数学教育におけるコンピュータ教育(1)
——“チューリング機械”の教材化——

宮下英明*・松沢正規**

目　次

*宮下　英明　　金沢大学教育学部
**松沢　正規　　金沢市立高尾台中学校

1 コンピュータと数学教育

1.1 コンピュータ教育が数学教育になるときの理由

コンピュータ指導を数学教育の一環に含めるとすれば，このときの理由は，

《コンピュータ指導が数学の教育として位置づく》

ということ――これのみ――でなければならない。数学教育は，コンピュータ指導全般のうち，〈数学の教育として位置づくコンピュータ指導〉を分担して受け持つ。

例えばつぎのようなことの指導は，数学教育が受け持つものではない：

・コンピュータ本体およびそれの周辺機器の操作
・各種市販ソフト（リレーショナル・データ・ベースソフト，ワープロソフト，プログラム開発言語，・・・・）の使用法
・あるコンピュータ機種のハード，OSの仕組み

1.2 数学教育が引き受けるコンピュータ教育――認識論的コンピュータ教育

現代に生きる／生かされる者にとって，コンピュータがますます閑却できないものになってきているのは，単に生活様式の変化（それは“便利化”のように受け取られているが）という理由からではない。教育的観点からは，コンピュータの今日的意義として強調されるべきは，“認識の転換”――著しくは，“パラダイムの変換”――のトリガーとしてのそれの意義である。

特に，コンピュータはわれわれが自分自身についてこれまで見ていなかったものに，目を向けさせる――より正確に言えば，“これまで見ていなかったもの”という形で，自分自身についての或ることを対象化させる――のである。

“自分自身について見ていなかったもの”には，自分の中の欠落部分も含まれる。われわれは，コンピュータを知ることで，自分のうちに有るものと同時に，無いものも対象化できるようになる。

こうして，“認識の転換を射程におく”ということが，コンピュータ教育を意義づける一つの形態になる。われわれはこのように意義づけられた“コンピュータ教育”を，“認識論的コンピュータ教育”と呼んでおこう。

数学教育は，認識論的コンピュータ教育を，各論的に引き受ける。

2 数学教育における認識論的コンピュータ教育の主題

2.1 記号論

数学教育が“コンピュータ教育”を各論的に引き受けるときのその主題は，端的に，“記号論”である。

コンピュータ教育で顕在化しようとする“これまではそれと意識してはいなかったもの”，“意識に欠落していたもの”は，数学教育の場合には，数学の記号論的事実ということになる。数学教育は，数学に対する認識を記号論的なものとするトリガーとして，コンピュータを利用する。

内容的に言えば，コンピュータ指導は，数学に対する存在論的な反省――例えばその一つとして，構成主義的ないし有限主義的な捉え直しの主題化――を促すものとして，数学教育の中に位置付けられるのである。そしてこのときの“コンピュータ”の意義は，“データ処理機械”

——特にそのうちの“記号列変形機械”——である。

2.2 イデア的発想から生成的発想への転換

2.2.1 数学に対するイデア的発想

数学は，〈存在〉の研究と思われがちである。〈実在〉の研究と思われてはいないまでも，数学の言説は〈存在〉に関する言説であると思われている。《そこにはことばしかない》という全く自明なことが，看過されている。

数学の実践は，テクストの生成であり，かつ生成規則のみに従う生成である。〈何か〉について語っているように見えるとしても，それは見掛け（体裁）に過ぎない。

数学は，本質的に，ことばの変形ゲームに終始する。ただ，数学は〈何か〉について語るという体裁で実践されている。そしてひとは，この見掛けに騙されて，数学が〈何か〉についての言説であると受け取ってしまう。

2.2.2 〈機械〉との出会いによるイデア的発想からの脱却

記号に対するわれわれの思い入れが，記号を〈何か〉についての言説として見させている。そこで逆に，思い入れを持たなければ，記号そのものを直視できるはずである。

コンピュータは，この目的のために援用できる。

実際，コンピュータとの出会いによって，ひとはイデア的発想から自然に脱却するようになる。生成的発想へと向かうようになる。

ひとは，コンピュータの上の記号については，それが〈何か〉を語るものでないことを，抵抗なく事実として受容できるのである。《コンピュータの上の記号は〈何か〉を語るものではなく，〈何か〉は，われわれが記号に対して勝手に読むところのものである》という認識をもてる。

要するに，人は“機械”の発する記号に対しては思い入れを持てないのである。そこで，このことを逆手にとって，《記号に対する思い入れを捨てさせるために機械を使う》という戦略が立つ。そして実際，コンピュータに出会うことで，ひとは〈記号〉に対する思い入れを捨てることができるようになり，〈記号〉そのものを見るようになるのである。

2.3 “機械”の概念

“人”と“機械”は，対立概念(註)とすべきではない。実際，われわれは人を機械として見ることができる——例えば，われわれは数学の実践を“機械”の実践として捉えることができる。

“人を機械として見る”の言い回しの意味は，“人の比喩として機械を用いる”ではない。このときの“機械”の語は，“人”を捉える視点の一つを表現しているのである。

“機械”とは，

“状況に流されることがそれの実践であるところの実践者”

の謂いである。したがって，例えば筆算（計算規則に従わされている実践）を行なっている人は“機械”である。

(註) “人”と“機械”は，本質的に，つぎの言い回しにおいて対立概念となる：

“人の外見をもつ機械”

“機械の外見をもつ人”

2.4 シンタクス，セマンティクスの概念

2.4.1 “シンタクス”，“セマンティクス”の解釈

数学の学習に対し，シンタクスの学習とセマンティクスの学習の区別を立てることができる。前者の学習内容は，言語としての数学の文法であり，後者のそれは，数学の文を処理する機械である。

ひとは，テクストに意味——例えば，数学——が込められていると考えたがる。しかし，テクストは“無意味”である。例えば，“数学のテクスト”の言い回しをわれわれは受容するが，数学のテクストはそれ自体では“無意味”である。テクストはデータであり，そして一般にデータは無意味である。

意味は，〈ある〉のではなく〈示される〉。即ち，データ処理の実践において示される。但し，実践の外に意味があるのではなく，実践そのものが意味である。データの意味は，このデータに対してデータ処理機械が如何に動作するかということである。——特に，データの意味は，それを処理する機械に依存する。

あるデータを“数学のテクスト”にしているものは，データ処理機械としてのわれわれの実践である。

結局，はじめに述べた数学学習の二つの内容——シンタクスとセマンティクス——は，

(1) データ形式

(2) データ処理機械

のことばで言い換えられる。逆に，“シンタクス”，“セマンティクス”は，この二つに対する言い回しに他ならない。

データ処理機械としての自分の参入を看過することが，テクストに数学が込められていると考えてしまう理由である。そしてコンピュータは，ひとを，この看過していた〈データ処理機械〉の意識化に導く。

2.4.2 セマンティクスの階梯——“メタ”の概念

われわれは，データ処理機械のデータ化（記述）を主題化することができる。

いま，データ処理機械Mに関するデータを，Mが処理するデータに対するメタデータと呼ぶことにする。——“メタ”の語のこの用法は，通常のものである。

メタデータを“メタデータ”とするものは，これを処理する別のもう一つの機械M′である。M′はMのメタ機械ということができる。

現前の“数学テクスト”は，《テクスト—メタテクスト》の枠組みで分析することができる。したがって，われわれの〈テクスト読解〉の実践では，われわれはその都度妥当な機械に変化して，テクストを処理していることになる。

2.4.3 推論規則，アルゴリズム

“推論規則”，“アルゴリズム”は，数学の対象式／関係式をデータとしたときの“データ処理機械”と見なせる。そしてこれが，数学教育で最も明証的に扱える〈機械〉である。

2.5 定数，変数の概念

高級言語で書かれたプログラムがどのように機械語のプログラムに展開され，また機械語プログラムがどのようにコンピュータの上に実現されるかを知ることで，定数，変数の概念が理解されるようになる。

3 〈データ処理機械〉の主題化

3.1 〈データ処理機械〉の主題化の意味

既に述べたように，ひとは数学のセマンティクスをテクストに帰してしまいがちである。この認識が誤りであることは，ひとがこのときしていることをつぎのように表現すれば明らかになる。即ち，ひとはこのとき，セマンティクスをシンタクスに含意させていることになる。

シンタクスに応じるのがテクストであるのに対し，セマンティクスに応じるものは，数学する主体——より正確には，その実践——である。そして，ひとはこれを看過しやすい。

数学教育として〈データ処理機械〉を主題化する意味は，曖昧にされてきたセマンティクスの帰属問題を正しく（まともに）取り上げることである。

3.2 一般論と各論

〈データ処理機械〉の指導がゴールとする“学習者の変容”は，学習者が

《数学には，テクストともに，データ処理機械が存在している》

という認識をもつようになること——この文脈で，〈データ処理機械〉の概念をもつようになること——である。

この指導は，一般論かつ各論というように，本来継続的に進められるべきである。

注意しておくが，一般論が目的であると考えてはならない。そもそも一般論は，各論によって鍛え上げられる。“一般論は各論の形で展開される他ない”といった消極的意味で，“各論の上の一般論の指導”を考えてはならない。

3.3 “明証性”

〈データ処理機械〉の主題においては，

〈機械〉の学習＝〈明証的な機械〉の学習

である。教材とする機械は明証的でなければならないというのではなく，明証性——特個的な明証性——そのものが学習の対象なのである。

ここで言う“明証性”の内容は，以下のようなことである：

(1) 機械の“最も低級”な機能が明らかにされている——暗部（ブラックボックス）を含んでいない；

(2) しかもそれは，われわれが“直観的に明らか”なものとして受容できるようなものである；そして，

(3) 機械の機能は，最も低級な機能から出発して構成されるものとして理解可能である。

但し，“暗部（ブラックボックス）が存在しない”とは，

《“暗部（ブラックボックス）が存在しない”というように，われわれは受容する》

ということである。また，“直観的に明らか”は，事態の単純さを意味しない。実際，“直観的に明らか”を，説明することはできない——そもそも“直観的に明らか”は，説明の放棄の謂いである。

3.4 〈データ処理機械〉に関する学習内容

〈データ処理機械〉に関する学習内容は，以下のものである；

(1) 機械の構成

(2) 機械の動作原理

(3) 機械が受容するデータの形式

4 データ処理機械のコンピュータ上の実現

4.1 データ処理機械のコンピュータ上の実現

〈機械＝明証的な機械〉をコンピュータの上で実現する——即ち，機械実現の実行プログラムに機械定義プログラムを受容させることで機械を実現する——ことを考えよう。

この場合，コンピュータがわれわれにとって明証的でないことは，問題にならない。実際，“機械の土台の明証性”は“機械の明証性”とは別の問題である。

機械をコンピュータの上に実現することの第一の意義は，“〈仮想機械〉の拒否”である。フィクションの仮想機械は，自分のクローンに過ぎない。機械は，自分に対し他者——自分をチェックする存在——として現前するものでなければならない。われわれは，機械が〈他者〉として登場する情況として，コンピュータを用いるのである。

4.2 コンピュータ上に実現された機械に関して主題化されることがら

コンピュータ上に実現された機械に関しては，§3.4で挙げた学習内容の他に，以下のようなことの主題化が考えられる：

(4) 機械定義用のプログラム言語としての機械語

(5) 機械語に対する高級言語

(6) 高級言語で書かれた機械定義プログラムから機械語プログラムを生成するツール——コンパイラ&リンカ

5 チューリング機械の教材化

教材となる〈データ処理機械〉の条件として，

(1) 構造が簡単であり，扱いやすい

(2) 価値がある

を考える。(1)と(2)は一見排反する条件のように見えるが，必ずしもそうではない。実際，“アルゴリズム”は，この条件を満たす〈データ処理機械〉である。

“アルゴリズム”を機械として表現したのが，チューリング機械である。われわれはこのチューリング機械の教材化を主題化する。

なお，構造が簡単であるということと，効率的であるということは，一般に排反する。実際，チューリング機械は，オーバヘッドの大きさが非常に目立つ機械である。プログラムの困難さと相俟って，“極めて簡単なことの他は何もできない機械”という印象をもたれ易い。

6 チューリング機械の存在論

6.1 存在論の問題化

チューリング機械の理論では，チューリング機械の存在論は閑却してよい主題である。しかし，われわれの場合のように，チューリング機械を実現しようとするときには，これの存在論が問題化する。

6.2 解釈としての“チューリング機械”

“チューリング機械”は解釈である。それは，

(1) 機械を，テープとヘッドと内部状態でなるものとし，

(2) それの動作を，“状態推移図/動作表”に従うものとする

解釈である。言い換えると，

“《一つの状態推移図/動作表を規則としてこれに従って動作する機械》のように解釈された機械を，この状態推移図/動作表によって定義されるチューリング機械と呼ぶ”

ということである。この意味で，

《一つの状態推移図/動作表を規則としてこれに従って動作するようにつくられた機械》

は，この状態推移図/動作表によって定義されるチューリング機械である。

特に，“チューリング機械”は“テープ，ヘッド，内部状態，そして状態推移図/動作表で定義される抽象的な機械”——〈仮想機械〉——ではない。

実際，“仮想機械”といえども，機械である。それは機械として仮想できるものでなければならない。そして，“テープとヘッドと内部状態の三つだけでなる機械”は，仮想できないのである。機械として仮想できるためには，これらの要素は他の部品によって結合され，連絡していなければならない。また，起動システムを閑却して機械を仮想することも，不可能である。

“状態推移図/動作表で定義される機械”——“状態推移図/動作表としての機械”——の仮想も不可能である。実際，われわれはそのような様を想像し得ない。

“チューリング機械”の概念には，“仮想機械”という含意はない。“チューリング機械”と解釈されたものがチューリング機械であり，チューリング機械は実在の機械であり得る。そしてまた，チューリング機械を仮想することもできる——仮想したそのチューリング機械は，仮想機械である。

6.3 潜在性と顕在化

“一つの動作表はチューリング機械を定義している”と考えるとき，われわれは，チューリング機械を潜在性として導入していることになる。〈機械〉は，テープが装着されて起動されたのち，テープ記号に対する反応として，顕在化する。これが〈動作〉である。

一般に，動作する以前の〈機械〉を対象化しようとするとき，対象化の形態は，〈機械＝潜在性〉となる。“動作以前”は，“潜在性”であるしかない。動作表によるチューリング機械の定義は，このようなものである。それは，動作以前の機械を対象化する方法に則っている。

この方法は，つぎの一般的方法の枠内にある：

《現象を生成的現象と見なし，“現象以前”を生成の潜在性として対象化する。明示的には，〈生成規則〉として対象化する。》(註)

動作表の意義は，“生成規則”である。われわれは，運動生成の潜在性として，そして明示的には運動生成規則として，チューリング機械を導入したのである。

(註) 例えば，“言語現象以前”の言語を，文法（文生成規則）として対象化する。

6.4 “起動”の概念の欠如

チューリング機械には，“起動（スイッチオン）”の概念がない。

テープへの“入力”によって起動すると考えることはできない。何故なら，何も書かれていないテープもまた，“入力テープ”になるからである。

テープの“装着”を起動と考えることはでき

よう。しかし,《テープ装着のつぎにくるものとしての“起動”の概念が，はじめから欠如している》と見る方が，自然である。

“テープの装着”の概念化は，同時に，テープが装着されていないチューリング機械”の概念化である。“テープの装着されていないチューリング機械”は，動作表のことではない。われわれはチューリング機械を〈身体〉と考える。そして，動作表は身体の解釈であり，身体そのものではない。

チューリング機械の起動を，われわれはテープが既に装着されている時点でのものとして考える。この起動は，身体の状態（“内部状態”ではない）の一変——一つの状態から別の一つの状態への変化——として考えねばならない。コンピュータで言うと，電気回路としてのそれのオフからオンへの状態変化である。チューリング機械には，この意味の“起動”の概念が欠如しているのである。

チューリング機械に“起動”の概念が欠如していることは，理論の上の問題にはならない。“起動”は，この理論では閑却してよい主題である。しかし，チューリング機械を実現しようとするときには，“起動”を導入しなければならなくなる。

6.5 “テープ”の意味

テープの意味は，“入力に使うテープ”（入力装置）ではない。それはコンピュータのメモリ・ボードに相当する。

“テープ入力”とは，起動時のテープに対する一つの特別な読み方（思いを込めた読み方）である。これに対しては，“メモリへの直接書込み”の読みの方が当たっていることになるが，いずれにしても，チューリング機械の主題においてわれわれは“テープへの書込み”の概念を必要としない。実際われわれは，テープを所与として扱うことになる。

6.6 テープ記号

われわれは，言い回しの簡単のため，

(1) テープの“無記号”を表わす記号

(2) テープ記号

をあわせて“テープ記号”と呼ぶことにする。

なお以下では，_を“無記号”の記号とする。

6.7 基底動作

いま,チューリング機械を一つ固定して考え，これの“基底動作”の概念をつぎのように定義する。即ち，動作の集合Mにおいて条件：

(1) 任意の動作を，Mの要素の組み合わせで実現することができる；

(2) Mのどの要素も，Mの他の要素を組み合わせるやり方ではつくることができない。

が満たされているとき，Mに属する各動作を基底動作と呼ぶ。

このとき，各テープ記号xに対する

(x) ヘッドが指しているコマに，記号xを書込む

動作は，基底動作である。そしてこれの他に基底動作となるものはつぎの三つである：

(r) ヘッドを右に一コマ移動する；

(l) ヘッドを左に一コマ移動する；

(e) ヘッドが指しているコマに記号が書かれていれば，これを消去する。

6.8 チューリング機械リアライザ

“チューリング機械”に対して，“チューリン

グ機械をその上に現出させることのできる機械”の概念が立つ。これを“チューリング機械リアライザ”と呼んでおく。“チューリング機械リアライザ”は，“チューリング機械”のメタ概念と見なせる。

“チューリング機械リアライザ”も，“チューリング機械”と同様，解釈である。一つの機械に対して，チューリング機械か否か，チューリング機械リアライザか否かが，決まるのではない。

なお，チューリング機械リアライザをチューリング機械として作成できたとき，それは“汎用チューリング機械”というものになる。

7 チューリング機械の表現

7.1 〈チューリング機械＝潜在性〉の表現

7.1.1 〈チューリング機械＝潜在性〉の関数表現

いま，つぎの集合を考える：

(1) “テープ記号の全体”と読まれる集合S

(2) “内部状態の全体”と読まれる集合Q

(3) “(基底) 動作の全体”と読まれる集合A

〈チューリング機械＝潜在性〉は，“内部状態”，“ヘッド（の動作）”，“テープ（記号）”の語を用いて〈物理的機械〉のように述べられるが，理論的には，上のS，Q，Aの集合に関する関数として定式化できる——そしてこのときの関数には，つぎのようなものが考えられる：

(1) 読み：

“(内部状態，記号)⟼(動作，内部状態)”

の関数：

$$Q \times S \longrightarrow A \times Q$$

(2) 読み：

“内部状態⟼(記号⟼(動作, 内部状態))”

の関数

$$Q \longrightarrow (A \times Q)^S$$

(3) 読み：

“内部状態⟼(動作，記号⟼内部状態)”

の関数：

$$Q \longrightarrow A \times Q^S$$

(1)はクラシカルな定式化である。

(2)は(1)に準ずるものであり，

《記号に応じて（動作，内部状態）に分岐し，動作して，つぎの内部状態にいく》

ということで，switch-do-goto タイプ——略して，SD タイプ——言える。

(3)は，

《動作してから，記号に応じて内部状態に分岐する》

ということで，do-switch-goto タイプ——略して，DS タイプ——と言える。

われわれは，以下，〈チューリング機械＝潜在性〉の表現関数としては(2)と(3)の二つを専ら考えていくことにする。

7.1.2 動作表

〈チューリング機械＝関数〉の一つの表現式が“動作表”である。そこで，機械に対する“チューリング機械”の解釈の一つは，それの動作を一つの動作表に従うものと見なすことである。

〈チューリング機械＝関数〉に SD，DS の二つのタイプが考えられることに対応して，動作図にも SD，DS の二タイプが考えられる。

クラシカルな動作表は SD タイプのもので，つぎのようになる：

	s_1	····	s_k
⋮		••••	
q_i	a_1q_1	····	a_kq_k
⋮		••••	

$\left\{\begin{array}{l} s_1, \cdots\cdot, s_k : \text{テープ記号の全体} \\ q_n : \text{内部状態}, \ a_n : \text{基底動作} \end{array}\right\}$

これに対するDSタイプの動作表を，われわれは，つぎのように書くことにする：

		s_1 ···· s_k
⋮	⋮	••••
q_i	a	q_1 ···· q_k
⋮	⋮	••••

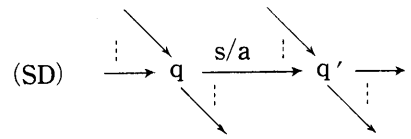
（a：基底動作）

7.1.3 状態推移図

〈チューリング機械＝関数〉の表現としては，動作表の他に“状態推移図”が一般的である。

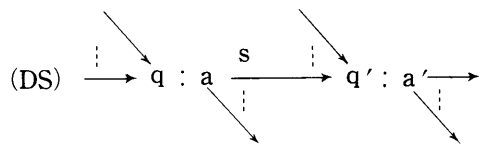
関数の二つのタイプSD，DSに対応して，状態推移図にもSD，DSの二タイプが考えられる。

クラシカルな状態推移図は，SDタイプのものであり，一つの状態推移が，内部状態q，q′，テープ記号s，動作aに対する図式：



で示される。

これに対し，DSタイプの動作表には，つぎの形の状態推移図が応ずることになる：



7.1.4 動作推移図

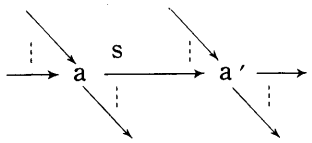
状態推移図における〈内部状態〉の意義は，“ネットワークのノード”である——これに尽きる。そして“異なる内部状態”が“異なるノード”として示されているところの状態推移図においては，〈内部状態〉の表記は本質的に過剰なことである。

実際，〈内部状態〉の表示は，無しで済ませることができる。それは，〈内部状態〉をノードとする状態推移図を，〈動作〉をノードとする“動作推移図”に書き換えるという形で，実現できる。そして状態推移図において考えられたタイプ分け——SDとDS——は，動作推移図においては解消する。

DSタイプの状態推移図を動作推移図に変える規則は簡単であり，それは

《ネットワークのノードになっている“q：a”を，“a”に換える》

である。§7.1.3の図(DS)は，つぎの図に変わる：

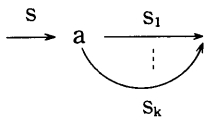


SDタイプの状態推移図に対しては，つぎのようにする：

(1) 何もしない動作nullを導入して，全ての内部状態をnullに書き換える。

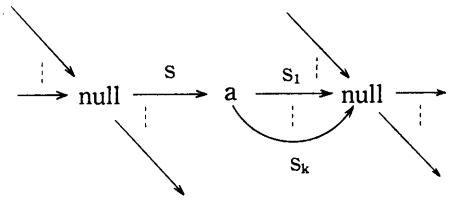
(2) 表記：

を，



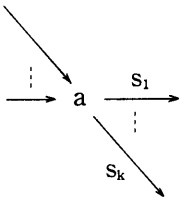
に書き換える——ここで，$s_1, \cdots\cdots, s_k$はテープ記号の全体。

§7.1.3の図(SD)は，つぎの図に変わる：



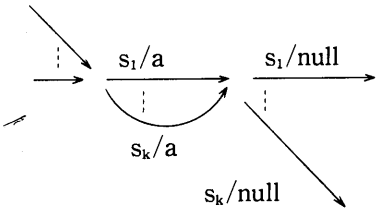
このように，SD，DSのどちらのタイプの状態推移図も動作推移図に変え得るわけであるが，逆に，動作推移図からSDとDSの両方のタイプの状態推移図を復元することも可能である。

実際，つぎのようにする。即ち，動作推移図でのノード：



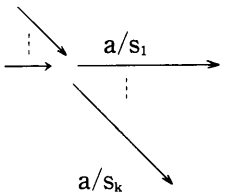
（aは動作，s_iはテープ記号）に対し，

$(1)_{SD}$　SDタイプの状態推移図の復元では，これを一旦



に書き換える。

$(1)_{DS}$　DSタイプの状態推移図の復元では，これを一旦



に書き換える。

(2) 空となったノード全体に，互いに異なる記号を書き込み，これらを〈内部状態〉と読む。

7.2　〈チューリング機械＝状態推移〉の表現

“働いているチューリング機械”は，ヘッドの位置(＝テープのコマ)，内部状態，テープの上の記号例の三つ組——簡単に，“状態”と呼ぶことにする——の変化である。

いま，テープを固定して考え，“テープのコマ全体”の集合をNとする。このとき，チューリング機械のヘッドの基底動作は，

“ヘッドのいまの位置⟶
(ヘッドが上書きするテープ記号，
ヘッドの移動先)”

の読みで関数：

$$N \longrightarrow S \times N$$

と見なせる。よって，動作全体の集合Aは，(S×

$N)^N$である。

また，つぎつぎと変化するテープ記号列の各々は，

“テープのコマ⟼そこに書かれている記号”

の読みで，関数：

$$N \longrightarrow S$$

と見なせる。そしてこのとき，テープ記号列全体の集合はS^Nである。

テープ記号列t，テープのコマn，テープ記号sの組（t, n, s）に

〈tのコマnに書かれているテープ記号を
sに書き換えて得られるテープ記号列〉

を対応させる関数を，Cとする。

さて，現象としてのチューリング機械は，状態推移：

（いまの
ヘッドの位置，
内部状態，
テープ記号列）
$\longrightarrow$
（つぎの
ヘッドの位置，
内部状態，
テープ記号列）

を初期状態から最終状態（停止状態）まで順に追っていくことで，集合$N \times Q \times S^N$の要素の有限列——即ち，自然数の系$\mathbb{N}$の或る切片$K = [0, e]$から集合$N \times Q \times S^N$への関数——として表現できることになる。

この状態推移は，〈チューリング機械＝潜在性〉の表現関数を用いて，つぎの形の表現を得る（ここで，動作を関数：$N \longrightarrow S \times N$と定めたことに留意すること）：

(1) SDタイプの

$$F : Q \longrightarrow (A \times Q)^S$$

に対し

$$(n, q, t) \longrightarrow (\, pr_2(\, (pr_1(\, F(q)(T(n))\,))(n)\,),\ pr_2(\, F(q)(T(n))\,),\ C(\, t, n, pr_1(\, (pr_1(\, F(q)(T(n))\,))(n)\,))$$

(2) DSタイプの

$$F : Q \longrightarrow A \times Q^S$$

に対し

$$(n, q, t) \longrightarrow (\, pr_2(\, (pr_1(F(q)))(n)\,),\ (pr_2(F(q)))(T(\, pr_2(\, (pr_1(F(q)))(n)\,))),\ C(\, t, n, pr_1(\, (pr_1(F(q)))(n)\,))$$

実際，初期状態としての一つの状態（n_0, q_0, t_0）を一旦与えれば，上の規則に従って，状態推移の連鎖が自動的に生成される。これが，

《チューリング機械にテープをセットし，初期条件を与えて起動すれば，チューリング機械は動作表に従って自動的に動く》

ということの定式化である。

8 意味論

テープの上の記号列は，無意味である。

例えば，二進数の掛け算を行なうチューリング機械の実行していることが，二進数の掛け算と読まれる必然性はない。“二進数の掛け算”は，あくまでも，われわれの意味づけによる。——一方，チューリング機械は，記号列の変形に“アルゴリズム”の意味づけをしている。

9 チューリング機械定義プログラム

9.1 動作表に対する“プログラム”の解釈

動作表は，“内部状態”を“アドレス”と読み直すことで，そのまま“実行プログラム”と見なすことができる。——但し，チューリング機械リアライザが受容する実行プログラム（その意味は“チューリング機械定義プログラム”）と

して。

例えば，三つの内部状態0，1，2と二つのテープ記号＿，1の上のSDタイプ動作表：

	＿	1
0	r0	l1
1	r1	l2
2	**	**

は，つぎのプログラムに解釈される：

```
0  switch {
       case ＿ :
           r ; goto 0 ;
       case 1 :
           1 ; goto 1 ;
   }
1  switch {
       case ＿ :
           r ; goto 1 ;
       case 1 :
           1 ; goto 2 ;
   }
2  halt ;
```

逆に，このような形式の動作プログラムで動く機械は，“チューリング機械”となる。

9.2 チューリング機械定義プログラムの二形式

状態推移図および動作推移図に対しては，“チューリング機械の設計図”としての使用が考えられる。即ち，

《状態推移図／動作推移図をチューリング機械定義プログラムに翻訳する》

という使い方である。

このとき，状態推移図から自ずと導びかれるプログラム形式と，動作推移図から自ずと導かれるプログラム形式は，異なる。

実際，状態推移図の翻訳では〈内部状態〉にプログラムの〈アドレス（ラベル）〉が対応し，〈状態推移〉に〈アドレス（ラベル）へのジャンプ〉が対応する。結局，そのプログラムは，アセンブリプログラムやBASICプログラムのようになる。

一方，動作推移図に直結するプログラム形式は，〈アドレス（ラベル）〉無しで済ませるという制約から，《モジュールによる構成》となり，Cプログラムのような“構造化された”プログラムをもたらす。

10 チューリング機械の開発

10.1 “プログラミング”の観点からの“チューリング機械開発”の発想

動作表に対して“チューリング機械定義プログラム”の見方がされるようになるとき，“チューリング機械の開発”が“プログラミング”の観点から発想されるようになる。

10.2 機械語—高級言語，コンパイル

動作表を直接書くことをチューリング機械のプログラミングとするとき，意図通りに動作するチューリング機械をプログラミングすることは，至難の業である。その作業は非常に煩瑣で，見通しの効かないものになる。

動作表を〈プログラム〉として見るとは，それを〈チューリング機械リアライザが受容する実行プログラム〉として見るということであった。そこでいま，動作表の記述を，チューリング機械リアライザが直接受容できる言語——“機械語”——による記述と見て，動作表作

成の困難は機械語に因ると考えてみよう。即ち，“動作表の記述言語”を《リアライザには受容し易いが，われわれには扱い難い機械語》として対象化し，これに，《われわれに扱いやすい言語》として“高級言語”を対置する。

そして，つぎのことを主題化する：

《チューリング機械の記述を高級言語で行ない，この記述を機械語に翻訳（コンパイル）することで，動作表を得る》

10.3 動作推移図とDSタイプの選択

チューリング機械を開発する場合，状態推移図を使うよりは動作推移図を使う方が有利である。

実際，〈アドレス（ラベル）へのジャンプ〉をたどる形でプログラムを書く／読むよりは〈処理〉をたどる形でプログラムを書く／読む方が，われわれの生理に合っている。前者の場合，プログラムはスパゲティプログラムになるが，このようなプログラムはわれわれの生理に合わない。

チューリング機械は，最終的に動作表で表現することになるが，動作表には既にSDとDSの二つのタイプが考えられている。したがって，どちらのタイプを採用すべきかがこのとき問題になる。そして，“状態推移図よりも動作推移図の方が有利”としたいまの場合，DSタイプを選択する方が有利となる。

何故なら，動作表は同タイプの状態推移図の直接の翻訳としてつくられるが，状態推移図への動作推移図の書き直しでは，既に見たように，状態推移図のタイプがDSである場合の方が簡単になるからである（§7.1.4）。

10.4 チューリング機械開発システムの作成

“チューリング機械開発システムの作成”として，具体的につぎのことを行なう：

(1) 動作推移図のダイレクトな翻訳としてチューリング機械定義プログラムを書くことを可能にする言語を作成する。

この言語は，動作表を“チューリング機械リアライザの受容する実行プログラム”と見なし，その記述を“機械語”による記述と見なすとき，機械語に対する高級言語として位置付けられるものである。その意味で，この節では，この言語を単に“高級言語”と呼ぶことにする。

(2) 高級言語で書かれたチューリング機械プログラム（ソースプログラム）からDSタイプの動作表を生成するツール——コンパイラ——を作成する。但し，モジュール別開発を可能にするため，コンパイラはリロケータブル・オブジェクトモジュールを生成するものとし，リロケータブル・オブジェクトモジュールのリンカをコンパイラと併せて作成する。

(3) DSタイプの動作表を，二つの形式で実現する。一つは，チューリング機械リアライザに合った仕様のもの（BIN形式），そしてもう一つは，われわれに合った——書きやすい／読みやすい——仕様のもの（TBL形式）である。

コンパイラ&リンカは，BIN形式とTBL形式の両方の動作表を生成する（但し，TBL形式の動作表の生成は，オプション）仕様とする。

(4) BIN／TBL形式のDSタイプ動作表を受容するチューリング機械リアライザを作成する。

(5) DS タイプ動作表の TBL 形式に対応して，SD タイプ動作表の TBL 形式を定める。

(6) DS タイプの BIN 動作表および TBL 動作表と SD タイプの TBL 動作表の間のコンバータを作成する。

10.5 チューリング機械の開発手順

“チューリング機械開発システム”を使用したチューリング機械の開発は，以下のような流れになる。

(1) 高級言語で，所期のチューリング機械の定義プログラム——ソースプログラム——を書く。

(2) ソースプログラムをコンパイラにかけてリロケータブル・オブジェクトモジュールを得る。

(3) リロケータブル・オブジェクトモジュールをリンカにかけて，BIN 形式の DS タイプ動作表を得る。

(4) チューリング機械リアライザに動作表とテープを与え，リアライザの上にデモンストレートされるチューリング機械の動作を見て，プログラムを——あるいはさらに動作推移図に遡ってこれを——デバグする。

(5) リンカのオプションによって，あるいは，BIN 形式の DS タイプ動作表をコンバータにかけることによって，TBL 形式の DS タイプ動作表を得る。

(6) コンバータによって，DS タイプの BIN／TBL 動作表から SD タイプの TBL 動作表を得る。