2013年5月31日

Lispプログラミング入門

西田豊明

今回あらすじ

1. Lispの実践的な使い方を学習する.

2. Lispインタープリタの動かし方，電卓的使い方，関数定義，条件分岐，S式の基本操作，プログラミング手法，プロトタイピング法などを中心に解説する.

3. チューリングマシンのシミュレータの構成方法を例にプログラミングの実際を学習する.

Lispプログラミングの流れ

「私の人工知能理論」

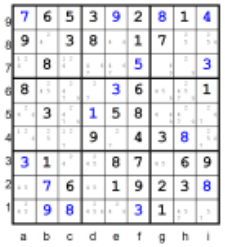
表現

人工知能プログラム

解釈実行

インタープリタ

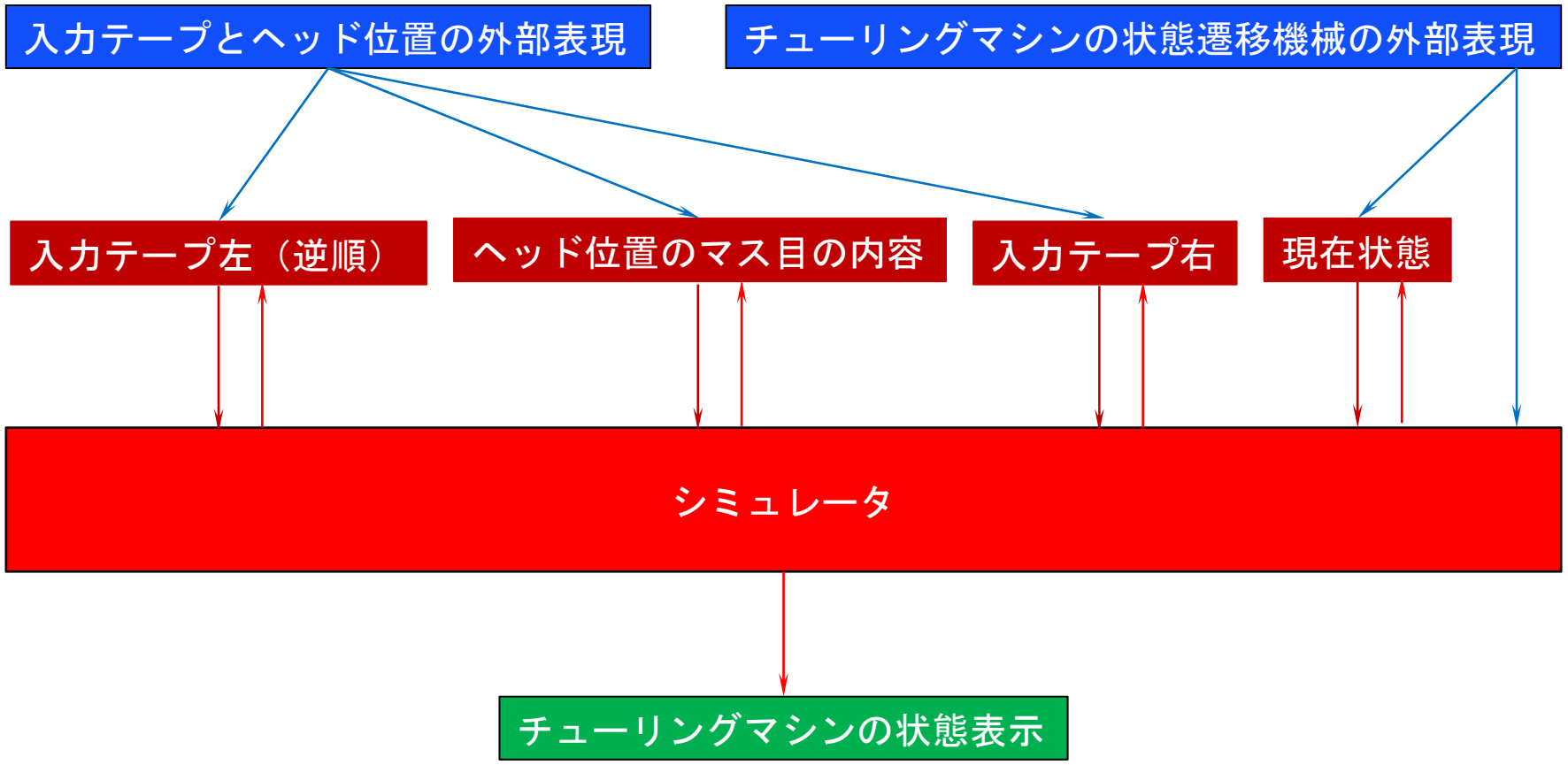
表示



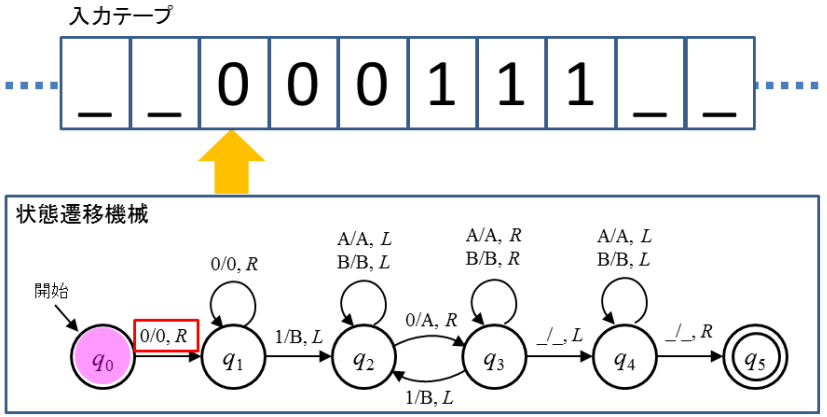
Gnu Common Lisp

チューリングマシンの場合

情報の流れ



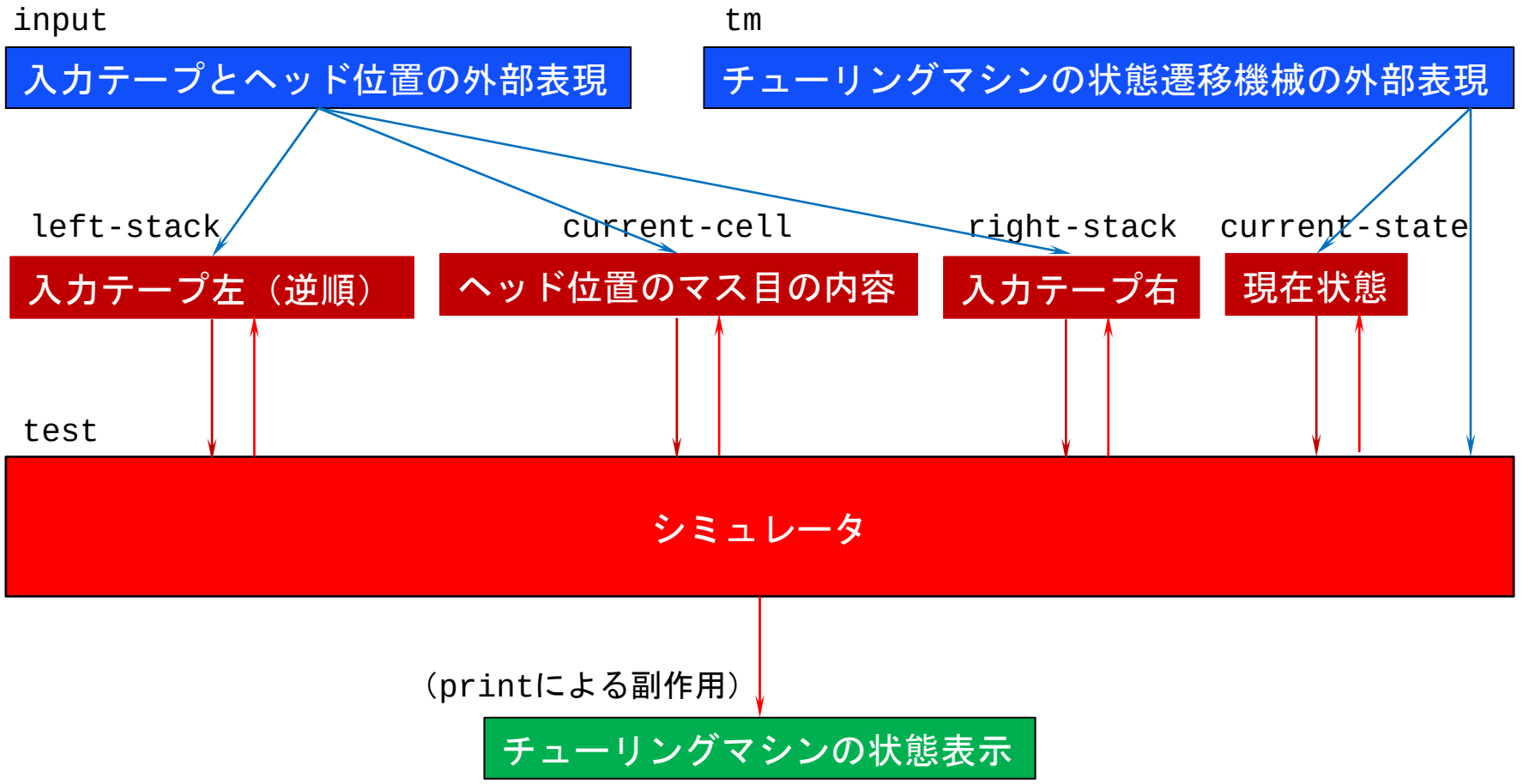
チューリングマシンの場合




```
((start q0)      … 初期状態がq0である
 (terminal q5)     … 停止状態はq5である.
 (states               … 以下では，各状態における状態遷移規則を表す
   (q0 (0 (Right) q1))     … 状態q0で，0を見たらヘッドを右に移動し，状態q1に遷移
   (q1 (0 (Right) q1)      … 状態q1で，0を見たらヘッドを右に移動し，状態q1に遷移
       (1 B q2))           … 状態q1で，1を見たらBと書き換え，状態q2に遷移
   (q2 (A (Left) q2)       … 状態q2で，Aを見たらヘッドを左に移動し，状態q2に遷移
       (B (Left) q2)       … 状態q2で，Bを見たらBと書き換え，状態q2に遷移
       (0 A q3))           … 状態q2で，0を見たらAと書き換え，状態q3に遷移
   (q3 (1 B q2)            … 状態q3で，1を見たらBと書き換え，状態q2に遷移
       (A (Right) q3)      … 状態q3で，Aを見たらヘッドを右に移動し，状態q3に遷移
       (B (Right) q3)      … 状態q3で，Bを見たらヘッドを右に移動し，状態q3に遷移
       (_ (Left) q4))      … 状態q3で，空白_を見たらヘッドを左に移動し，状態q4に遷移
   (q4 (A (Left) q4)       … 状態q4で，Aを見たらヘッドを左に移動し，状態q4に遷移
       (B (Left) q4)       … 状態q4で，Bを見たらヘッドを左に移動し，状態q4に遷移
       (_ _ q5))))         … 状態q4で，空白_を見たら，状態q5に遷移
```

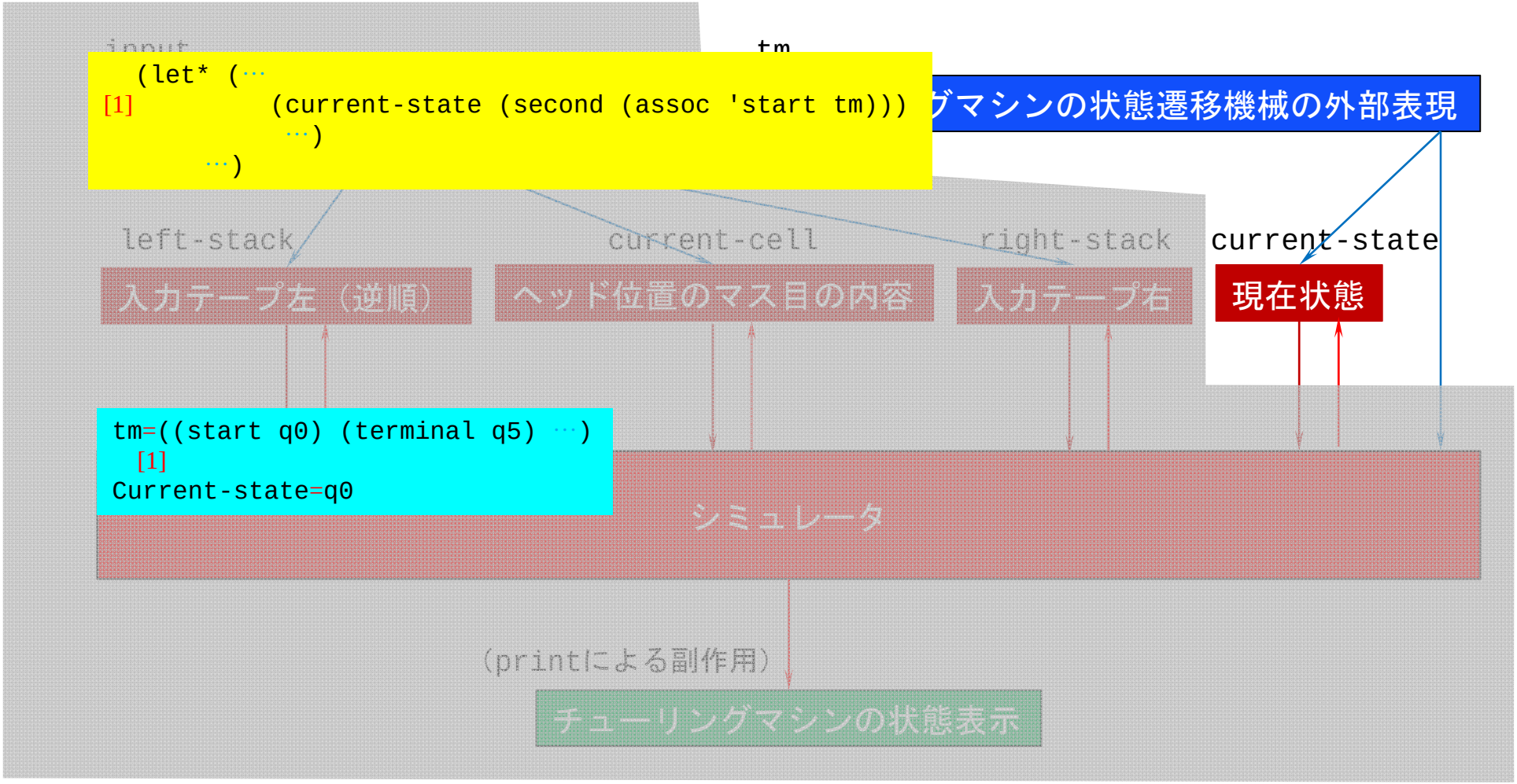
チューリングマシンの場合

変数



チューリングマシンの場合

準備１



チューリングマシンの場合

準備２

```
[1](setq right-stack (cdr (member '* input)))
[2](cond (right-stack
[2]            (setq current-cell (car right-stack))
[2]            (setq right-stack (cdr right-stack)))
[2]      (t  (setq current-cell '_)))
[3] (dolist (cell input)
[3]     (cond ((eq cell '*) (return)))
[3]     (push cell left-stack))
```

input

入力テープとヘッド位置の外部表現

left-stack

入力テープ左（逆順）

current-cell

ヘッド位置のマス目の内容

right-stack

入力テープ右

current-state

現在状態

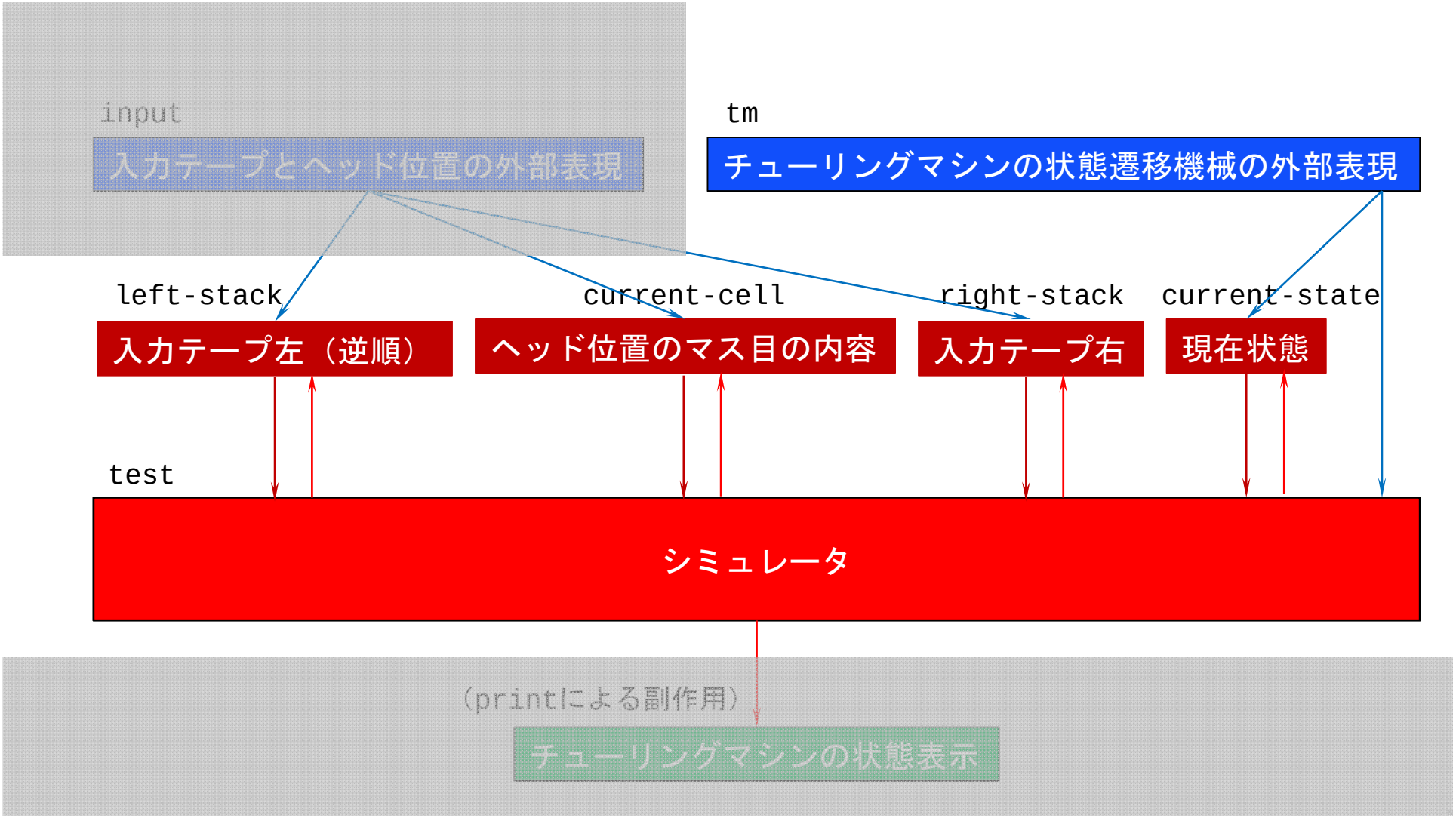
```
input=(1 2 3 * 4 5 6 7); left-stack=(); current-cell=NIL; right-stack= ()
  [1]
right-stack=(4 5 6 7)
  [2]
current-cell=4; right-stack=(5 6 7)
  [3]
left-stack=(3 2 1)
```

（printによる副作用）

チューリングマシンの状態表示

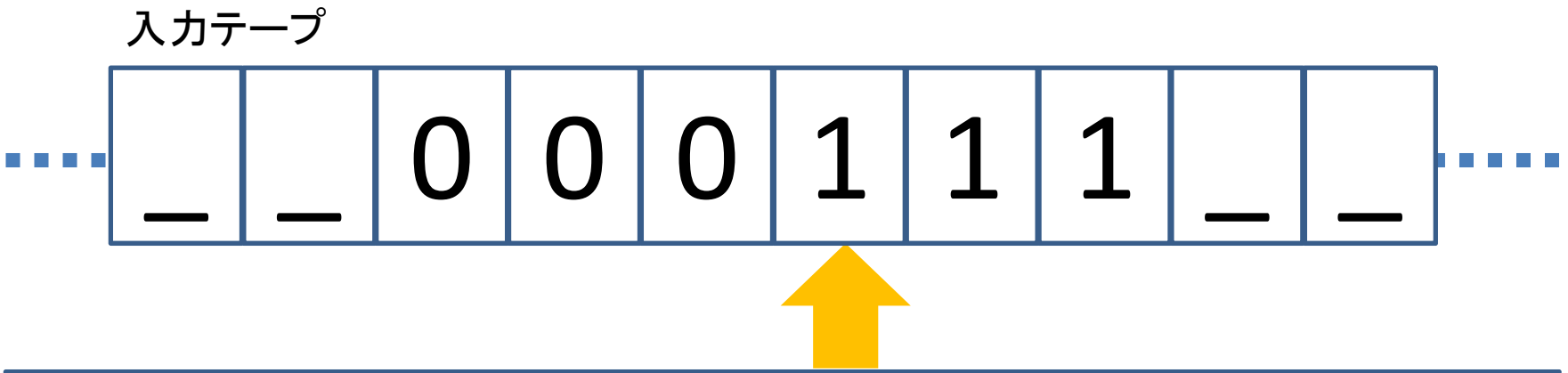
チューリングマシンの場合

中心部分



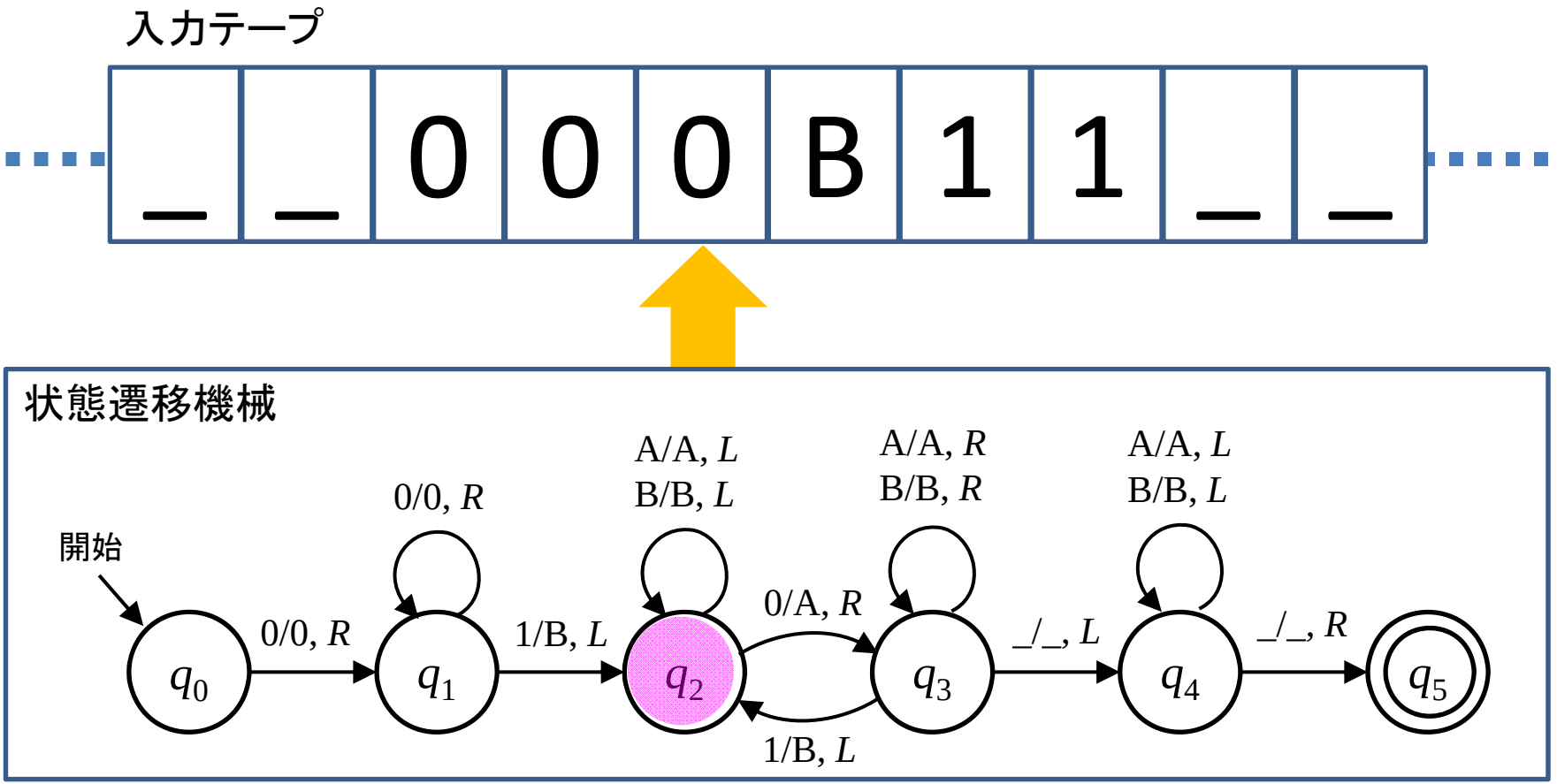
チューリングマシンの場合

典型的な状況例



チューリングマシン

典型的な状況例



チューリングマシンの場合

中心部分

tm

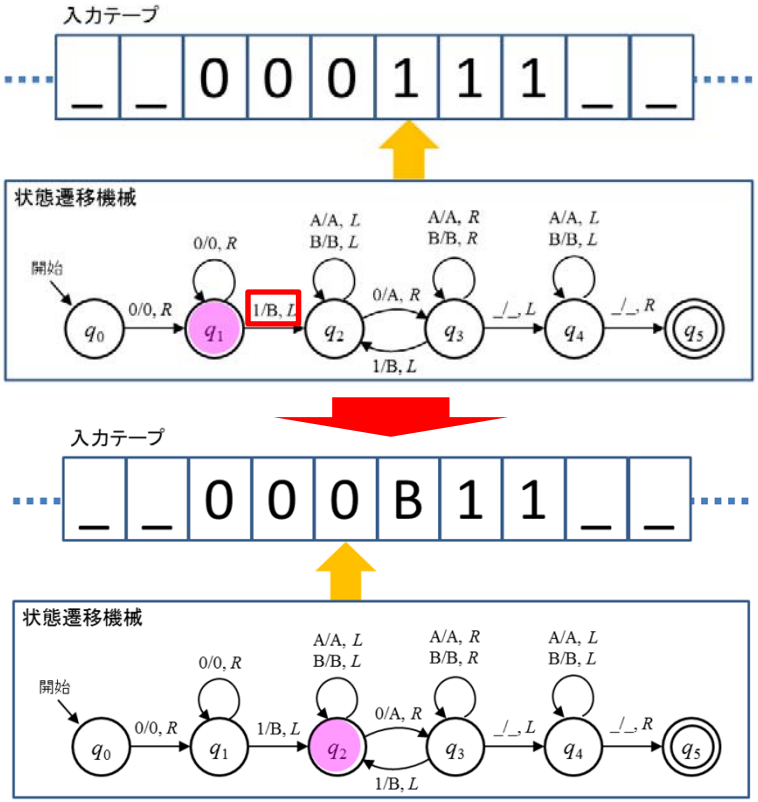
チューリングマシンの状態遷移機械の外部表現

left-stack　current-cell　right-stack　current-state

入力テープ左（逆順）　ヘッド位置のマス目の内容　入力テープ右　現在状態

test

シミュレータ

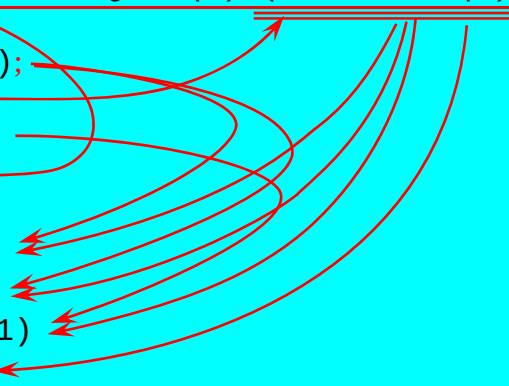



```
tm=( ··· (states
          ···(q1 (0 0 right q1) (1 B left q2)) ···) ···)
···
left-stack=(0 0 0);
current-cell=1;
right-stack=(1 1)
current-state=q1
  [?]
left-stack=(0 0);
current-cell=0;
right-stack=(B 1 1)
current-state=q2
```

チューリングマシンの場合

中心部分

```
tm=( … (states
          …(q1 (0 0 right q1) (1 B left q2)) …) …)
…
left-stack=(0 0 0);
current-cell=1;
right-stack=(1 1)
current-state=q1
  [?]
left-stack=(0 0);
current-cell=0;
right-stack=(B 1 1)
current-state=q2
```



```
[1] (setq rules (cdr (assoc current-state states)))
[2] (setq rule (assoc current-cell rules))
[3] (cond ((null rule) (return)))
[4] (setq current-cell (second rule))
[5] (setq current-state (fourth rule))
[6] (case (third rule)
[7]   (right (cond
[7]             (right-stack
[7]                  (setq left-stack `(,current-cell . ,left-stack))
[7]                  (setq current-cell (car right-stack))
[7]                  (setq right-stack (cdr right-stack)))
[7]             (t   (setq left-stack `(,current-cell . ,left-stack))
[7]                  (setq current-cell '_))))
[7]   (left  (cond
[7]             (left-stack
[7]                  (setq right-stack `(,current-cell . ,right-stack))
[7]                  (setq current-cell (car left-stack))
[7]                  (setq left-stack (cdr left-stack)))
[7]             (t (setq right-stack `(,current-cell . ,right-stack))
[7]                  (setq current-cell '_))))
```

チューリングマシンの場合

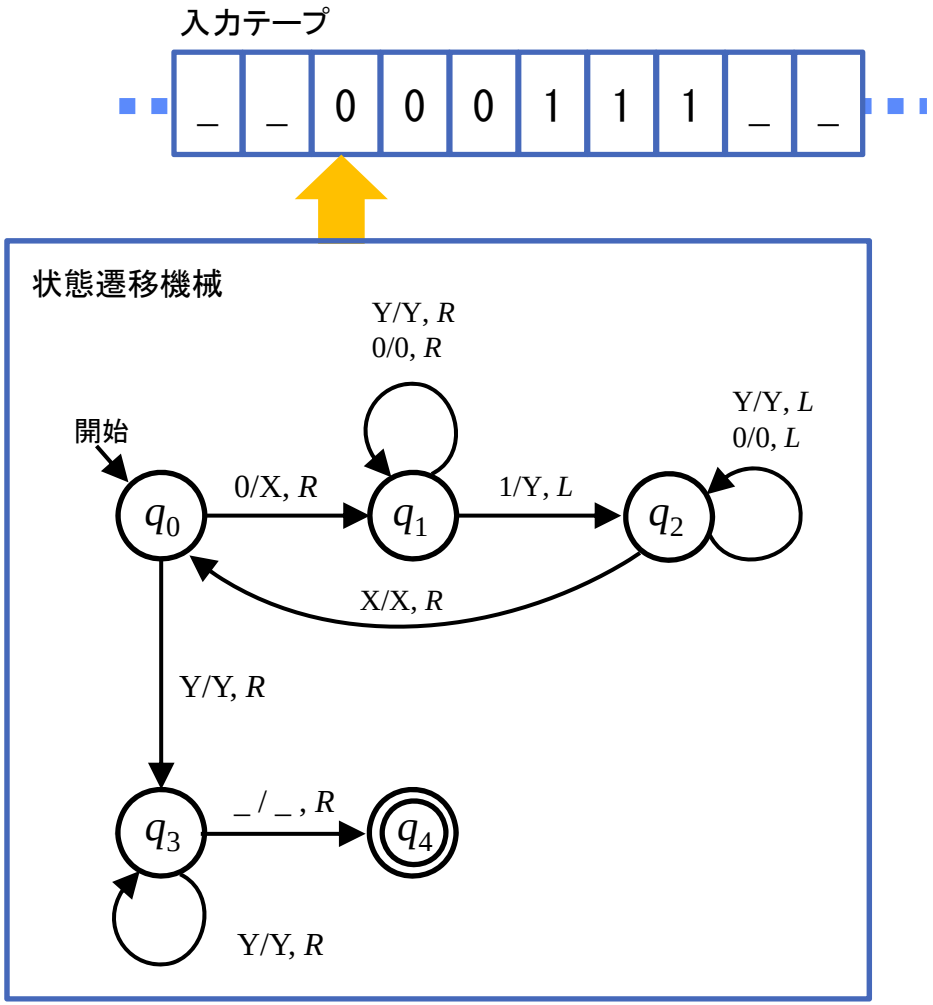
プログラムtest

```
(defun test ()
  (let* ((tm tm0)
         (input input-tm0)
         (left-stack nil)
         (right-stack)
         (current-state (second (assoc 'start tm)))
         (current-cell nil)
         (states (cdr (assoc 'states tm)))
         (terminal-states (cdr (assoc 'terminal tm)))
         (rules nil)
         (rule nil)
         (counter 1)
         (credits 1))
    (setq right-stack (cdr (member '* input)))
    (cond (right-stack
             (setq current-cell (car right-stack))
             (setq right-stack (cdr right-stack)))
          (t  (setq current-cell '_)))
    (dolist (cell input)
       (cond ((eq cell '*) (return)))
       (push cell left-stack))
    (loop
      (cond ((<= credits counter)
                (princ "How many more steps to go?") (terpri)
                (setq credits (read))
                (cond ((<= credits 0) (return))
                      (t (setq counter 0))))
            (t (setq counter (1+ counter))))
      (princ "====================================") (terpri)
      (print-state left-stack current-cell right-stack current-state)
      (cond ((member current-state terminal-states)
                (princ "====================================") (terpri)
                (princ "*** Halt in a termnial state ***") (terpri)
                (princ "====================================") (terpri)
                (return)))
       ★
)))
```

```
[1] (setq rules (cdr (assoc current-state states)))
[2] (setq rule (assoc current-cell rules))
[3] (cond ((null rule) (return)))
[4] (setq current-cell (second rule))
[5] (setq current-state (fourth rule))
[6] (case (third rule)
[7]   (right (cond
[7]             (right-stack
[7]                  (setq left-stack `(,current-cell . ,left-stack))
[7]                  (setq current-cell (car right-stack))
[7]                  (setq right-stack (cdr right-stack)))
[7]             (t   (setq left-stack `(,current-cell . ,left-stack))
[7]                  (setq current-cell '_))))
[7]   (left  (cond
[7]             (left-stack
[7]                  (setq right-stack `(,current-cell . ,right-stack))
[7]                  (setq current-cell (car left-stack))
[7]                  (setq left-stack (cdr left-stack)))
[7]             (t (setq right-stack `(,current-cell . ,right-stack))
[7]                  (setq current-cell '_))))
```

ホームワーク

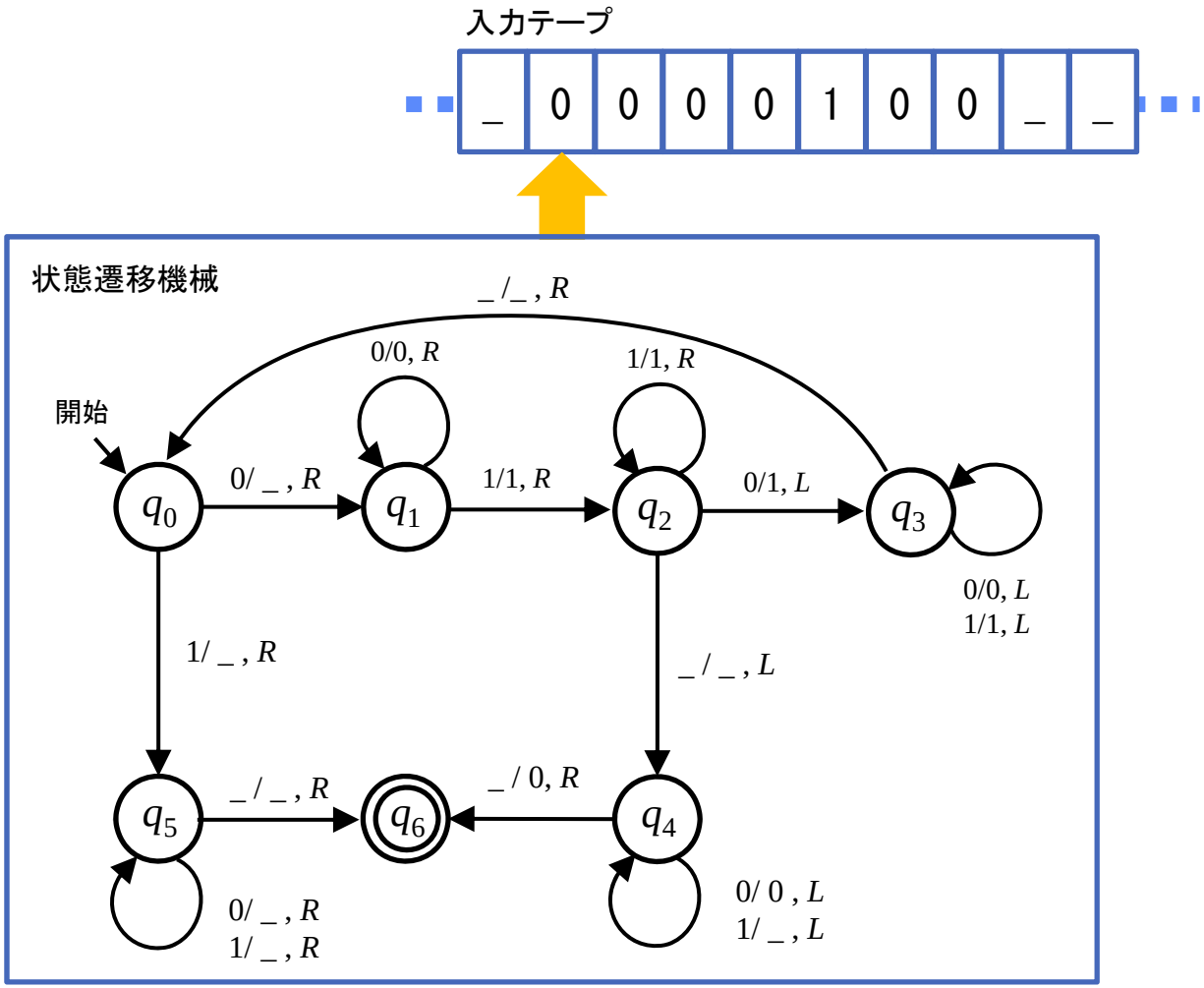
次のチューリングマシンの動作をシミュレートせよ



[Hopcroft 2001]

ホームワーク

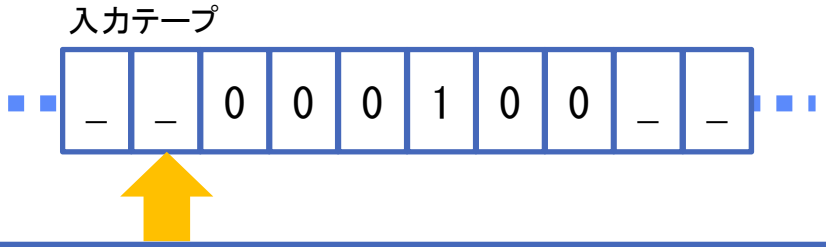
次のチューリングマシンの動作を分析せよ



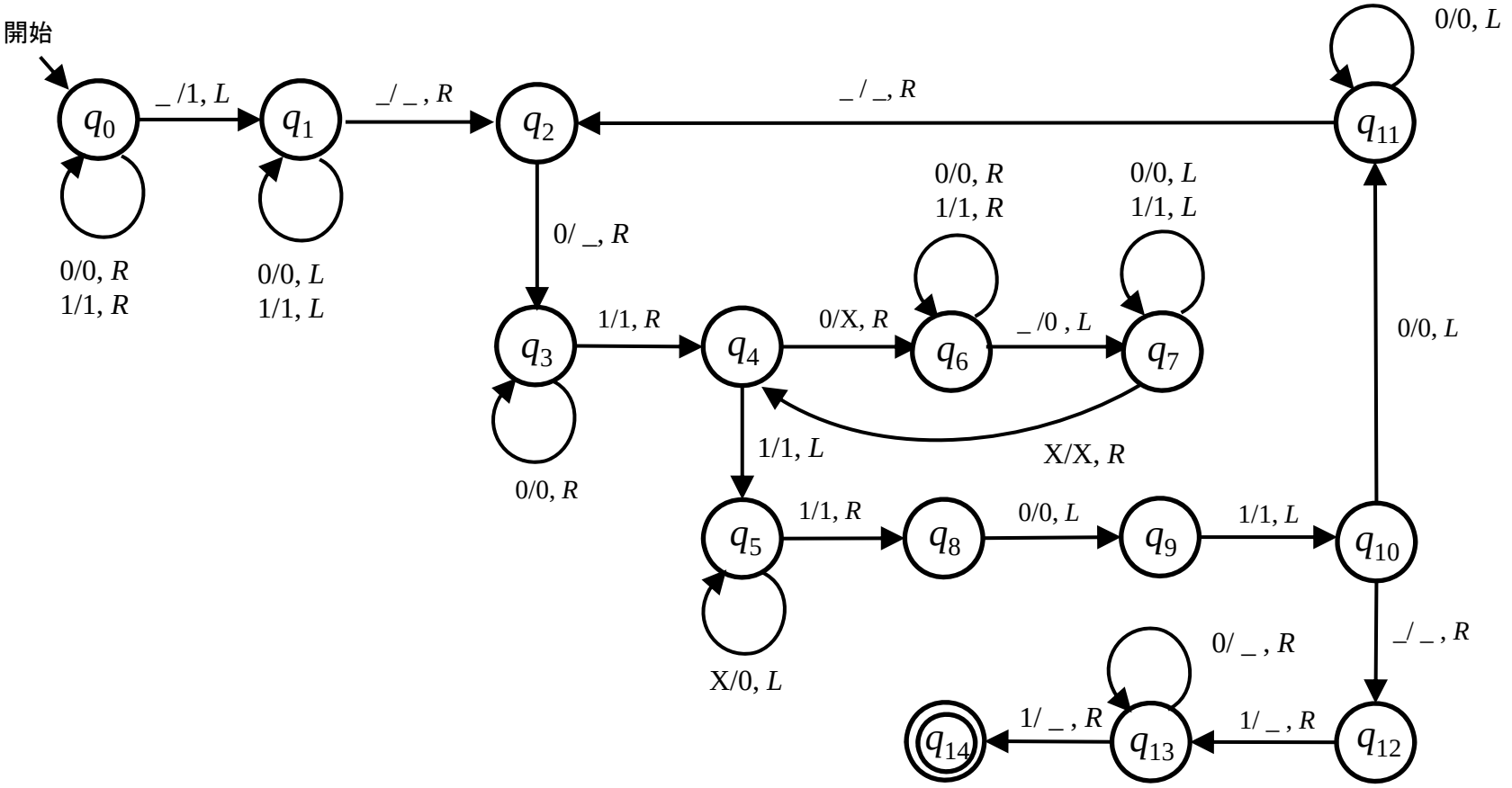
[Hopcroft 2001]

ホームワーク

次のチューリングマシン
の動作を分析せよ



状態遷移機械



[Hopcroft 2001]