15　チューリング機械リアライザ

15.1　実行プログラムとしてのチューリング機械リアライザ

われわれは，一つの実行プログラムによって，コンピュータ上にチューリング機械を実現す

る。そしてこの実行プログラムを，〈チューリング機械リアライザ〉と位置づける。(この実行プログラムが動いているコンピュータ全体も，チューリング機械リアライザである)。

チューリング機械のこの実現は，“コンピュータによるチューリング機械のシミュレーション”ではない。実際，“チューリング機械”というものが存在しているわけではない——“チューリング機械“は解釈であり，“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る。

15.2 チューリング機械リアライザの機能

ここでは，チューリング機械リアライザの機能をつぎのように考える：

(1) チューリング機械定義ファイル(動作表ファイル)を読み込み，オフ状態でしかもテープがまだセッティングされていない段階のチューリング機械——メモリボードを装着していない電源オフのコンピュータに相当——を実現する。

(2) つぎにこのオフ状態において，“テープのセッティング”を，テープファイルの読み込み，あるいは文字列の直接入力を受けつけるという形で，実現する。これで，電源を入れる前のコンピュータに相当するチューリング機械が実現されたことになる。

(3) キー入力に反応して，チューリング機械を起動する。いままでは見えていなかったチューリング機械が，〈テープ入力に対するチューリング機械の反応〉という形で顕在化する。

15.3 チューリング機械リアライザ TU

われわれは，チューリング機械リアライザ——兼デバッガ（チューリング機械定義プログラムの）——として，実行プログラム

TU. EXE

を用意する。

TUは，BINファイル／TBLファイルを読み込み，これに記述されているチューリング機械を実現する。即ち，設定されたテープに対するこれの反応——初期状態から停止状態に至るまでのヘッドの移動およびテープ上の記号列の変化——を表示する。

デバッガとしての用途のために，トレース機能も付加している。

15.4 TU起動の書式

TUの起動の書式は，

TU [-e〈エディタ名〉]
[-f〈動作表ファイル名〉]
[-t〈テープファイル名〉]

である。

〈エディタ名〉は，TUの子プロセスとしてファイル編集するときに使用するエディタの名である。

〈動作表ファイル名〉では，拡張子を省略できる。省略されているとき，TUは拡張子がBINであるとしてファイルを検索する。見つからなければ，拡張子がTBLの名で検索する。

〈テープファイル名〉は，入力テープを定義するファイルの名である。拡張子——テープファイルの拡張子はTAP——を省略できる。

以上のファイル名は，パス名で指定する。

なお，ユーザは，バッチファイルを作成することにより，オプション指定の煩瑣を回避することができる。

例：

```
tu_. bat
    tu -eVZ -fADD -t5&8
```

TUは，指定された動作表ファイル／テープファイルが見つかれば，これを読み込む。

指定の動作表ファイル／テープファイルを見出せなかった場合，あるいは動作表ファイル／テープファイルがはじめから指定されていなかった場合，TUは初期画面において，欠けているファイルの指定を待つ。指定が得られ，かつ指定のファイルが見出せれば，これを読み込む。

TUは，最初にロードした動作表ファイルから，テープ記号を決定する。

15.5 動作表のファイル形式

15.5.1 TBLとBIN

動作表ファイル（チューリング機械定義ファイル)は，“動作表”のように見える必要はない。動作表ファイルについて肝心なことは，われわれにとってそれの可読性ではなく，チューリング機械リアライザにとってそれの可読性である。

われわれにとっての読みやすさと，リアライザにとっての読みやすさとは，同じではない。そこでわれわれは，動作表ファイルを二通りに定めることにする。

一つは，われわれにとっての可読性を優先した仕様であり，DSタイプの動作表をそのまま記述したテクストファイルである。そしてもう一つは，リアライザにとっての可読性（処理の効率性）とファイルのコンパクト性を優先した仕様のバイナリファイルである。この二種類の仕様ファイルは，それぞれ拡張子TBLとBINによって区別する。

チューリング機械リアライザTUは，この二種類のファイルを動作表ファイルとして受容できる。

BINファイルはMファイルのコンパイルによってつくられる。TBLファイルは，以下のいずれかの方法でつくられる：

(1) 直接エディタを用いてつくる；
(2) BINファイルをコンバータにかける；
(3) リンカの-tオプションで，BINファイルと同時に生成する。

TBL形式の動作表を受容できるようにしているのは，任意の動作表がコンパイル&リンクの方法で生成できるわけではなく，また，コンパイル&リンクの方法による動作表の作成が“最適化”の問題を孕んでいるからである。

15.5.2 TBLファイルの仕様

TBLファイルの仕様を，以下に述べる。なお，“行”は，改行コードで定義される“行”を意味するものとする。

リアライザは，行の先頭に書かれたーを，区切り記号として認識する。かつ，区切り記号のある行——以下，“区切り行”と呼ぶ——は，スキップして読み込まない。

区切り行は，三つ設けられる。したがって，TBLファイルの内容は，四つの領域でなる。

リアライザは，一番目と四番目の領域を，スキップして読み込まない。したがってこの領域を，表題/注釈の記述に使用できる。またこの二つの領域は，空でもよい。

第二および第三の領域の各行においては，スペースあるいはタブでなる記号列が，データの区切り記号になる。——即ち，リアライザは，スペース/タブ列をスキップして読み込まず，また，スペース/タブと異なる文字に出会うと，つ

ぎのスペース/タブに出会うまでの記号列——以下これを単に“記号列”と呼ぶ——をデータとして読み込む。

第二領域は一行でなり，リアライザはこれを，〈テープ記号〉の列挙と認識する。テープ記号は1バイトの文字でなければならず，スペース/タブ列で区切って書く。

第三領域の各行は，〈内部状態〉，〈動作〉，そしてテープ記号に対応する〈状態推移〉の記述である。

最終行の〈内部状態〉を，〈終了（停止）状態〉として固定する。これに対する〈動作〉，〈状態推移〉は，すべて＊とする。

例：

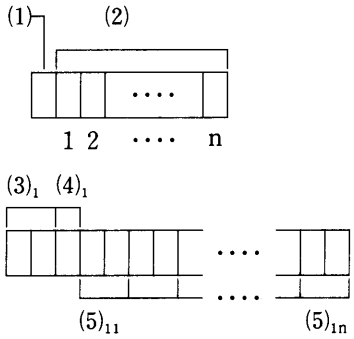
lb. tbl

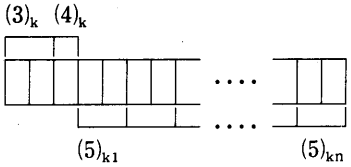
空白（“無記号”）に出会うまで左移動

```
--------------------------------
                  _     1
--------------------------------
    q0     l      q1    q0
    q1     *      *     *
--------------------------------
```

15.5.3 BIN ファイルの仕様

BIN ファイルの仕様をつぎのように定める（1マスが1バイト）：





(1) テープ記号の数：

1バイトの整数

(2) テープ記号：

テープ記号（キャラクターコード）

（但し，r，l，e以外）

$(3)_i$ i番目の内部状態（アドレスに表現）：

2バイトの整数

$(4)_i$ i番目の内部状態に対する動作：

右移動：文字r（キャラクタコード）

左移動：文字l（キャラクタコード）

消去　：文字e（キャラクタコード）

書込　：テープ記号（キャラクタコード）

$(5)_{ij}$ i番目の内部状態に対応する動作の後，j番目のテープ記号に対応して推移する先の内部状態（アドレス）：

2バイトの整数

15.5.4 TB0ファイル

SD タイプの動作表のファイル形式を，以下のように定める——ファイルの拡張子は，TB0とする。

ファイル形式は，TBL ファイルの場合に準じ，第三領域の書き方のみ異なる。

即ち，第三領域の各行において，最初の記号列は〈内部状態〉であり，そしてこれの後に〈動作—状態推移〉の列挙が続く。

最終行を除いては，各〈動作—状態推移〉は〈動作〉の1バイト文字と推移先の〈内部状態〉の記号列の連結である。

最終行の〈内部状態〉を，〈終了(停止)状態〉として固定する。これに対する〈動作—状態推移〉は，すべて記号列＊＊とする。

例：

lb. tb0

空白（“無記号”）に出会うまで左移動

```
--------------------------------
          _       1
--------------------------------
    q0      lq1     lq1
    q1      _q2     lq1
    q2      **      **
--------------------------------
```

15.6 ファイルコンバータ

われわれはユーティリティとして，BIN, TBL, TB 0 三種類のファイルの間のコンバータ

CONV. EXE

を用意する。

コンバートの方向は，つぎのオプションによって指定する：

—xy

ここで，x，yは，

BIN を表わすb

TBL を表わすt

TB0 を表わす0

のいずれかで，オプションの意味は“xからyへのコンバート”である。

BIN と TBL の間のコンバートは，データ型の変換に過ぎない。

TBL から TB0 へのコンバートは，つぎのようにする。即ち，

		s_1 ···· s_k
⋮	⋮	····
q	a	q_1 ···· q_k
⋮	⋮	····

の各行に対し

		s_1 ···· s_k
⋮	⋮	····
q	a	r ···· r
⋮	⋮	····
r	null	q_1 ···· q_k

の処置をする。そして

q	a	r ···· r

の形の行を

q	ar ···· ar

に書き換え，

q	null	q_1 ···· q_k

の形の行を

q	$s_1 q_1$ ···· $s_k p_k$

に書き換える。

逆に，TBO から TBL へのコンバートは，つぎのようにする。即ち，

	s_1 ···· s_k
⋮	····
q	$a_1 q_1$ ····$a_k q_k$
⋮	····

の各行に対し，

	s_1 ···· s_k
⋮	····
q	$s_1 r_1$ ···· $s_k r_k$
⋮	····
r_1	$a_1 q_1$ ···· $a_1 q_1$
⋮	····
r_k	$a_k q_k$ ···· $a_k p_k$

の処遇をする。そして

$$q \mid s_1 q_1 \quad \cdots\cdots \quad s_k q_k$$

の形の行を

$$q \mid \text{null} \mid q_1 \quad \cdots\cdots \quad q_k$$

に書き換え，

$$q \mid \quad ar \quad \cdots\cdots \quad ar$$

の形の行を

$$r \mid a \mid q \quad \cdots\cdots \quad q$$

に書き換える。

15.7 テープ

15.7.1 テープの書式

テープの書式は次の通りである：

(1) テープの記述では，動作表ファイルで指定されているテープ記号と矛盾しない限り，任意の文字が使える。

動作表ファイルの指定するテープ記号と矛盾するとき，TUはエラーメッセージを表示する。このとき，テープと動作表のうちエラーとして捨てられるのは，後から入力した方である。

(2) コンマは，〈ヘッド記号〉として予約されている。即ち，

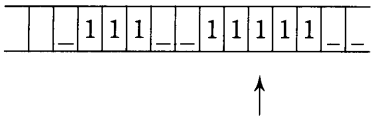
《コンマの直後の記号が，初期状態でヘッドの指している記号である》

となる。

例えば，

＿１１１＿＿１１，１１１

は，



を意味する。

(3) テープの記述に，〈ヘッド記号〉が欠けていたり，〈ヘッド記号〉が複数個あるのは，エラー。このときは，エラーメッセージが表示される。

(4) テープ記号の読み込みにおいて，空白はスキップされる。

(5) テープの記述は，テープの左端からの記述とみなされる。

(6) TUは，記号列の右端から以降を“無記号”記号(＿)が書き込まれているものと判断し，トレースの際必要に応じてこの記号を補う。

15.7.2 テープの入力

テープ入力の方法はつぎの二通りである。

(1) あらかじめ入力式を書き込んだテクストファイルを拡張子TAPでつくり，TUに読み込ませる。ファイルの指定は，TUを起動するときのコマンドラインか，TUのファンクションキーメニューによる。

(2) TUのファンクションキーメニューからテープ入力モードを選択し，直接入力する。