3　Ｔｕｒｉｎｇ 機械の定義

[I] 単純な仕掛けによって動作する仮想的な “機械” を想定し、それによって、自然数値関数を “計算” することを考える。

[II] 機械の構成と基本動作

- 機械は１本のテープと１個のヘッドからなる。
- テープは左右に無限に長く、その上に無数の “コマ” が並んでいる。
- それぞれの “コマ” に**文字**がひとつずつ書き込まれている。
- ヘッドはテープ上のひとつの “コマ” に接触している。
- ヘッドは接触している “コマ” に書かれた文字を読む。
- 機械は、後述する**指令**によって以下の２種の動作をする。
  - “コマ” に書かれた文字を書き換える。
  - ヘッドは、左右どちらかにひとコマ動く。これはテープが逆方向にひとコマ移動することと同じである。

[III] 文字・内部状態・時点表示

- テープ上の “コマ” に書かれる文字は、あらかじめ定められた２つ以上有限個、たとえば

  $$0 \quad 1 \quad * \quad \#$$

  の中から選ばれるとする。ここで 0 は空白を表し、とくに指定しない “コマ” には 0 が書かれているものとする。
- 機械の**内部状態**を想定し、小文字のアルファベット

  $$a \quad b \quad c \quad \ldots$$

  で表す。内部状態は可算無限個取り得るとする。
- ある時点での機械全体の様子は、テープ上の文字列、ヘッドの位置、内部状態によって完全に定まる。いま、たとえば

テープ上の文字列	$1\,1*1\,1\,1\,0\,\#\,1*$
ヘッドの位置	左から３番目の $*$ の書かれた “コマ”
内部状態	a

  であるとき、機械全体の様子を

  $$1\,1\,a*1\,1\,1\,0\,\#\,1*$$

  によって表すことにする。このような表示を**時点表示**という。

[IV] 指令と動作、手続き

- x, y を内部状態、 U, V を文字、すなわち $\mathrm{U}, \mathrm{V} \in \{\, 0\,, 1\,, \ast, \#\, \}$ とするとき、

$$x\,\mathrm{U}\,\mathrm{V}\,y, \qquad x\,\mathrm{U}\,\mathrm{L}\,y, \qquad x\,\mathrm{U}\,\mathrm{R}\,y$$

をそれぞれ**書き換え指令**、**左移動指令**、**右移動指令**という。ここで、L、R はテープ上に書かれる文字とは異なるものである。

- これらの指令によって機械の動作が以下のように定義される。
内部状態が x で、ヘッドが接触している “コマ” に書かれている文字が U であるとき、
  - 書き換え指令　$x\,\mathrm{U}\,\mathrm{V}\,y$　は、ヘッドの触れている “コマ” を V に書き換え、内部状態を y とする。
  - 左移動指令　$x\,\mathrm{U}\,\mathrm{L}\,y$　は、ヘッドを左にひとコマ移動し、内部状態を y とする。
  - 右移動指令　$x\,\mathrm{U}\,\mathrm{R}\,y$　は、ヘッドを右にひとコマ移動し、内部状態を y とする。

- 指令による時点表示の変化をまとめると次のようになる:

指　令	実　行　前		実　行　後
$x\,\mathrm{U}\,\mathrm{V}\,y$	$\cdots x\,\mathrm{U}\cdots$	$\longrightarrow$	$\cdots y\,\mathrm{V}\cdots$
$x\,\mathrm{U}\,\mathrm{L}\,y$	$\cdots \mathrm{U}'x\,\mathrm{U}\cdots$	$\longrightarrow$	$\cdots y\,\mathrm{U}'\mathrm{U}\cdots$
$x\,\mathrm{U}\,\mathrm{R}\,y$	$\cdots x\,\mathrm{U}\,\mathrm{V}'\cdots$	$\longrightarrow$	$\cdots \mathrm{U}\,y\,\mathrm{V}'\cdots$

内部状態が x でないか、または、ヘッドの読み取っている文字が U でないときは、これらの指令で時点表示は変化しないものとする。

- **手続き**とは、最初の２文字の組が相異なる有限個の指令からなる集合のことである。たとえば、

$$\mathrm{Z}_1 = \{a\ 1 \ast a,\ \ a \ast \#\, b,\ \ b\,\#\,\mathrm{R}\,c,\ \ c\ 0\ \mathrm{L}\,d\}\,,$$
$$\mathrm{Z}_2 = \{a\ 1 \ast a,\ \ a \ast \#\, b,\ \ b\,\#\,\mathrm{R}\,c,\ \ b\,\#\,\mathrm{L}\,d\}$$

とすると、Z_1 は手続きであるが、Z_2 は手続きではない。

- **チューリング機械**とは手続きのことであるが、以下で定義する**計算**の仕方も込めてチューリング機械ということが多い。

[V] チューリング機械の動作、計算

- 手続き Z と時点表示 T に対して、T を初期値とする Z-**計算**とは、時点表示の列

$$\mathrm{T}_0, \mathrm{T}_1, \cdots, \mathrm{T}_n$$

で、

  - ○ $\mathrm{T} = \mathrm{T}_0$.
  - ○ 各 i $(0 < i \le n)$ に対して、Z に属する指令が存在して、T_{i-1} にその指令をほどこすと T_i が得られる。
  - ○ T_n は、Z に属するどの指令によっても変化しない。

をみたすものである。最後に得られた時点表示 T_n を、Z-計算の値といい $\mathrm{Z}\,(\mathrm{T})$ で表す。

- **例１**

$$\mathrm{Z} = \{a \ast \# \, b, \;\; b \, \# \, \mathrm{R} \, c, \;\; c \; 1 \; \mathrm{L} \; d\}$$

$$\mathrm{T} = 1\;1\;a \ast 1\;1\;1\;0\;1\;1 \ast$$

であるとき、T を初期値とする Z-計算を求めよう: 初期値

$$\mathrm{T}_0 = 1\;1\;a \ast 1\;1\;1\;0\;1\;1 \ast$$

は指令 $a \ast \# \, b$ によって

$$\mathrm{T}_1 = 1\;1\;b \, \# \, 1\;1\;1\;0\;1\;1 \ast$$

に変化する。さらに、指令 $b \, \# \, \mathrm{R} \, c$ によって

$$\mathrm{T}_2 = 1\;1 \, \# \, c \; 1\;1\;1\;0\;1\;1 \ast$$

となり、指令 $c \; 1 \; \mathrm{L} \; d$ によって

$$\mathrm{T}_3 = 1\;1\;d \, \# \, 1\;1\;1\;0\;1\;1 \ast$$

となる。Zが $d \, \#$ で始まる指令を含まないので Z-計算はここで停止し、

$$\mathrm{Z}\,(\mathrm{T}) = \mathrm{T}_3$$

となる。

- **例２**　手続き Z を

$$\mathrm{Z} = \{a * \mathrm{R}\, a,\ \ a \# \mathrm{R}\, a,\ \ a\, 0\, \mathrm{R}\, a,\ \ a\ 1\ 1\ b\}$$

と定義すれば、時点表示

$$a * 0\, 0 \# * 1 *$$

に対して

$$\begin{aligned} & a * 0\, 0 \# * 1 * \\ \longrightarrow\ & * a\, 0\, 0 \# * 1 * \ \longrightarrow\ * 0\, a\, 0 \# * 1 * \\ \longrightarrow\ & * 0\, 0\, a \# * 1 * \ \longrightarrow\ * 0\, 0 \# a * 1 * \\ \longrightarrow\ & * 0\, 0 \# * a\, 1 * \ \longrightarrow\ * 0\, 0 \# * b\, 1 * \end{aligned}$$

となって終了する。
しかし、別の時点表示を初期値とすると、たとえば

$$\begin{aligned} & a * 0\, 0 \# \ \longrightarrow\ * a\, 0\, 0 \# \ \longrightarrow\ * 0\, a\, 0 \# \\ \longrightarrow\ & * 0\, 0\, a \# \ \longrightarrow\ * 0\, 0 \# a \ \longrightarrow\ * 0\, 0 \# 0\, a \\ \longrightarrow\ & * 0\, 0 \# 0\, 0\, a \ \longrightarrow\ * 0\, 0 \# 0\, 0\, 0\, a \ \longrightarrow\ \cdots \end{aligned}$$

のように操作は無限に続き、終了することがない。