわち，$0 \leqslant i < n$ であるすべての i について

$$y[i] \;=\; \#\{j \mid 0 \leqslant j < i.\ p[j] < p[i]\} \tag{14.1}$$

が成り立っている．

14.3 双方向変換

- この分野を切り開いてきた研究グループの成果 [24]
- Symmetric Lenses [31]
- Semantic approach to automatic generation of put-functions [76]
- “Notions of Bidirectional Computation and Entangled State Monads” に関連研究がいろいろある．

参考文献

[1] Axelsen, H. B.: Clean Translation of an Imperative Reversible Programming Language, *Compiler Construction* (Knoop, J., ed.), Lecture Notes in Computer Science, Vol. 6601, Springer-Verlag, pp. 144–163 (online), DOI:10.1007/978-3-642-19861-8_9 (2011).

[2] Axelsen, H. B. and Glück, R.: A Simple and Efficient Universal Reversible Turing Machine, *Language and Automata Theory and Applications* (Dediu, A.-H., Inenaga, S. and Martín-Vide, C., eds.), Lecture Notes in Computer Science, Vol. 6638, Springer-Verlag, pp. 117–128 (online), DOI:10.1007/978-3-642-21254-3_8 (2011).

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

[4] Axelsen, H. B. and Glück, R.: Reversible Representation and Manipulation of Constructor Terms in the Heap, *Reversible Computation. Proceedings* (Dueck, G. W. and Miller, D. M., eds.), Lecture Notes in Computer Science, Vol. 7948, Springer-Verlag, pp. 96–109 (online), DOI:10.1007/978-3-642-38986-3_9 (2013).

[5] Axelsen, H. B., Glück, R., De Vos, A. and Thomsen, M. K.: MicroPower: Towards Low-Power Microprocessors with Reversible Computing, *ERCIM NEWS*, Vol. 79, pp. 20–21 (2009).

[6] Axelsen, H. B. and Yokoyama, T.: Programming Techniques for Reversible Comparison Sorts, *Programming Languages and Systems* (Feng, X. and Park, S., eds.), Lecture Notes in Computer Science, Vol. 9458, Springer-Verlag, pp. 407–426 (2015).

[7] Bennett, C. H.: Logical Reversibility of Computation, *IBM Journal of Research and Development*, Vol. 17, No. 6, pp. 525–532 (online), DOI:10.1147/rd.176.0525 (1973).

[8] Bennett, C. H.: The Thermodynamics of Computation—a Review, *International Journal of Theoretical Physics*, Vol. 21, No. 12, pp. 905–940 (1982).

[9] Bennett, C. H.: Time/space trade-offs for reversible computation, *SIAM Journal on Computing*, Vol. 18, No. 4, pp. 766–776 (online), DOI:http://dx.doi.org/10.1137/0218053 (1989).

[10] Bennett, C. H.，Landauer, R.，唐木幸比古 (訳)：計算の物理的な限界はあるか？，サイエンス， Vol. 9, pp. 104–114 (1985). (原文: Bennett, C. H. and Landauer, R.: The Fundamental Physical Limits of Computation, *Scientific American*, Vol. 253, No. 1, pp. 48–56 (1985).).

[11] Bennett, C. H. and Schumacher, B.: マクスウェルの悪魔現る，日経サイエンス，Vol. 2011 年 8 月号, pp. 32–37 (2011). (原題: Maxwell's Demons Appear in the Lab).

[12] Bérut, A., Arakelyan, A., Petrosyan, A., Ciliberto, S., Dillenschneider, R. and Lutz, E.: Experimental verification of Landauer's principle linking information and thermodynamics, *Nature*, Vol. 483, pp. 187–189 (2012).

[13] Carothers, C. D., Perumalla, K. S. and Fujimoto, R. M.: Efficient Optimistic Parallel Simulations Using Reverse Computation, *ACM Transactions on Modeling and Computer Simulation*, Vol. 9, No. 3, pp. 224–253 (online), DOI:10.1145/347823.347828 (1999).

[14] Chen, W. and Udding, J. T.: Program inversion: More than fun!, *Science of Computer Programming*, Vol. 15, No. 1, pp. 1–13 (online), DOI:http://dx.doi.org/10.1016/0167-6423(90)90042-C (1990).

[15] Danos, V. and Krivine, J.: Reversible Communicating Systems, *Conference on Concurrency Theory* (Gardner, P. and Yoshida, N., eds.), Lecture Notes in Computer Science, Vol. 3170, Springer-Verlag, pp. 292–307 (online), DOI:10.1007/978-3-540-28644-8_19 (2004).

[16] Danos, V. and Krivine, J.: Formal Molecular Biology Done in CCS-R, *Electronic Notes in Theoretical Computer Science*, Vol. 180, No. 3, pp. 31–49 (online), DOI:http://dx.doi.org/10.1016/j.entcs.2004.01.040 (2007).

[17] del Rio, L., berg, J., Renner, R., Dahlsten, O. and Vedral, V.: The thermodynamic meaning of negative entropy, *Nature* (2011).

[18] Di Pierro, A., Hankin, C. and Wiklicky, H.: Reversible combinatory logic, *Mathematical Structures in Computer Science*, Vol. 16, No. 4, pp. 621–637 (online), DOI:10.1017/S0960129506005391 (2006).

[19] Dijkstra, E. W.: EWD 671: Program inversion (1978). published as [20].

[20] Dijkstra, E. W.: Program inversion, *Selected Writings on Computing: A Personal Perspective*, Springer-Verlag, pp. 351–354 (1982).

[21] Early, D., Gao, A. and Schellekens, M.: Frugal Encoding in Reversible $\mathcal{MOQA}$: A Case Study for Quicksort, *Reversible Computation. Proceedings* (Glück, R. and Yokoyama, T., eds.), Lecture Notes in Computer Science, Vol. 7581, Springer-Verlag, pp. 85–96 (2013).

[22] Feynman, R. P.: Reversible computation and the thermodynamics of computing (chapter 5), *Feynman Lectures on Computation* (Hey, A. J. G. and Allen, R. W., eds.), Addison-Wesley, pp. 137–184 (1996). Feynman, R. P. (Hey, A. J. G., Allen, R. W. 編, 原康夫，中山健，松田和典訳): ファインマン計算機科学, 第 5 章「可逆計算と計算の熱力学」, 岩波書店, (1999).

[23] Feynman, R. P.: ファインマン計算機科学，岩波書店 (1999).

[24] Foster, J. N., Greenwald, M. B., Moore, J. T., Pierce, B. C. and Schmitt, A.: Combinators for Bi-Directional Tree Transformations: A Linguistic Approach to the View Update Problem, *ACM Transactions on Programming Languages and Systems*, Vol. 29, No. 3, pp. 1–65 (online), DOI:10.1145/1232420.1232424 (2007).

[25] Frank, M. P.: Reversible Turing machines. Recursive insolubility in $n \in \mathbf{N}$ of the equation $u = \theta^n u$, where θ is an "isomorphism of codes", http://www.ai.mit.edu/~mpf/rc/Lecerf/lecerf.html (1998). Original paper [46].

[26] Fredkin, E. and Toffoli, T.: Conservative logic, *International Journal of Theoretical Physics*, Vol. 21, pp. 219–253 (1982).

[27] Glück, R. and Kawabe, M.: A Method for Automatic Program Inversion Based on LR(0) Parsing, *Fundamenta Informaticae*, Vol. 66, No. 4, pp. 367–395 (2005).

[28] Gries, D. and van de Snepscheut, J. L.: Formal Development Programs and Proofs, *Formal Development of Programs and Proofs* (Dijkstra, E. W., ed.), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, chapter Inorder Traversal of a Binary Tree and Its Inversion, pp. 37–42 (online), http://dl.acm.org/citation.cfm?id=70020.70023 (1990).

[29] Gries, D.: *The Science of Programming*, Texts and Monographs in Computer Science, chapter 21 Inverting Programs, pp. 265–274, Springer-Verlag (1981).

[30] Hayes, B.: Reverse Engineering, *American Scientist*, Vol. 94, No. 2, pp. 107–111 (2006).

[31] Hofmann, M., Pierce, B. and Wagner, D.: Symmetric Lenses, *Principles of Programming Languages. Proceedings*, New York, NY, USA, ACM, pp. 371–384 (online), DOI:10.1145/1926385.1926428 (2011).

[32] Hopcroft, J. E., Motwani, R., Rotwani and Ullman, J. D.: *Introduction to Automata Theory, Languages and Computability*, Addison-Wesley, 2nd edition (2000).

[33] Jones, N. D., Gomard, C. K. and Sestoft, P.: *Partial evaluation and automatic program generation*, Prentice-Hall, Inc. (1993).

[34] Kluge, W. E.: A Reversible SE(M)CD Machine, *Implementation of Functional Languages. Proceedings, Selected Papers* (Koopman, P. and Clack, C., eds.), Lecture Notes in Computer Science, Vol. 1868, Springer-Verlag, pp. 95–113 (2000).

[35] Knuth, D. E.: *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison Wesley Longman Publishing Co., Inc. (1973).

[36] Knuth, D. E.: *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*, Addison Wesley Longman Publishing Co., Inc. (1998).

[37] Krivine, J.: *Reversible Computation: 4th International Workshop, RC 2012, Copenhagen, Denmark, July 2-3, 2012. Revised Papers*, chapter A Verification Technique for Reversible Process Algebra, pp. 204–217 (online), DOI:10.1007/978-3-642-36315-3_17, Springer Berlin Heidelberg (2013).

[38] Kurzweil, R.: シンギュラリティは近い一人類が生命を超越するとき，NHK 出版 (2010).

[39] Kutrib, M.: Aspects of Reversibility for Classical Automata, *Computing with New Resources* (Calude, C. S., Freivalds, R. and Kazuo, I., eds.), Lecture Notes in Computer Science, Vol. 8808, Springer-Verlag, pp. 83–98 (2014).

[40] Kutrib, M. and Malcher, A.: Reversible Pushdown Automata, *Language and Automata Theory and Applications* (Dediu, A.-H., Fernau, H. and Martn-Vide, C., eds.), Lecture Notes in Computer Science, Vol. 6031, Springer Berlin Heidelberg, pp. 368–379 (online), DOI:10.1007/978-3-642-13089-2_31 (2010).

[41] Kutrib, M. and Malcher, A.: Reversible pushdown automata, *Journal of Computer and System*

Sciences, Vol. 78, No. 6, pp. 1814–1827 (online), DOI:dx.doi.org/10.1016/j.jcss.2011.12.004 (2012).

[42] Kutrib, M., Malcher, A. and Wendlandt, M.: Reversible queue automata, *Sixth Workshop on Non-Classical Models for Automata and Applications - NCMA 2014, Kassel, Germany, July 28-29, 2014. Proceedings*, pp. 163–178 (2014).

[43] Landauer, R.: Irreversibility and Heat Generation in the Computing Process, *IBM Journal of Research and Development*, Vol. 5, No. 3, pp. 183–191 (online), DOI:10.1147/rd.53.0183 (1961).

[44] Landauer, R.，小柳義夫 (訳)：情報は物理過程だ，パリティ， Vol. 6, No. 11, pp. 30–39 (1991). (原文: Landauer, R.: Information is Physical, *Physics Today*, Vol. 44, No. 5, pp. 23–29 (1991).).

[45] Lange, K.-J., McKenzie, P. and Tapp, A.: Reversible space equals deterministic space, *Journal of Computer and System Sciences*, Vol. 60, No. 2, pp. 354–367 (2000).

[46] Lecerf, Y.: Machines de Turing réversibles. Récursive insolubilité en $n \in \mathbf{N}$ de l'équation $u = \theta^n u$, où θ est un « isomorphisme de codes », *Comptes Rendus Hebdomadaires des Séances de L'académie des Sciences*, Vol. 257, pp. 2597–2600 (1963).

[47] Li, M. and Vitányi, P.: Reversibility and Adiabatic Computation: Trading Time and Space for Energy, *Proceedings: Mathematical, Physical and Engineering Sciences*, Vol. 452, No. 1947, The Royal Society, pp. 769–789 (1996).

[48] Li, M. and Vitányi, P.: Reversible simulation of irreversible computation, *Computational Complexity, 1996. Proceedings., Eleventh Annual IEEE Conference on*, pp. 301–306 (1996).

[49] Lutz, C.: Janus: A time-reversible language (1986). Letter to R. Landauer.

[50] McCarthy, J.: The inversion of functions defined by Turing machines, *Automata Studies* (Shannon, C. E. and McCarthy, J., eds.), Princeton University Press, pp. 177–181 (1956).

[51] Mogensen, T. Æ.: Report on an Implementation of a Semi-inverter, *International Andrei Ershov Memorial Conference on Perspectives of Systems Informatics. Proceedings*, Springer-Verlag, pp. 322–334 (online), DOI:10.1007/978-3-540-70881-0_28 (2007).

[52] Mogensen, T. Æ.: Partial Evaluation of the Reversible Language Janus, *Partial Evaluation and Program Manipulation. Proceedings*, ACM Press, pp. 23–32 (online), DOI:10.1145/1929501.1929506 (2011).

[53] Mogensen, T. Æ.: Partial Evaluation of Janus Part 2: Assertions and Procedures, *Perspectives of Systems Informatics* (Clarke, E., Virbitskaite, I. and Voronkov, A., eds.), Lecture Notes in Computer Science, Vol. 7162, Springer Berlin Heidelberg, pp. 289–301 (online), DOI:10.1007/978-3-642-29709-0_25 (2012).

[54] Mogensen, T. Æ.: Reference Counting for Reversible Languages, *Reversible Computation. Proceedings* (Yamashita, S. and Minato, S.-i., eds.), Lecture Notes in Computer Science, Vol. 8507, Springer-Verlag, pp. 82–94 (online), DOI:10.1007/978-3-319-08494-7_7 (2014).

[55] Morita, K.: Reversible computing and cellular automata — A survey, *Theoretical Computer Science*, Vol. 395, No. 1, pp. 101–131 (online), DOI:10.1016/j.tcs.2008.01.041 (2008).

[56] Morita, K.: A Deterministic Two-Way Multi-Head Finite Automaton can be Converted into a Reversible one with the same Number of Heads, *Preliminary Proceedings of 4th Workshop on Reversible Computation* (Glück, R. and Yokoyama, T., eds.), pp. 28–40 (2012).

[57] Morita, K.: Universal reversible Turing machines with a small number of tape symbols, *Fundamenta Informaticae*, Vol. 138, No. 1–2, pp. 17–29 (2015).

[58] Morita, K., Shirasaki, A. and Gono, Y.: A 1-Tape 2-Symbol Reversible Turing Machine, *IEICE Trans.*, Vol. E 72, No. 3, pp. 223–228 (1989).

[59] Mu, S.-C., Hu, Z. and Takeichi, M.: An Injective Language for Reversible Computation, *Mathematics of Program Construction. Proceedings* (Kozen, D., ed.), Lecture Notes in Computer Science, Vol. 3125, Springer-Verlag, pp. 289–313 (online), DOI:10.1007/978-3-540-27764-4_16 (2004).

[60] Neumann, J. V.: *Theory of Self-Reproducing Automata*, University of Illinois Press, Champaign, IL, USA (1966).

[61] Nishida, N. and Vidal, G.: Program Inversion for Tail Recursive Functions, *International Conference on Rewriting Techniques and Applications. Proceedings* (Schmidt-Schauß, M., ed.), Leibniz International Proceedings in Informatics, Vol. 10, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 283–298 (online), DOI:10.4230/LIPIcs.RTA.2011.283 (2011).

[62] Pan, W. and Nalasani, M.: Reversible logic, *Potentials, IEEE*, Vol. 24, No. 1, pp. 38–41 (online), DOI:10.1109/MP.2005.1405801 (2005).

[63] Perumalla, K.: RCC - Reverse C Compiler. http://www.cc.gatech.edu/computing/pads/rcc.html.

[64] Perumalla, K. S.: *Introduction to Reversible Computing*, CRC Press (2013).

[65] Perumalla, K. S. and Fujimoto, R. M.: Source-code Transformations for Efficient Reversibility, Technical Report GIT-CC-99-21, Georgia Institute of Technology (1999).

[66] Pin, J.-E.: On the Language Accepted by Finite Reversible Automata, *International Colloquium on Automata, Languages and Programming. Proceedings* (Ottmann, T., ed.), Lecture Notes in Computer Science, Vol. 267, Springer-Verlag, pp. 237–249 (1987).

[67] Pin, J.-E.: On Reversible Automata, *1st Latin American Symposium on Theoretical Informatics (LATIN)*, pp. 401–416 (1992).

[68] Thomsen, M. K., Axelsen, H. B. and Glück, R.: A Reversible Processor Architecture and Its Reversible Logic Design, *Reversible Computation. Proceedings* (De Vos, A. and Wille, R., eds.), Lecture Notes in Computer Science, Vol. 7165, Springer-Verlag, pp. 30–42 (online), DOI:10.1007/978-3-642-29517-1_3 (2012).

[69] Toffoli, T.: Reversible Computing, *Automata, Languages and Programming. Proceedings* (de Bakker, J. W. and van Leeuwen, J., eds.), Lecture Notes in Computer Science, Vol. 85, Springer-Verlag, pp. 632–644 (1980).

[70] Toffoli, T.: Bicontinuous extensions of invertible combinatorial functions, *Mathematical systems theory*, Vol. 14, No. 1, pp. 13–23 (online), DOI:10.1007/BF01752388 (1981).

[71] Toyabe, S., Sagawa, T., Ueda, M., Muneyuki, E. and Sano, M.: Experimental demonstration of information-to-energy conversion and validation of the generalized Jarzynski equality, *Nature Physics*, Vol. 6, pp. 988–992 (2010).

[72] Turing, A. M.: On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society. Second Series*, Vol. 42, pp. 230–265 (1936).

[73] van de Snepscheut, J. L. A.: Inversion of a Recursive Tree Traversal, *Inf. Process. Lett.*, Vol. 39, No. 5, pp. 265–267 (online), DOI:10.1016/0020-0190(91)90026-E (1991).

[74] van de Snepscheut, J. L. A.: *What Computing Is All About*, Text and Monographs in Computer Science, chapter 11 Program Inversion, Springer-Verlag (1993).

[75] Vieri, C. J.: Pendulum: A Reversible Computer Architecture, Master's thesis, Massachusetts Institute of Technology (1995).

[76] Voigtländer, J.: Bidirectionalization for Free! (Pearl), *Principles of Programming Languages. Proceedings*, New York, NY, USA, ACM, pp. 165–176 (online), DOI:10.1145/1480881.1480904 (2009).

[77] Wenzler, J.-S., Dunn, T., Toffoli, T. and Mohanty, P.: A Nanomechanical Fredkin Gate, *Nano Letters*, Vol. 14, No. 1, pp. 89–93 (online), DOI:10.1021/nl403268b (2013).

[78] Yokoyama, T., Axelsen, H. B. and Glück, R.: Principles of a Reversible Programming Language, *Computing Frontiers. Proceedings*, ACM Press, pp. 43–54 (2008).

[79] Yokoyama, T., Axelsen, H. B. and Glück, R.: Reversible Flowchart Languages and the Structured Reversible Program Theorem, *International Colloquium on Automata, Languages and Programming. Proceedings* (Aceto, L., Damgård, I., Goldberg, L. A., Halldórsson, M. M., Ingólfsdóttir, A. and Walukiewicz, I., eds.), Lecture Notes in Computer Science, Vol. 5126, pp. 258–270 (online), DOI:10.1007/978-3-540-70583-3_22 (2008).

[80] Yokoyama, T., Axelsen, H. B. and Glück, R.: Minimizing Garbage Size by Generating Reversible Simulations, *Reversibility, cellular automata, and unconventional computation. Proceedings*, IEEE Computer Society, pp. 379–387 (2012).

[81] Yokoyama, T., Axelsen, H. B. and Glück, R.: Towards a Reversible Functional Language, *Reversible Computation. Proceedings* (De Vos, A. and Wille, R., eds.), Lecture Notes in Computer Science, Vol. 7165, Springer-Verlag, pp. 14–29 (online), DOI:10.1007/978-3-642-29517-1_2 (2012).

[82] Yokoyama, T. and Glück, R.: A Reversible Programming Language and Its Invertible Self-Interpreter, *Partial Evaluation and Semantics-Based Program Manipulation. Proceedings*, ACM Press, pp. 144–153 (online), DOI:10.1145/1244381.1244404 (2007).

[83] Zuliani, P.: Logical reversibility, *IBM Journal of Research and Development*, Vol. 45, No. 6, pp. 807–818 (online), DOI:10.1147/rd.456.0807 (2001).

[84] 森田憲一：2. 計算における可逆性：可逆チューリング機械と可逆論理回路 (〈特集〉逆計算：計算の理論における逆問題)，情報処理， Vol. 35, No. 4, pp. 306–314 (1994).

[85] 森田憲一：3. 可逆セル・オートマトン (〈特集〉逆計算：計算の理論における逆問題)，情報処理， Vol. 35, No. 4, pp. 315–321 (1994).

[86] 森田憲一：可逆計算，近代科学社 (2012).

[87] 古田　彩：多数の宇宙で計算する—常識を揺るがすコンピュータ— (2006).

[88] 古田　彩：多世界から生まれた計算機，日経サイエンス， Vol. 2008 年 04 号, pp. 96–103 (2008).

[89] 森田憲一：可逆コンピューティング—ビリヤードボールでコンピュータが作れるか？—，情報処理，Vol. 53, No. 5, pp. 496–502 (2012).

[90] 青本和彦，上野健爾，加藤和也，神保道夫，砂田利一，高橋陽一郎，深谷賢治，俣野　博，室田一雄 (編)：岩波数学入門辞典，岩波書店 (2005).