オートマトンと計算理論

第３部　チューリング機械と計算理論

火曜５・６限目　必修科目

尾張　正樹

居室：J2415 (情報２号館４階)

masakiowari@inf.shizuoka.ac.jp

講義資料：¥¥fs.inf.in.shizuoka.ac.jp¥share¥class¥2018オートマトン

第３部
チューリング機械と計算理論

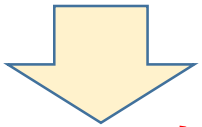
第五章：
チューリング機械とO型文法

最強のオートマトンと
　対応する　最も強力な文法を導入する。

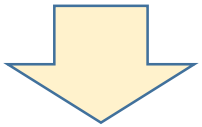
1. チューリング機械
2. O型文法
3. チューリング機械によっても認識できない言語

1．チューリング機械(1)

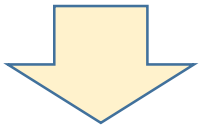
プッシュダウンオートマトンの性能の限界は、スタックが先頭しか読めないことから来ていた。



よって、先頭以外も読める補助メモリをオートマトンに付けると、能力のより強いオートマトンができると思われる。



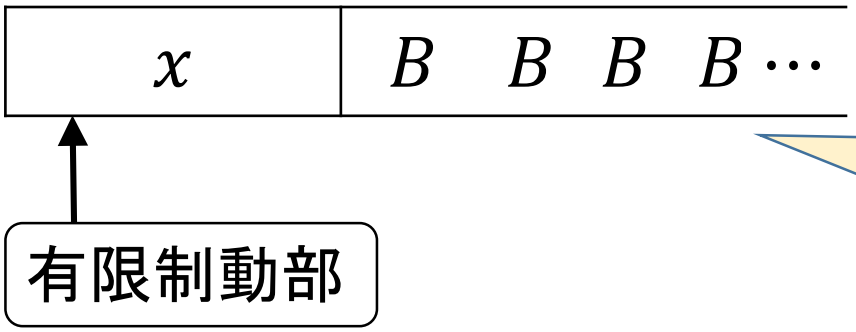
更に、メモリを書き換えられれば、より便利だと思われる。



しかし、そこまでやるなら、そもそも入力テープを読み書き可能にすればよい。

1. チューリング機械 (2)

チューリング機械のイメージ



テープは右方法には無限に伸びていて、そこに空白記号Bが書かれている。

faとの違い:

(i) テープは読むだけでなく、書くこともできる。
つまり、今のヘッドの下の記号を他の記号に書き換えれる。

(ii) ヘッドは左にも動ける。（これがないと(i)の意味が無くなる。）

(iii) テープは右方向に際限なく伸びている。（入力列xより右は空白を示す記号Bが書かれている）

1. チューリング機械 (3)

チューリング機械の1ステップの動作:

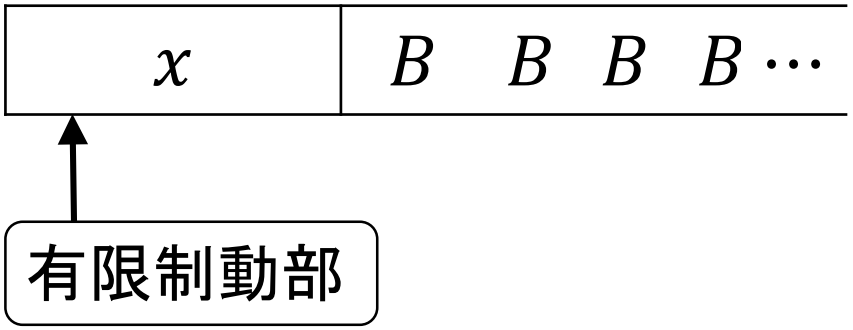
(a) テープのヘッドの下の記号を読む。

読んだ記号と、現在の状態に応じて

(b) 状態を変える。

(c) ヘッドの下の記号を書き換える。

(d) ヘッドを1コマ右または左に動かす。



1. チューリング機械 (4)

定義5.1
チューリング機械（TM）は$M = (K, \Sigma, \Gamma, \delta, s_0, B, F)$で表される。

K: 状態　の集合
Σ: 入力記号　の集合
$\Gamma \supset \Sigma$: テープ記号　の集合　　(入力記号はテープ記号)
$s_0 \in K$: 初期状態
B: 空白 (特別なテープ記号、入力記号ではない)　$B \in \Gamma - \Sigma$
$F \subset K$: 受理状態　の集合
（K, Σ, Γ, F は全て有限集合）

δ:　$K \times \Gamma \to K \times \Gamma \times \{L, R\}$　　状態遷移関数

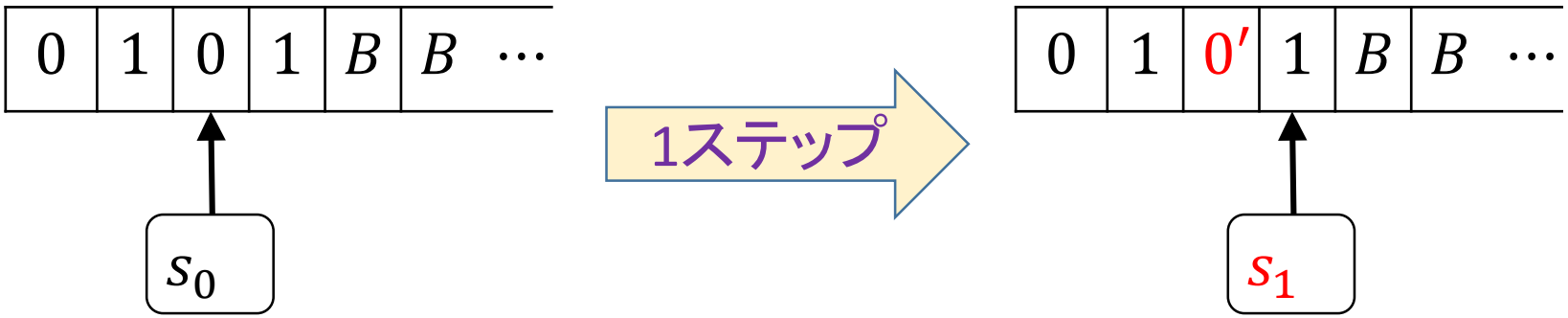
Lは『左』、Rは『右』を意味する。

1. チューリング機械 (5)

δ: $K \times \Gamma \to K \times \Gamma \times \{L, R\}$　状態遷移関数

例えば、$\delta(s_0, 0) = (s_1, 0', R)$ は

- 状態s_0でテープのヘッドの下の記号が0ならば、
- 次の状態はs_1で、ヘッドの下の記号を$0'$に書き換え、
- ヘッドを1コマ右に動かす　　ことを意味する。



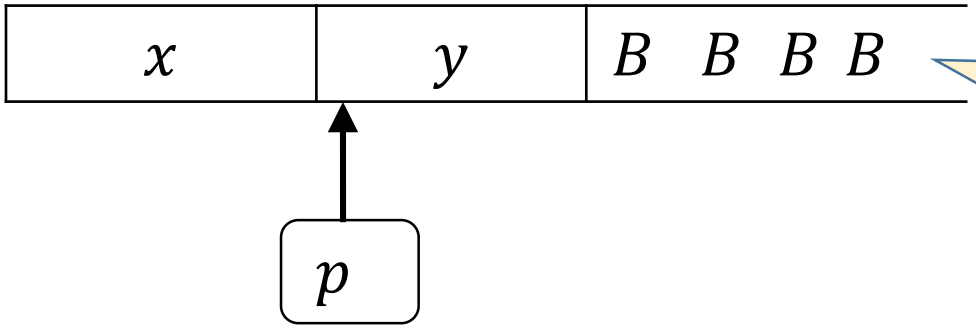
空白記号Bを書き込むことはできないとする。

1.チューリング機械 (7)

定義： 状態$p \in K$、テープ記号の列$x, y \in \Sigma^*$としたとき、三つ組(p, x, y)を様相という。

Mの様相が(p, x, y)である。

⇔ 現在、状態はpで、
テープの左端からヘッドのあるマス目の左隣りまでが列xで、
ヘッドのあるマス目を含んで、それより右で空白Bが初めて現れる直前までが列y



空白Bを書き込むことはできないので、**ある駒がBなら、それより右は全てB**

1.チューリング機械 (8)

pdaの場合と全く同様に、『1ステップで移れる』と『何ステップかで移れる』を定義する。
（正式な定義はpdaよりもさらに複雑になるので書かない）

様相C_1とC_2に対して:

$C_1 \Rightarrow C_2$: TMがC_1からC_2に1ステップで移れる
つまり、C_1のとき状態遷移関数δを適用するとC_2になる。

$C_1 \overset{*}{\Rightarrow} C_2$: TMが$C_1$から$C_2$に何ステップかで移れる。
つまり$k \geq 0$ステップで移れるとき。

1.チューリング機械 (9)

定義:

入力列xに対して、ある受理状態$s_f \in F$とある列$y, z \in \Gamma^*$があって、

$$(s_0, \epsilon, x) \overset{*}{\Rightarrow} (s_f, y, z)$$

のとき、xは受理されるという。

yとzはどのような列でもよい。
つまり、初期様相からスタートしてTMのいずれかの受理状態に到達できれば受理である。

1.チューリング機械 (10)

例題 5.1 $L = \{1^{2^n} \mid n \geq 0\}$を認識するTMを構成せよ。

(解答)

Lに入る列は、

$1^2 = 11$

$1^4 = 1111$

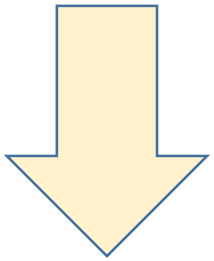
$1^8 = 11111111$

$1^{16} = 1111111111111111$

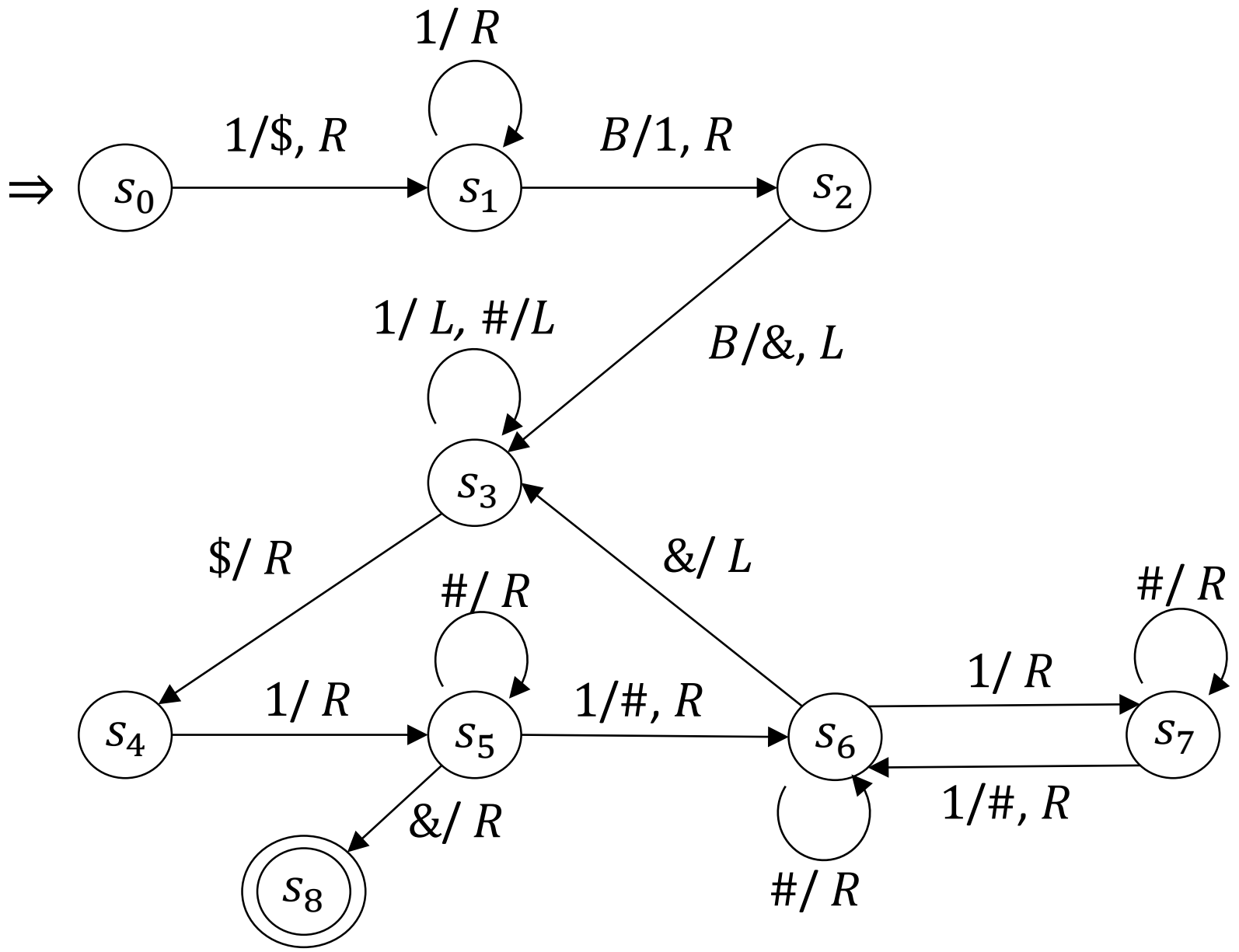
・・・などである。

Lがcfgで生成できないことは、以前言及した。

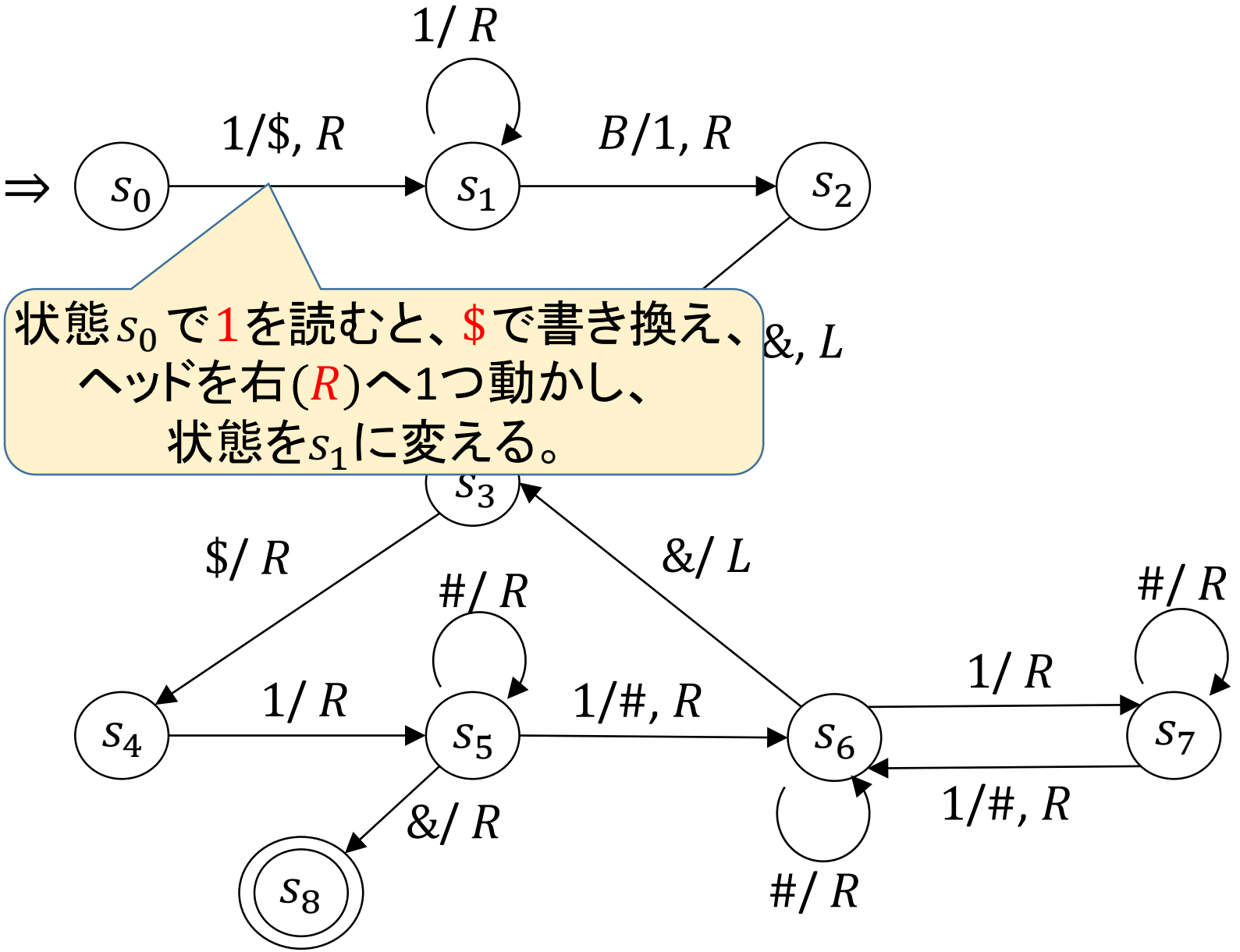
Lを認識するTMの状態遷移図は以下のように書ける



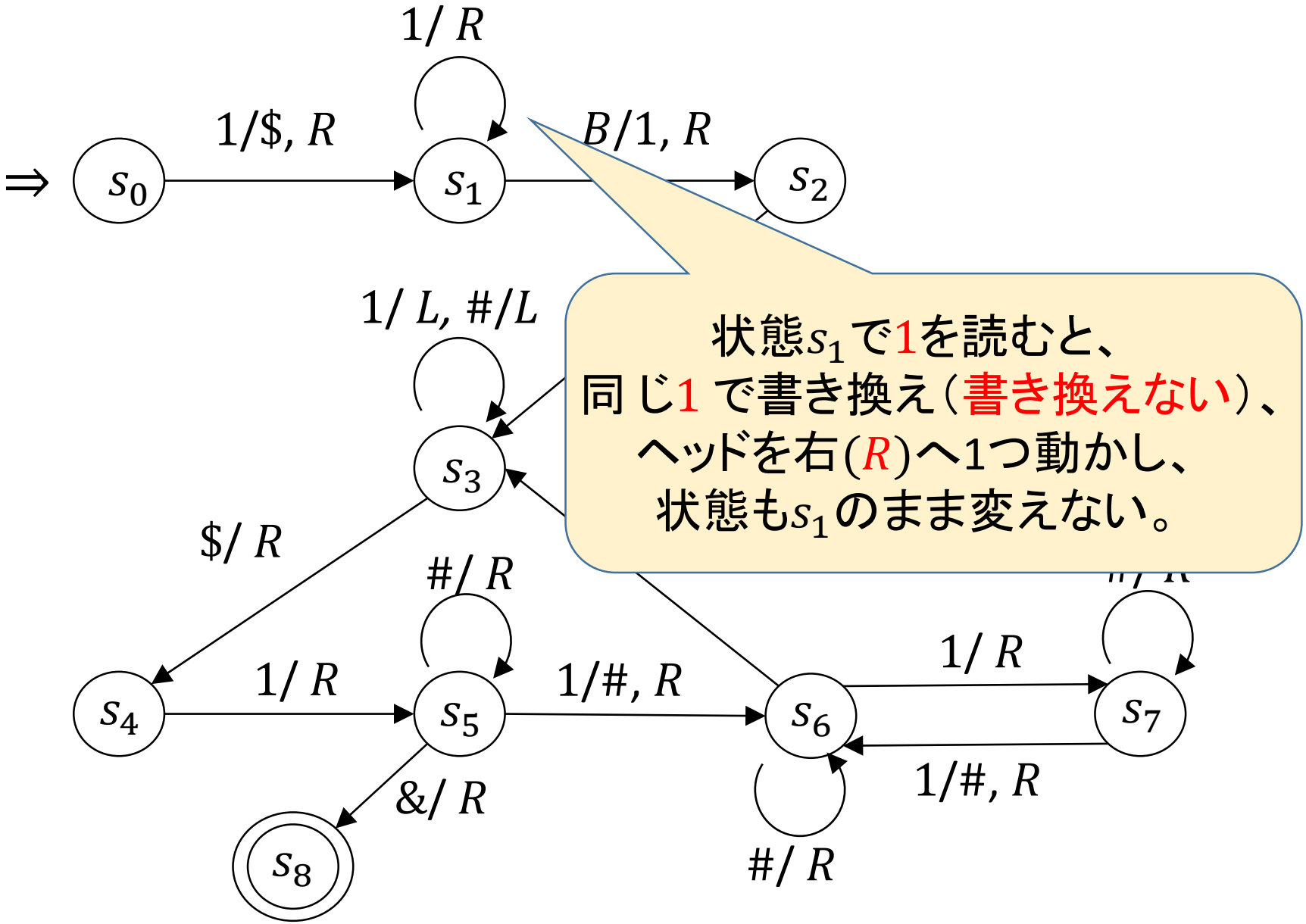
状態遷移図



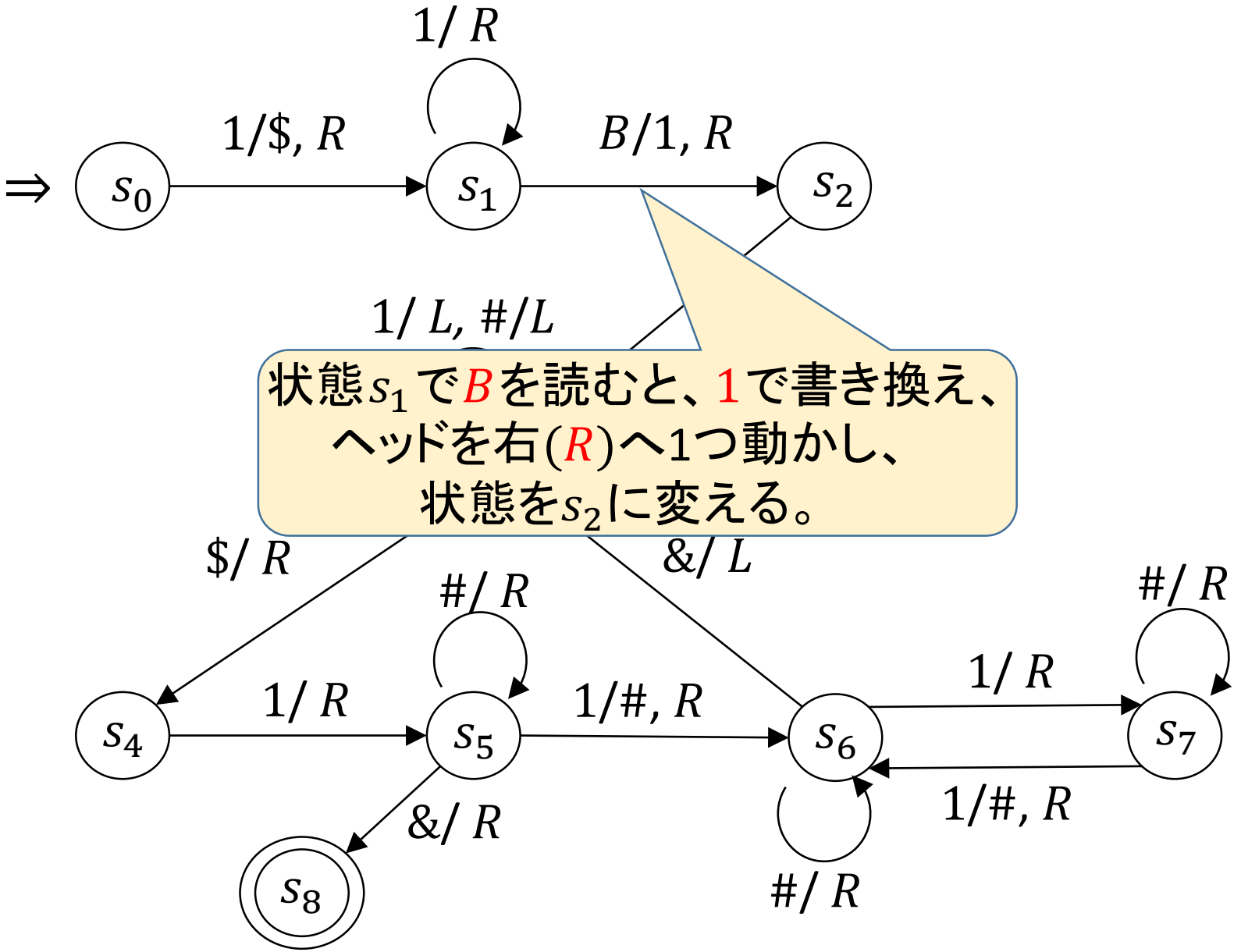
1.チューリング機械 (12)



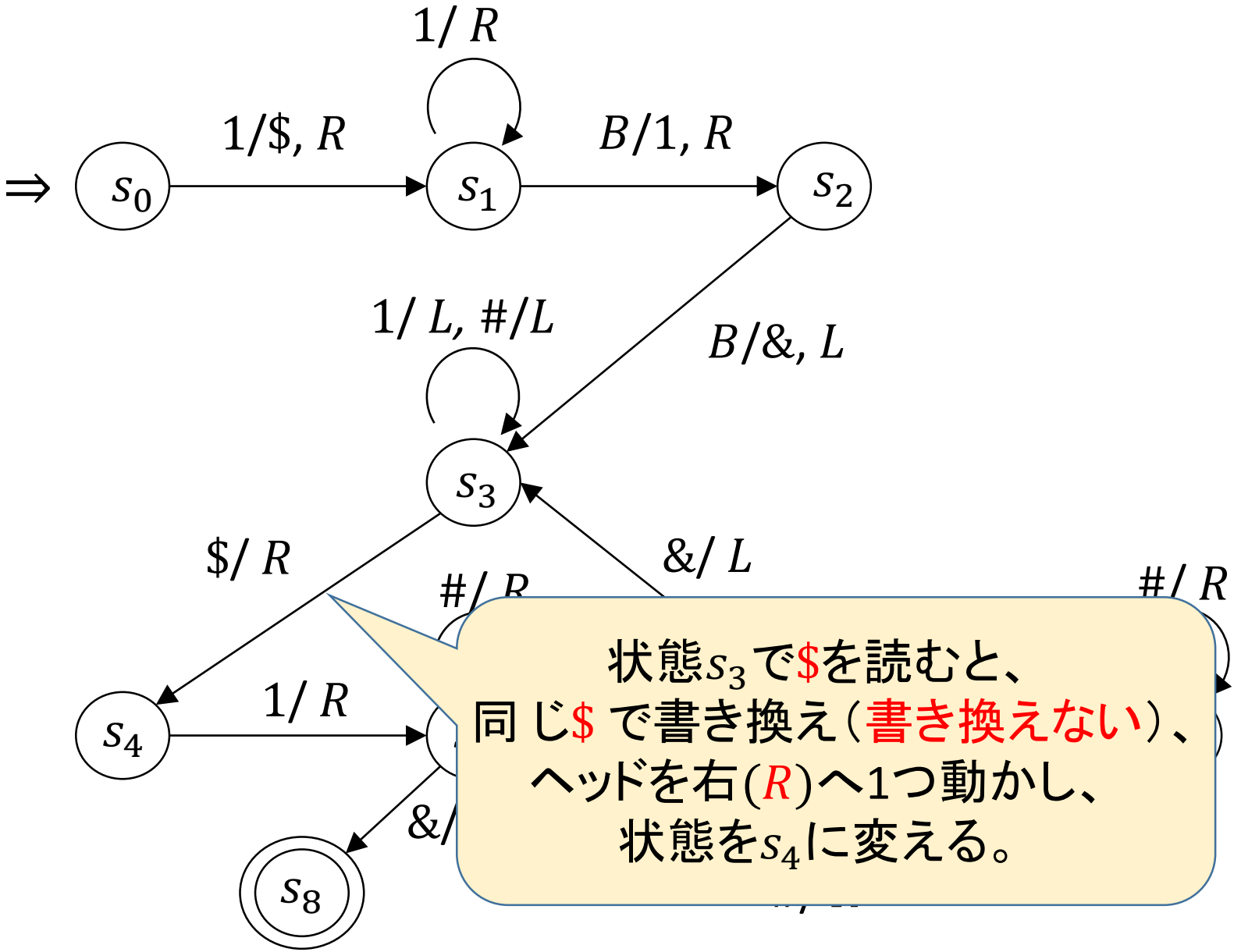
1.チューリング機械 (13)



1.チューリング機械 (14)

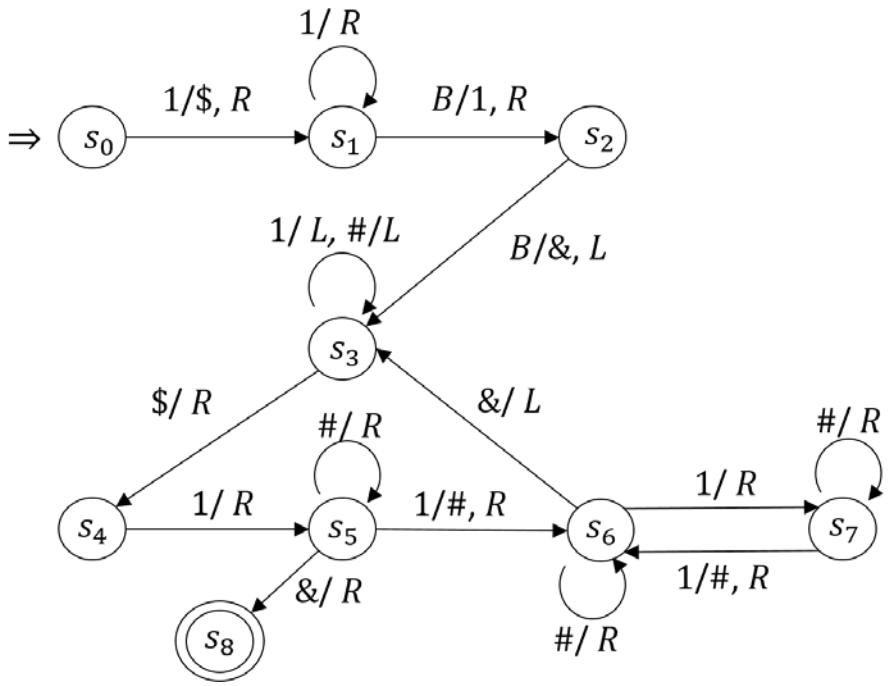


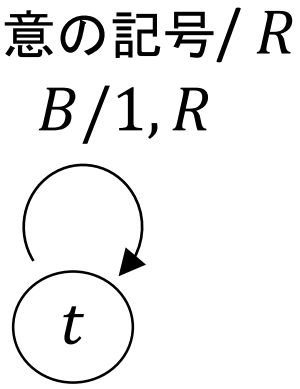
1.チューリング機械 (14)



1.チューリング機械 (15)

まとめると、矢のラベル“$a/b, c$”は、
aを読むと、bで書き換え、cの方向にヘッドを動かすの意味。





状態遷移図で未定義となっている矢は全て、
そこにいくと脱出でず受理されないゴミ箱状態
に遷移していると考える。（次の様相がないと考えてもよい。）

1.チューリング機械(16)

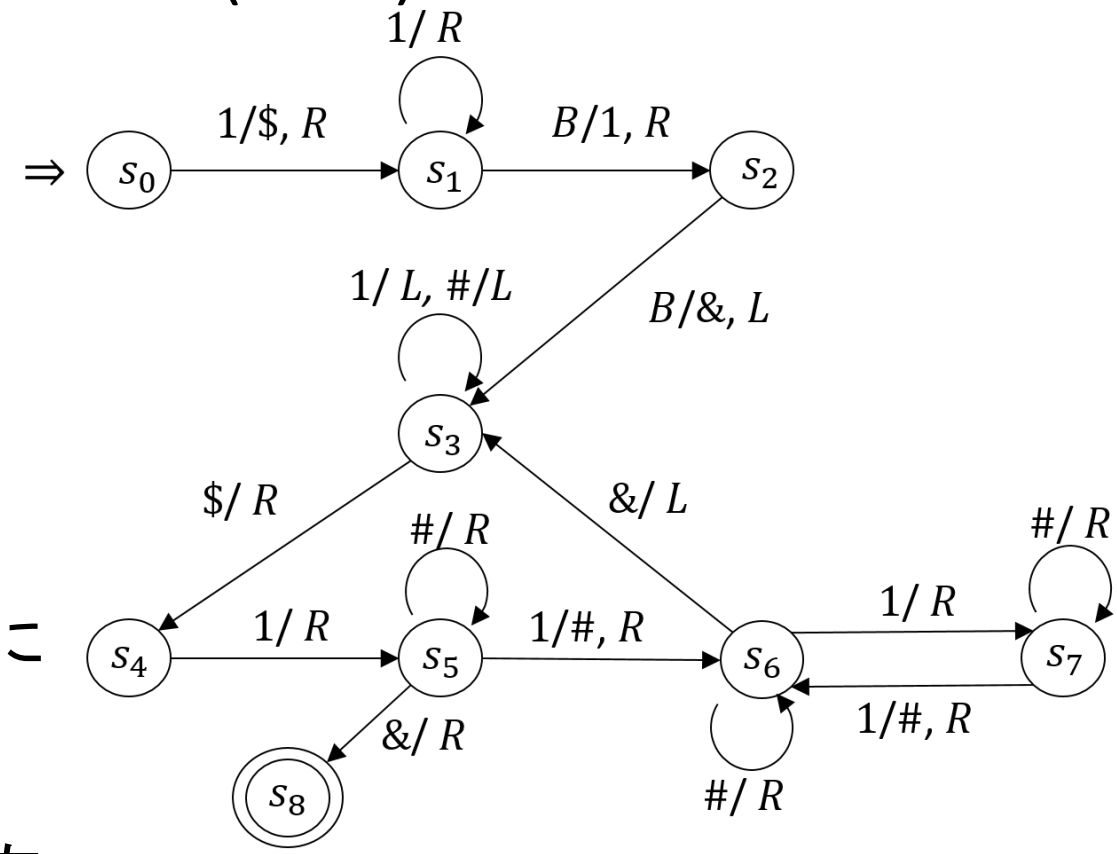
入力1^8への動作
1 1 1 1 1 1 1 1 B B B

状態s_0で1を$に
$ 1 1 1 1 1 1 1 1 B B B

s_1では1は読み飛ばしBを1に
$ 1 1 1 1 1 1 1 1 1 B B

s_2ではBを&にしてヘッドは左
$ 1 1 1 1 1 1 1 1 & B :要するに内容を変えずに両端を$と&で囲った

s_3とs_4を経てs_5へ、テープは変わらずヘッドは左から3つ目に行く、
$ 1 1 1 1 1 1 1 1 & B



1.チューリング機械 (17)

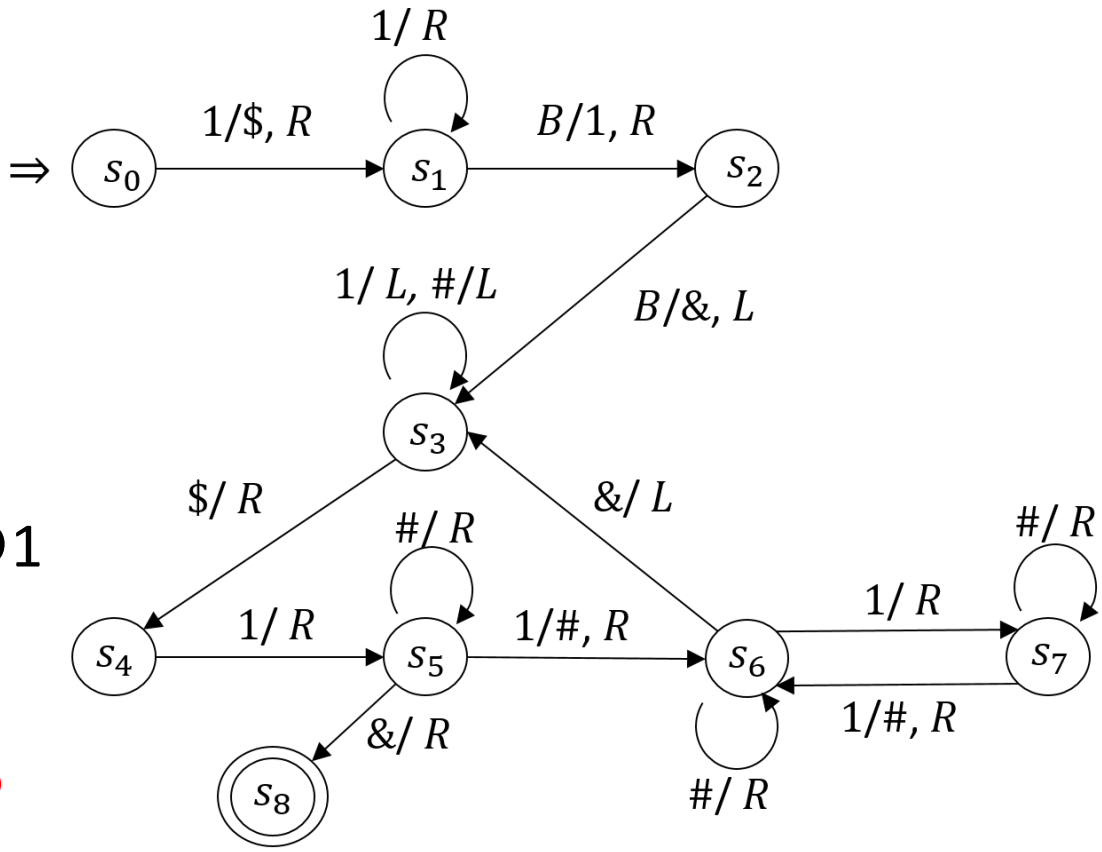
$ 1 1 1 1 1 1 1 & B　　でヘッドは左から3マス目、状態はs_5

s_5で1を#に変換してs_6へ
$ 1 # 1 1 1 1 1 1 & B

s_6とs_7を往復すると
$ 1 # 1 # 1 # 1 # & B

s_6で&を読んで、
$s_3 \rightarrow s_4$となりヘッドは最初の1

もし1の数が奇数ならここで、
s_7で&を読んで不受理になる



1.チューリング機械 (18)

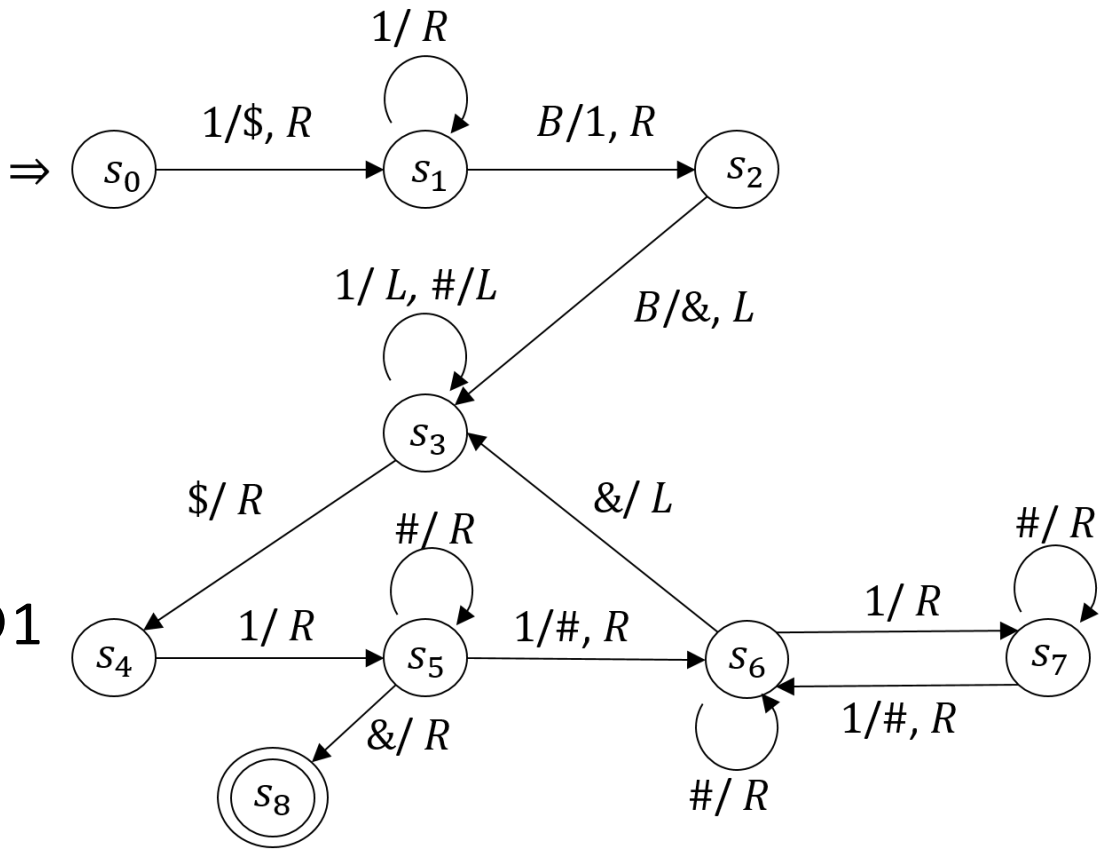
$ 1 # 1 # 1 # 1 # & B　　でヘッドは左から4マス目、状態はs_5

s_5で1を#に変換してs_6へ
$ 1 # # # 1 # 1 # & B

s_6とs_7を往復すると
1を一つおきに#に変える
$ 1 # # # 1 # # # & B

s_6で&を読んで、
$s_3 \rightarrow s_4$となりヘッドは最初の1

もし1の数が奇数ならここで、
s_7で&を読んで不受理になる



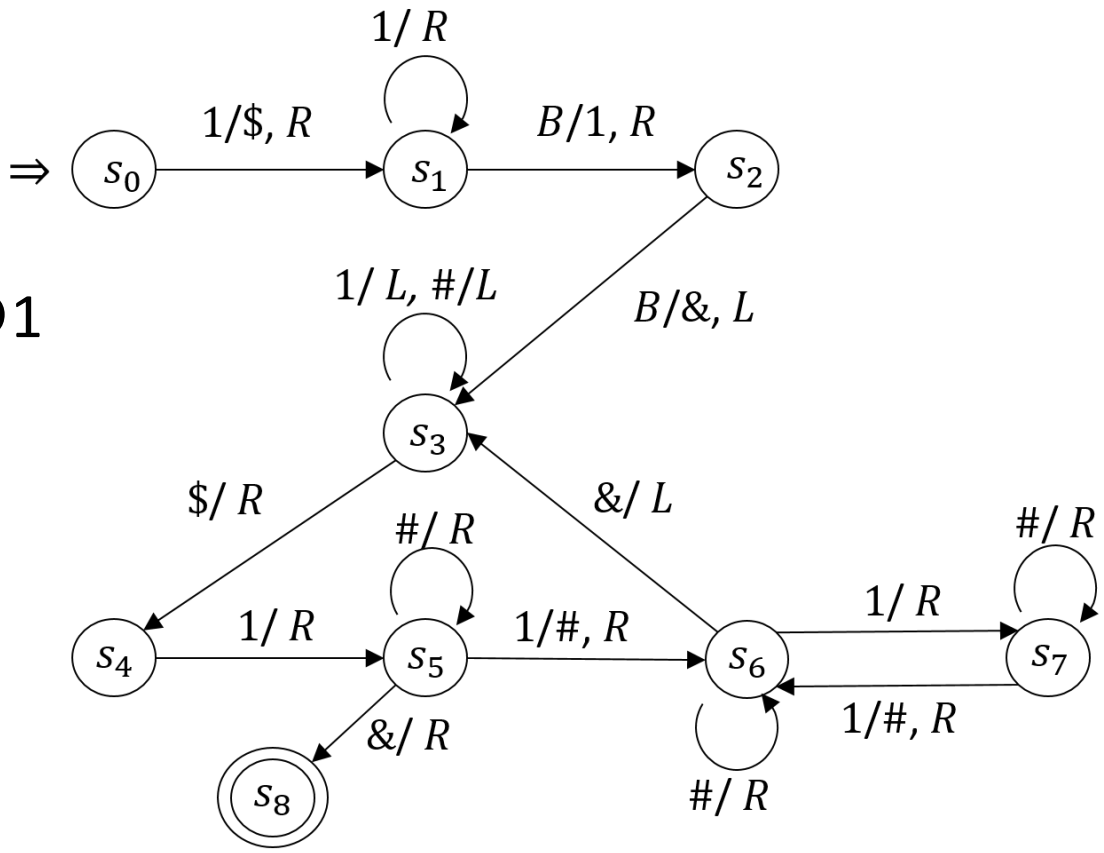
1.チューリング機械 (19)

$ 1 # # # 1 # # # & B　　でヘッドは左から6マス目、状態はs_5

s_5で1を#に変換してs_6へ
$ 1 # # # # # # # & B

s_6で&を読んで、
$s_3 \rightarrow s_4$となりヘッドは最初の1

今回はs_5で&を読み、
s_8へ行き受理される。

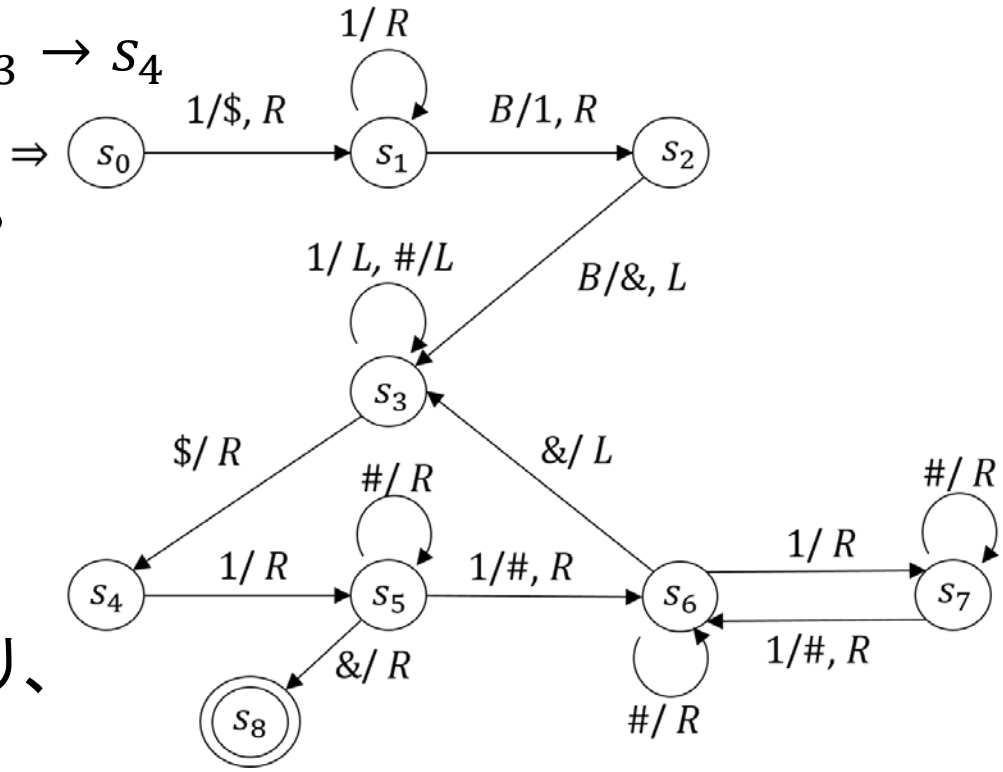


1.チューリング機械 (20)

まとめると、最初1^kが入力とすると、
最初s_4までで、$\$\ 1^k\ \&\ B$ となり、
$s_4 \to s_5 \to (s_6 \to s_7)^n \to s_6 \to s_3 \to s_4$
と一周するたびに、
1の次の1が#に置き換えられる。
よって1は$k/2$ 個になる。

l周すると1は$k/2^l$ 個になる。

$k = 2^n$のときのみ、
最後に$\$1\ \#\ \#\ \#\ \ldots\ \#\ \#\ \#\ \&\ B$となり、
s_8に行き受理される。

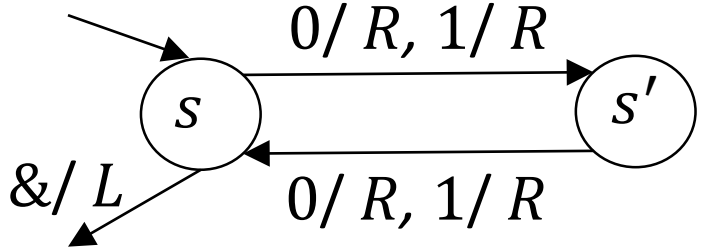


1.チューリング機械 (21)

例題 5.2 $L = \{xx \mid x \in \{0,1\}^*\}$ を認識するTMを構成せよ。

(解答)
状態遷移図は書き下さない。
pdaのときのように十分詳しい動作を述べる。

① 受理されるべき列の長さは偶数なので、
列全体をスキャンし、長さが偶数であることを確認。
長さが奇数なら停止して不受理。

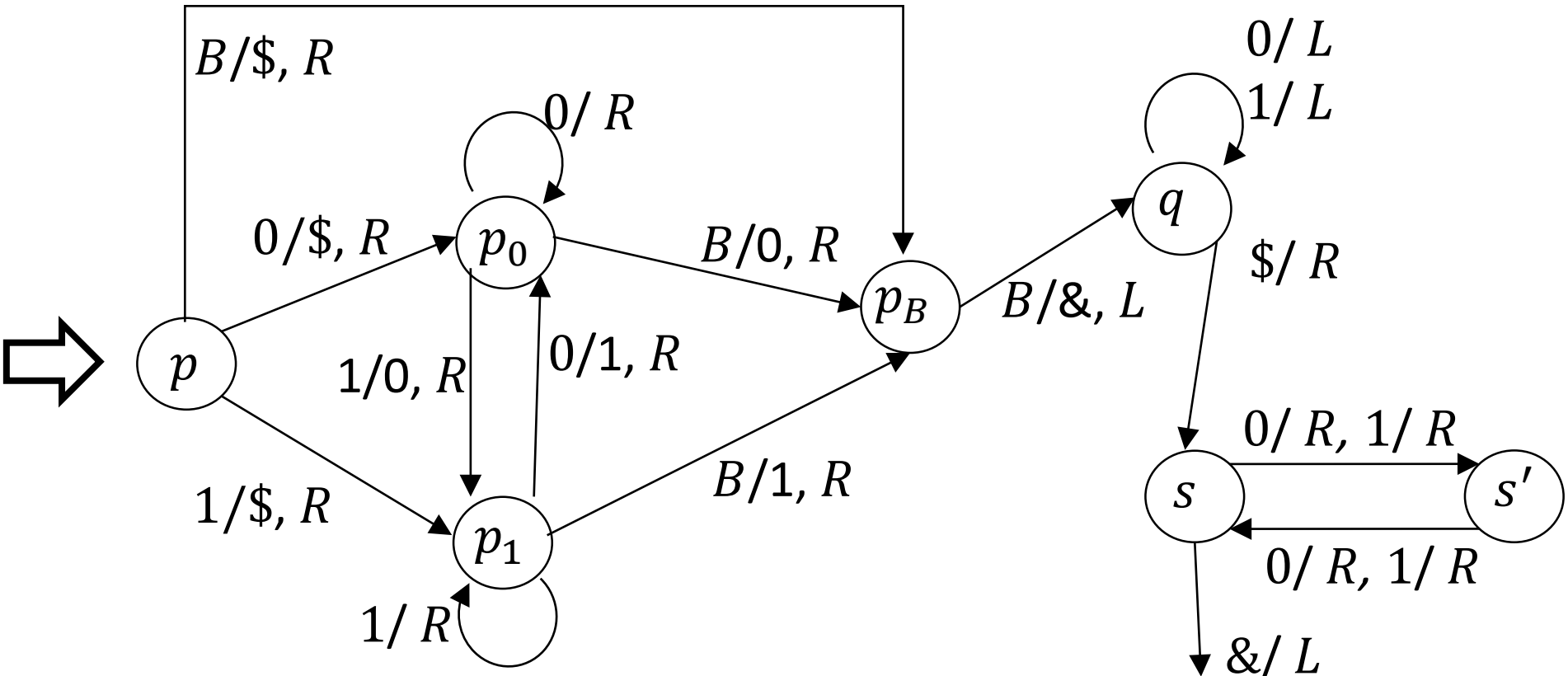


前回の例と同様に、列の両端が分かるようにする。
(列を$と&で挟んでおく。)

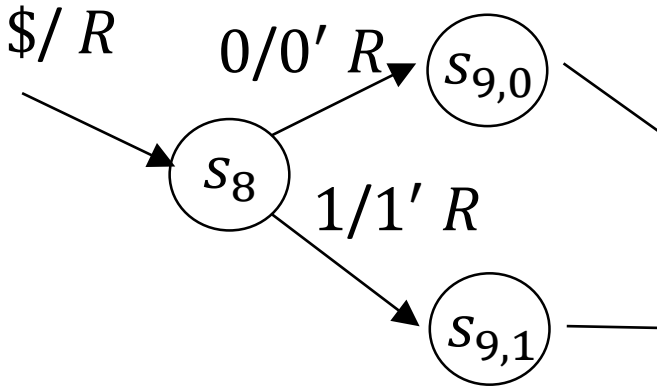
1.チューリング機械 (21')

前回の例と同様に、列の両端が分かるようにする。
(列を$と&で挟んでおく。)

入力列を$と&で挟む方法は以下を参照:



1.チューリング機械 (22)



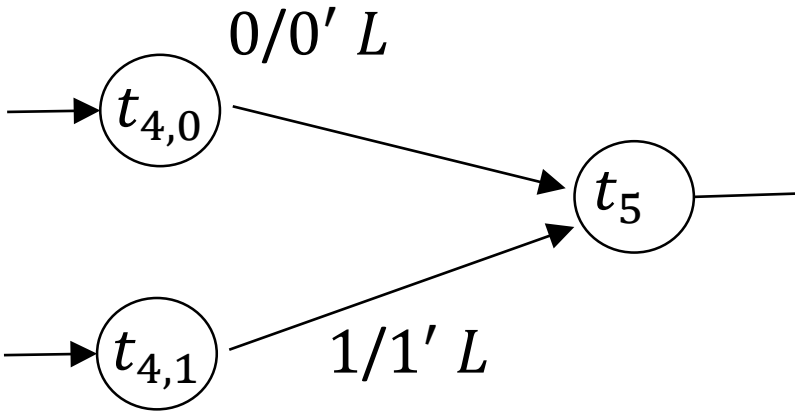
② 長さが偶数であるなら、入力列は
$a_1 a_2 \cdots a_n a_{n+1}\, a_{n+2} \cdots a_{2n}$ と書ける。

(基本方針)
ヘッドを左端に戻して、a_1の値(0 or 1)を有限状態で記憶し、
a_1を$a_1{}'$で書き換える。(0 → 0′, 1 → 1′とする):

$$a_1' a_2 \cdots a_n a_{n+1}\, a_{n+2} \cdots a_{2n}$$

次に、ヘッドを右に動かしてa_{n+1}を探す。 （詳細は後述）
記憶しておいたa_1と比較。
異なれば非受理、
同じなら$a_{n+1}{}'$に書き換える。

$$a_1' a_2 \cdots a_n a_{n+1}'\, a_{n+2} \cdots a_{2n}$$

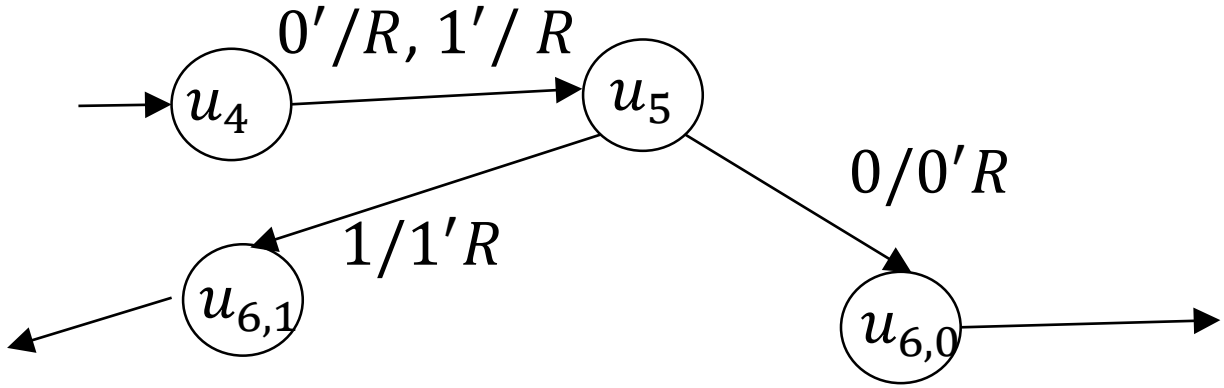


1.チューリング機械 (23)

$a_1' a_2 \cdots a_n a'_{n+1}\, a_{n+2} \cdots a_{2n}$

ヘッドを左に動かしていく。
プライムのついた記号(今は$a_1{}'$)を見つけたら、
その右隣にヘッドを動かしてa_2を記憶し、$a_2 \to a_2'$とする。

$a_1' a_2{}' \cdots a_n a'_{n+1}\, a_{n+2} \cdots a_{2n}$

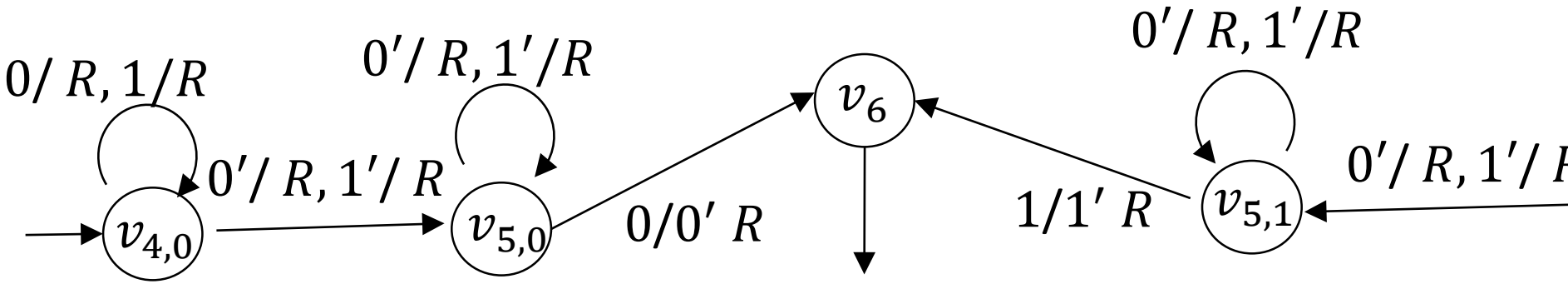


そこからヘッドを右に動かして・・・

1.チューリング機械 (23′)

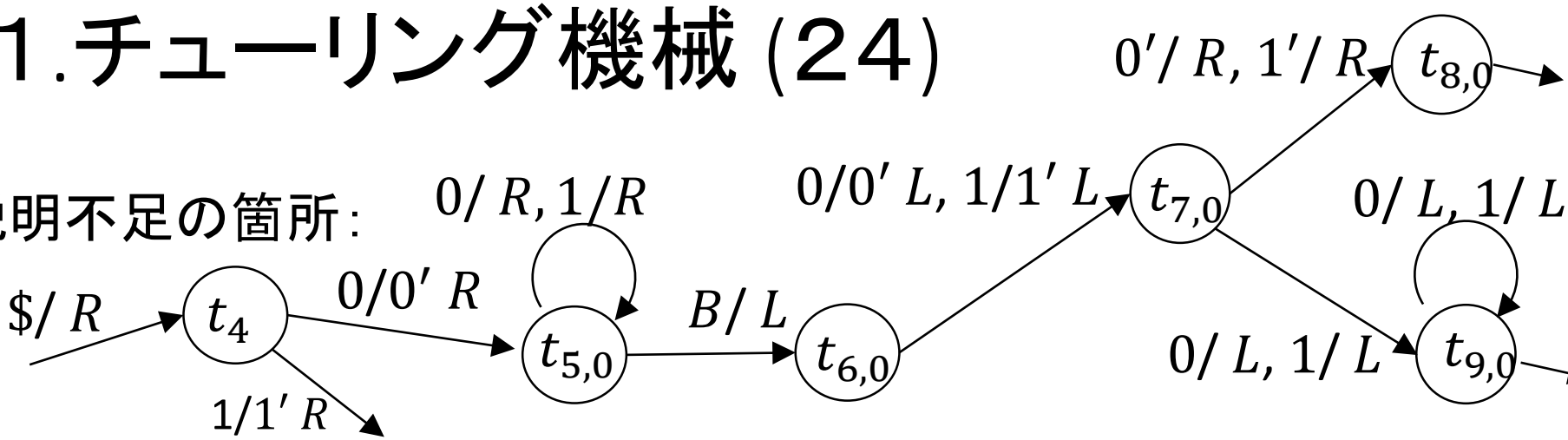
そこからヘッドを右に動かして、いったんプライムのついた記号に出会ってから、最初に出会うプライム無しの記号がa_{n+2}なので、a_2とa_{n+2}を比較。
同じなら、$a_{n+2} \to a'_{n+2}$とする。

$$a'_1 a'_2 \cdots a_n a'_{n+1}\, a'_{n+2} \cdots a_{2n}$$



この作業を続けていき、a_nとa_{2n}が等しいことまで分かれば、列がxxの形をしていることがチェックできたことになるので、受理する。

1.チューリング機械 (24)



②で最初に『a_{n+1}を探す』やり方が、上記の説明ではよくわからない。

(a_{n+1}の探し方)

最初にa_1にプライムを付ける。

$$a_1{}'a_2 \cdots a_n a_{n+1}\, a_{n+2} \cdots a_{2n} BB$$

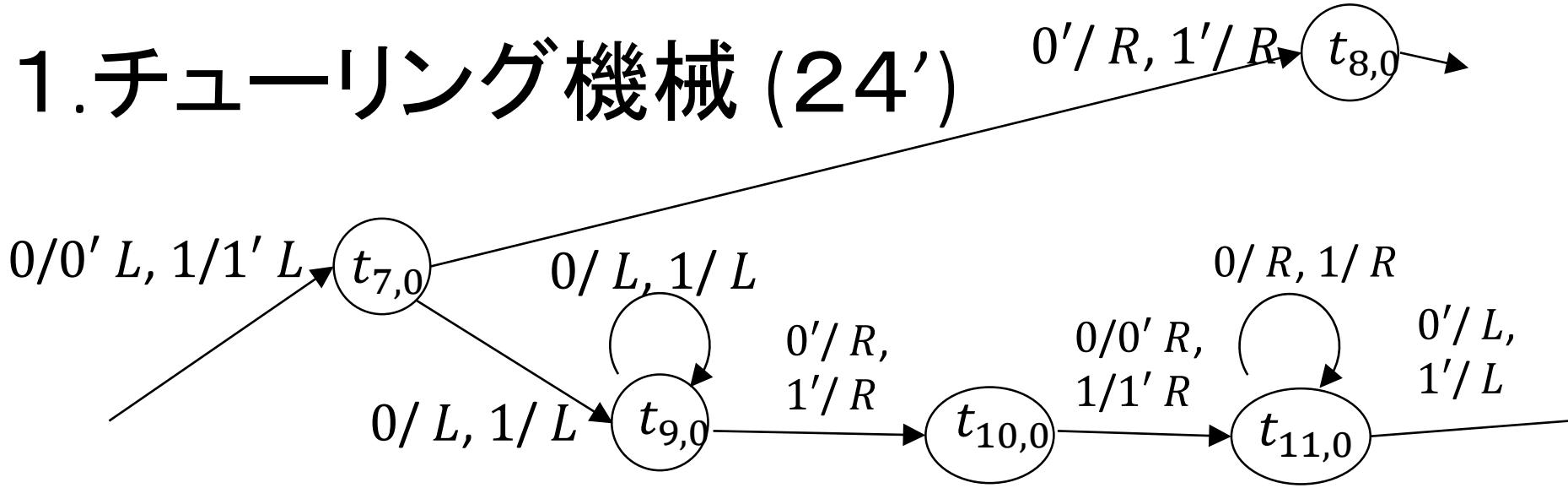
次にヘッドを右にスライドさせBに出会うと一つ左に戻してa_{2n}を見つけa_{2n}にプライムを付ける。

$$a_1{}'a_2 \cdots a_n a_{n+1}\, a_{n+2} \cdots a_{2n}{}' BB$$

次にヘッドを左にスライドさせて最初に出会うプライム付きの記号(a_1)の一つ右隣りのa_2を見つけて、a_2にプライムを付ける。

$$a_1{}'a_2{}' \cdots a_n a_{n+1}\, a_{n+2} \cdots a_{2n}{}' BB$$

1.チューリング機械 (24’)



次にヘッドを左にスライドさせて最初に出会うプライム付きの記号(a_1)の一つ右隣りのa_2を見つけて、a_2にプライムを付ける。
次にヘッドを右にスライドさえて最初に出会うプライム付きの記号(a_{2n})の一つ左隣りのa_{2n-1}を見つけて、a_{2n-1}にプライムを付ける。

$$a_1{}'a_2{}' \cdots a_n a_{n+1}\, a_{n+2} \cdots a_{2n-1}{}' a_{2n}{}' BB$$

・・・・・これを続けていったときに、a_{2n-k}にプライムを付けた後、
ヘッドを一つ左にするとプライム付きが現れたら、
このa_{2n-k}は実は、a_{n+1}であることが分かる。
後は、a_1とa_{n+1}以外のプライムを全て消せばOK！

1.チューリング機械 (25)

例題 5.3 $\{0^n 1^{n^2} \mid n \geq 1\}$を認識するTMを構成せよ。

解答:（左端を分かるようにしておく）
構成は大まかに以下になる:

①n個の0を1の右の空白部分にコピーする。
　この作業をn買い続ければ、1の右に0^{n^2}ができる。

$$0^n 1^n 1^n \cdots 1^n 0^n 0^n \cdots 0^n BBB$$

②1の数と、1の右に並ぶ0の数が同じかどうかをチェックする。

1.チューリング機械 (25′)

$$0^n 1^n 1^n \cdots 1^n 0^n 0^n \cdots 0^n BBB$$

②1の数と、1の右に並ぶ0の数が同じかどうかをチェックする。

②の詳細:

$$0 \cdots 01'11 \cdots 10'00 \cdots 0 \Longrightarrow 0 \cdots 01'1'1 \cdots 10'0'0 \cdots 0$$

左端から1を1′に変えて、右に行き最初の0を0′に変える。

『左に戻って最初に出会った1′の右の1を1′に変えて、右に向かって1と0′を通り過ぎて、最初の0を0′に変える。』

これを繰り返し、1が全て1′になったとき、0が全て0′になればOK！

1.チューリング機械 (26)

①の詳細:　　$0^n1^n1^n\cdots1^n0^n0^n\cdots0^nBBB$

最初の0をチェックする($0 \to 0'$)。右に行き1を通り越して空白を見つけると0を書き込む。

$$0'0\cdots01\cdots10BBB$$

a) 左に行き$0'$を見つけるとその右の0を$0'$にする。
b) 右に行き1を通り越し0も通り越し、空白を見つけると0を書き込む。

$$0'0'0\cdots01\cdots100BBB$$

$$0'0'\cdots0'1\cdots100\cdots0BBB$$

(n 個の $0'$ と、n 個の 0)

a)b)を繰り替えし、1の直ぐ左に$0'$を見つけると、1の左にあるすべての0がチェックされたので、これらの$0'$を0に戻して、最初の$0'$を$\hat{0}$とする。

1.チューリング機械 (27)

$$\overbrace{\hat{0}0\cdots01}^{n}\cdots\overbrace{100\cdots0}^{n}BBB$$

$\hat{0}$の右隣が1でないかを確認する。
もし、1なら既に、$\hat{0}^n1\cdots10^{n^2}BBB$　　となっている。

そうでない場合は、$\hat{0}$を$\hat{0}'$にする。
右に行き1を通り越し0も通り越し、空白を見つけると0を書き込む。

$$\hat{0}'0\cdots01\cdots10^n0BBB$$

1.チューリング機械 (27′)

$$\hat{0}'0\cdots01\cdots10^n0BBB$$

a′) 左に行き$0'$もしくは$\hat{0}'$を見つけると、
その右が、0なら$0'$に、$\hat{0}$なら$\hat{0}'$にして b)へ、もし1ならc)へ進む。
b) 右に行き1を通り越し0も通り越し、空白を見つけると0を書き込む。
a′)に戻る。

$$\hat{0}'\overbrace{0'\cdots0'}^{n}1\cdots10^n\overbrace{0\cdots0}^{n}BBB$$

c) 1の直ぐ左に$0'$を見つけると、1の左にあるすべての0もしくは$\hat{0}$がチェックされたので、$0'$を0に戻しながら左へ進んでいき、最初に出会った$\hat{0}'$の右の 0を$\hat{0}$とする。その後$\hat{0}'$を$\hat{0}$に戻す。

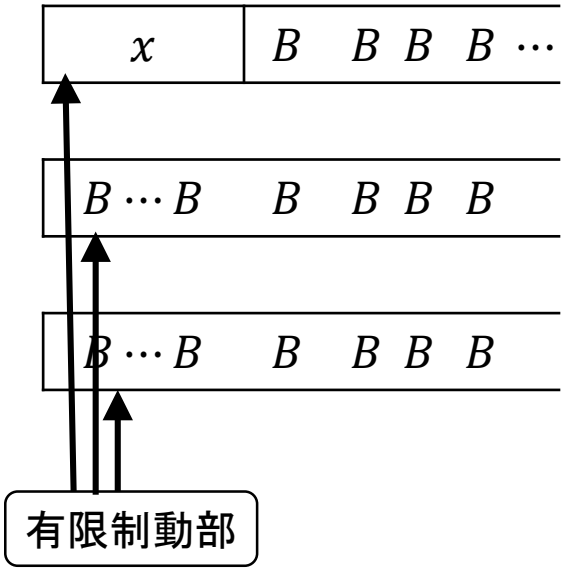
$$\hat{0}\hat{0}0\cdots01\cdots10^n0^nBBB$$

以上を繰り返すと、1の左の0が全て$\hat{0}$になったときには、

$$\hat{0}^n1\cdots10^{n^2}BBB$$ となっている。

1.チューリング機械 (28)

多テープチューリング機械：テープを2本以上有するTM.



3本の場合の状態遷移関数：

$\delta(s, a_1, a_2, a_3) = (s', b_1, b_2, b_3, d_1, d_2, d_3)$

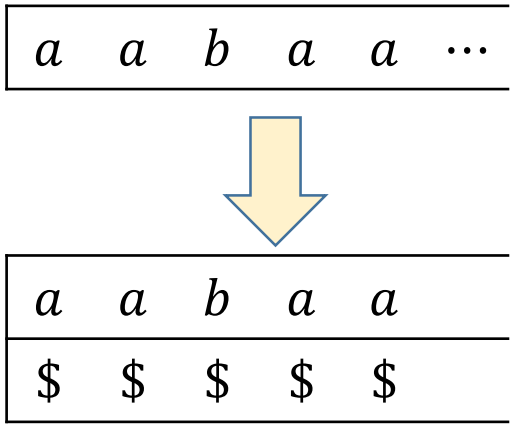
3本のテープの入力(a_1, a_2, a_3)を読んで、
それらを書き換え(b_1, b_2, b_3)
各ヘッドを独立に移動させる(d_1, d_2, d_3).

d_i $(1 \leq i \leq 3)$ は、L (左), R (右), S (ヘッドを動かさない)のいずれか。
(注)Sを入れないと、例えば1本目と2本目のテープのヘッドの位置の差は常に偶数になってしまう。

2本目以降のテープは、最初は全て空白Bが書かれている。
全てのヘッドはテープの左端からスタートする。

1.チューリング機械 (29)

定理 5.1 ある言語Lが多テープTMで認識されるならLは標準のTMによっても認識される。
証明の概要（2テープの場合）



多テープTMを模倣するTM Mは、
まずテープを”2重トラック”に変更する。

例えば上のトラックにa、下のトラックにbならば、
(a, b)を新たな一つの記号として定義する。
上のトラックのみ$a \to c$としたければ、$(a, b) \to (c, b)$と書き変える。

最初、下のトラックには$を($B$の代わりに)書き入れておく。
入力列が$aabaa$なら、最初に$(a, \$)(a, \$)(b, \$)(a, \$)(a, \$)$とする。
それより右の最初は空白のところは、使う直前に二重トラックにする。

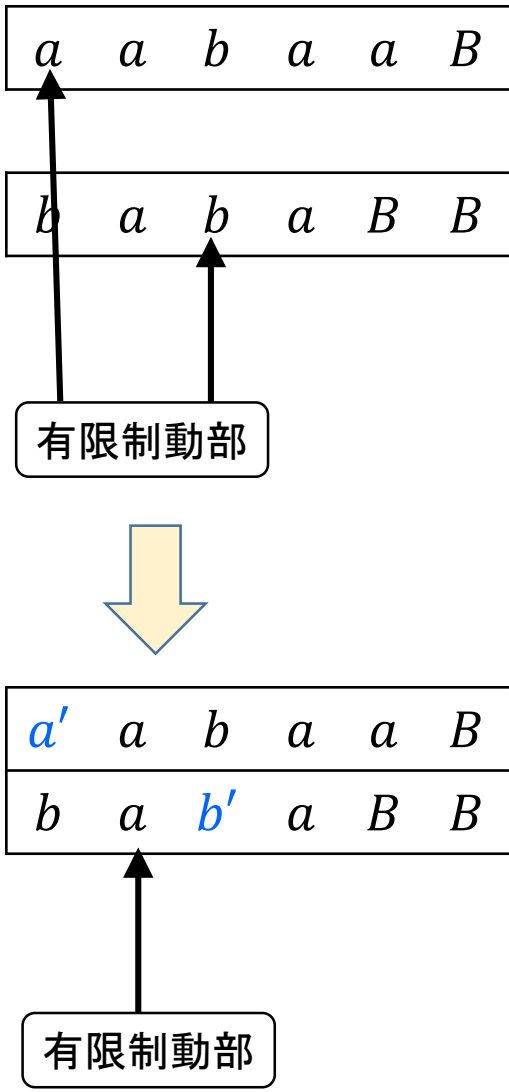
1.チューリング機械 (30)

2重トラックに改造した後の模倣の仕方：

模倣元の2テープTMのヘッドの位置にある記号にプライムを付ける。

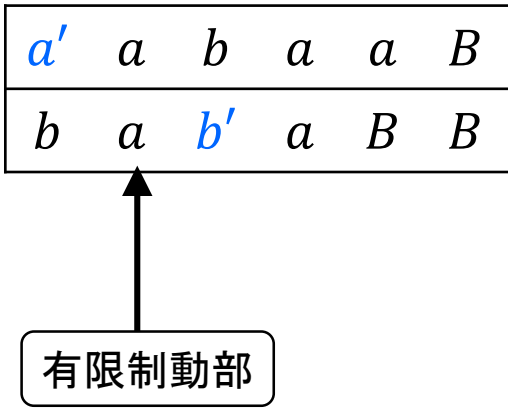
例えば、(a', b)は一つ目のテープのヘッドがこの位置にあるということを指す。

(a', b')は一つ目のテープのヘッドと、二つ目のテープのヘッドが、両方この位置にあるということを示す。



1.チューリング機械 (30’)

1ステップの模倣は以下の2段階からなる:



①テープを左端からスキャンして、
上のトラックにプライムがついている記号を探す。
1本目のテープの行動を模倣し、
ヘッドの位置を書き変える。

②同様のことを下のトラックに行い、2本目のテープを模倣する。
テープを左端からスキャンして、
下のトラックにプライムがついている記号を探す。
2本目のテープの行動を模倣し、
ヘッドの位置を書き変える。

以上の方法で、2テープTMを模倣できる。

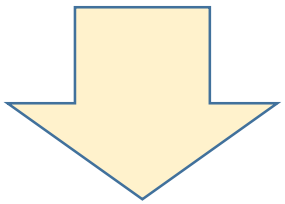
1.チューリング機械 (31)

非決定性チューリング機械:
決定性TMとの違い、状態遷移関数が動作の集合への写像となる:

$$\delta(s, a) = \{(s_1, a_1, L), (s_2, a_2, R)\}$$

非決定的選択:どの動作を実行するかの選択
受理条件:
(上手にguessして非決定的選択を行い)受理状態に到達するなら受理、どのような非決定的選択をしても受理状態に到達できないなら非受理

決定性TMで認識できる言語は、非決定性TMでも認識できるのは当然だが、実は逆も正しい



1.チューリング機械 (32)

定理 5.2 非決定性TMで認識できる言語は決定性TMでも認識できる。
証明の概要:
非決定的選択は、いかなる状態とテープ記号の組合わせに対しても2個以下だと仮定する。
つまり、

$$\delta(s,a) = \{(s_0,a_0,d_0),(s_1,a_1,d_1)\} \qquad -(*)$$

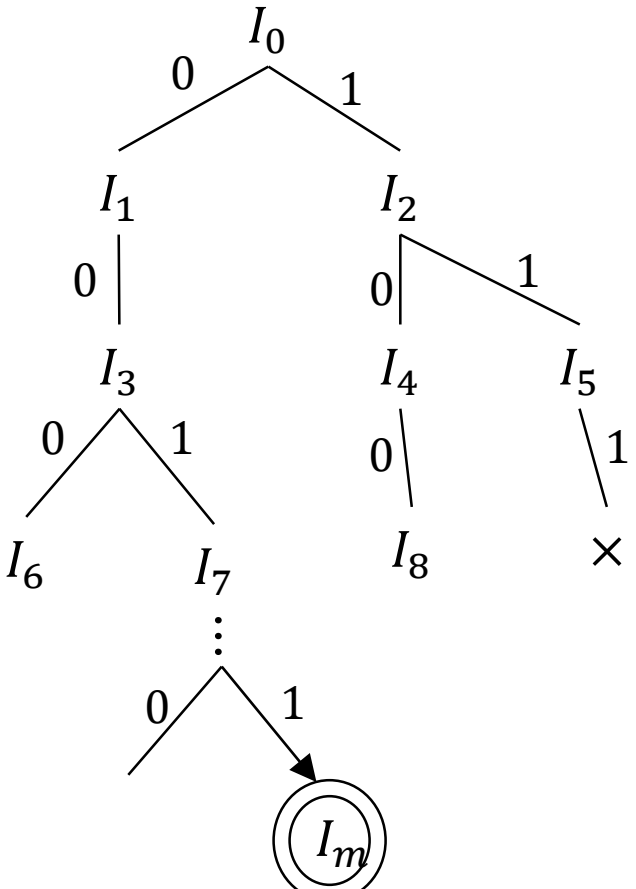
の右辺の集合の大きさが2以下とする。(d_0,d_1はLかR)
(一般には任意の定数K個以下になるが、拡張は容易にできる。)

右の$\{(s_1,a_1,L),(s_2,a_2,R)\}$は2つの非決定性選択を示すが、
任意の$s \in K, a \in \Gamma$に対して、
集合$\delta(s,a)$の元を0と1で番号付けする。
つまり(s_0,a_0,d_0)と(s_1,a_1,d_1)のように0,1で選択を指定できるようにする。

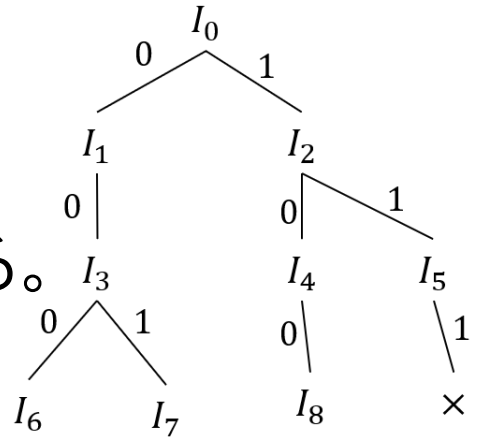
1.チューリング機械 (33)

与えられた非決定性TM Tを決定性TM Mで模倣する。
Tは各ステップで、最大2通りの非決定性選択が可能なので、
一つの様相から1ステップで遷移できる様相が最大で2つ存在。
よって、初期様相I_0を根とする木で書ける。

各ステップの非決定的選択は
$0\ or\ 1$で指定できるので、
全ての様相はバイナリーの列で指定できる。
右の例では、$I_3 = I(00), I_8 = I(100),$
また、$I(101)$は存在しない。



1.チューリング機械 (34)



Mは2テープTMとする。1本目はTのテープを模倣する。
2本目は、バイナリーの列の最後に#を加えたもの、
例えば$0001101\#$が書いてある。
この場合、Mは$I(0), I(00), I(000)$という順で$I(0001101)$まで模倣する。
具体的には

$a\ a\ b\ a\ a\ B$

$0\ 0\ 0\ 1\ 1\ 0$

・2つ目のテープのヘッドは最初左端におく。（ここでは0）
・Tの最初のステップの模倣としては、“0”の選択をする。
・2つ目のテープのヘッドを一つ右に動かす。（ヘッドは0に行く）
・Tの第2ステップの模倣としては、”0”の選択をする。
・・・・・
・Tの第7ステップの模倣としては、”1”の選択をする。
・2つ目のテープのヘッドを一つ右に動かすと#を読むので、
模倣をひとまず終了させる。

1.チューリング機械 (35)

2つ目のテープに$a_1 a_2 \cdots a_k$#が書かれている時は、　$(a_i \in \{0,1\})$
『Tの第lステップの模倣として"a_l"の選択をし、ヘッドを右へ』
を$l = 1$から、$l = k$まで続け、#を読むと模倣をひとまず終了させる。

以上のように、一つのフェーズでは、初期様相から始め2つ目のテープで指定される非決定的選択にしたがって模倣する。

2つ目のテープは、$0\#, 1\#, 00\#, 01\#, 10\#, 11\#, 000\#, \cdots$というように、$0\#$から初めて、長さが短い順に、同じ長さの列は二進数表示と思ったときに小さいもの順に並べる。

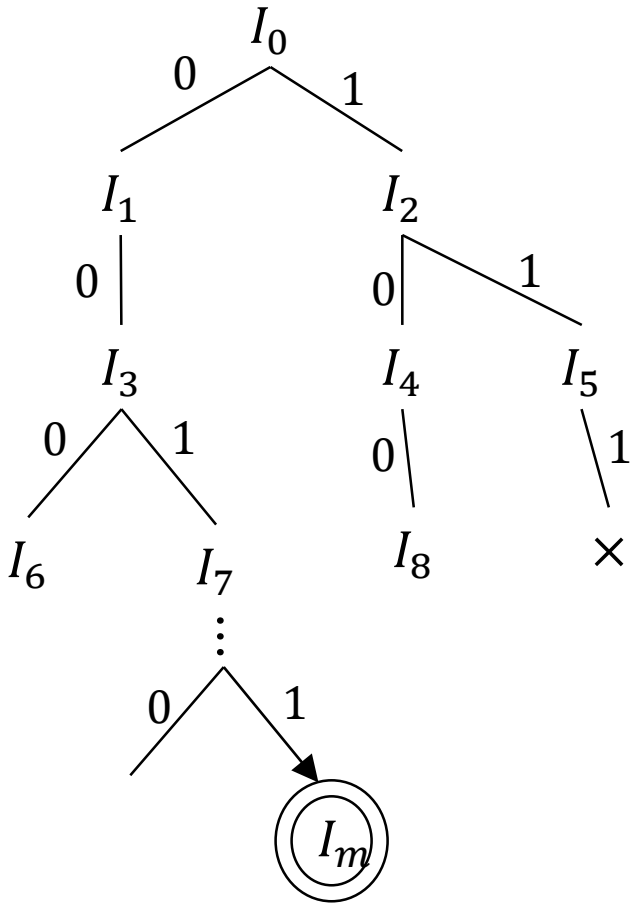
結局、Mは、2つ目のテープを$0\#$から初めて、上記の方法で模倣を続けていけば、もしTがある非決定的選択で受理状態に入るとき、またその時に限り2つ目のテープがある値のときに受理状態に入る。
つまり、この方法で模倣は成功する。

1.チューリング機械 (36)

結局、Mは、2つ目のテープを0#から初めて、上記の方法で模倣を続けていけば、もしTがある非決定的選択$(a_1 \cdots a_m)$で受理状態に入るならば、2つ目のテープが$a_1 \cdots a_m$#のときにMは受理状態に入る。

また、非決定的選択$(a_1 \cdots a_m)$で受理状態に入らないならば、2つ目のテープが$a_1 \cdots a_m$#　のときには受理状態に入らない。

つまり、この方法で模倣は成功する。



（注意）一般的にMはTよりも指数関数的に多くのステップを要する。

2. 0型文法 (1)

チューリング機械に対応する、最も能力の高い文法が0型文法。

0型文法$G = (V, \Sigma, P, S)$の生成規則 : $\alpha \to \beta$
αとβは任意の列

現在の列の部分列としてαが存在するなら、
それをβで置き換えてもよい。

定義　ある列$\alpha, \beta, \gamma_1, \gamma_2 \in (\Sigma \cup V)^*$に対して

$$\delta_1 = \gamma_1 \alpha \gamma_2, \delta_2 = \gamma_1 \beta \gamma_2$$

と書くことができ、かつ$\alpha \to \beta \in P$であるとき、
列δ_1から列δ_2が1ステップで導出される$(\delta_1 \Rightarrow \delta_2)$と呼ぶ。

何ステップかで導出される$(\delta_1 \overset{*}{\Rightarrow} \delta_2)$などの定義はcfgと同じ。

2. 0型文法 (2)

例題 5.1 $\{xx \mid x \in \{a,b\}^*\}$を生成する0型文法を求めよ。

解答: 正解は以下の生成規則

$$S \to XT, X \to AXa \mid BXb \mid \epsilon$$
$$Aa \to aA, Ab \to bA, Ba \to aB, Bb \to bB,$$
$$AT \to aT, BT \to bT, T \to \epsilon$$

$baabaa$を生成してみる:

$$S \Rightarrow XT \Rightarrow AXaT \Rightarrow AAXaaT \Rightarrow AABXbaaT \Rightarrow AABbaaT$$
$$\Rightarrow AAbBaaT \Rightarrow AAbaBaT \Rightarrow AAbaaBT \Rightarrow AAbaabT$$
$$\overset{*}{\Rightarrow} AbaabAT \Rightarrow AbaabaT \overset{*}{\Rightarrow} baabaAT \Rightarrow baabaaT \Rightarrow baabaa$$

一般的に、xxで$x \in \{a,b\}^*$生成したければ、a,bの代わりにA,Bを用いたときにxに対応する列を$\mathcal{X}$とするとき、

$S \Rightarrow XT \overset{*}{\Rightarrow} \mathcal{X}^R XxT \overset{*}{\Rightarrow} \mathcal{X}^R xT \overset{*}{\Rightarrow} xxT \Rightarrow xx$ と導出すればよい。

2. 0型文法 (3)

例題 5.1 $\{xx \mid x \in \{a, b\}^*\}$を生成する0型文法を求めよ。

解答のつづき:
この文法で生成されるならxxの形をしている証明は、

・X, Tとその他の記号(A, B, a, b)の順番は変更不能
・AとBの順番と、aとbの順番の変更が不能
・$T \to \epsilon$を使った後は、A, Bは終端記号に変換できないので、
$T \to \epsilon$は最後に使わざるを得ない。

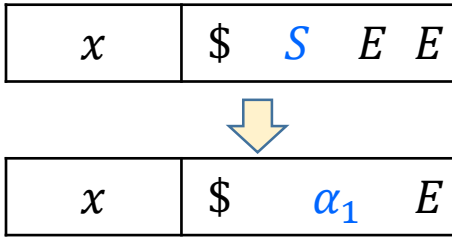
などの条件を使うことで可能になる。
複雑なので詳細は省略する。

2. 0型文法 (4)

定理 5.3 チューリング機械と0型文法は等価である。
つまり、言語 L がチューリング機械で認識される必要十分条件は、
Lが0型言語で生成可能であることである。

証明: まず与えられた文法Gを模倣する非決定性TM Mを作る。
この証明においては、Mの空白記号はEとする。

列xがGで生成可能ならば、$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow x$
という列が存在する。α_iを次々にMで作り出すことを考える。



- まずテープのxの右にSを書く。
- Sを左辺に持つ規則を非決定的に選択し、
  Sをその右辺で置き換える。
- 次に第2ステップで置き換える部分列βを非決定的にguessする。

というように進めていく。

2. 0型文法 (5)

具体例:

(1) まずテープのxの右にSを書く。

(2) Sを左辺に持つ規則の中から$S \rightarrow AAbbbaBB$を非決定的に選択し、 Sを$AAbbbaBB$ で書き変える。

(3) 次に第2ステップで置き換える部分列βを非決定的に選択する。

ここでは、$Abb \rightarrow BBbaa$が選択されたとする。

$AAbbbaBB$の"Abb"を$BBbaa$で書き変える。

(列の長さが増えるので、$AAbbbaBB \rightarrow AAbb\%\%baBB$
のように最初に右にシフトしてから書き変える。)

・・・・・

のように続けていく。

(1)

x	\$ S E E ⋯

(2)

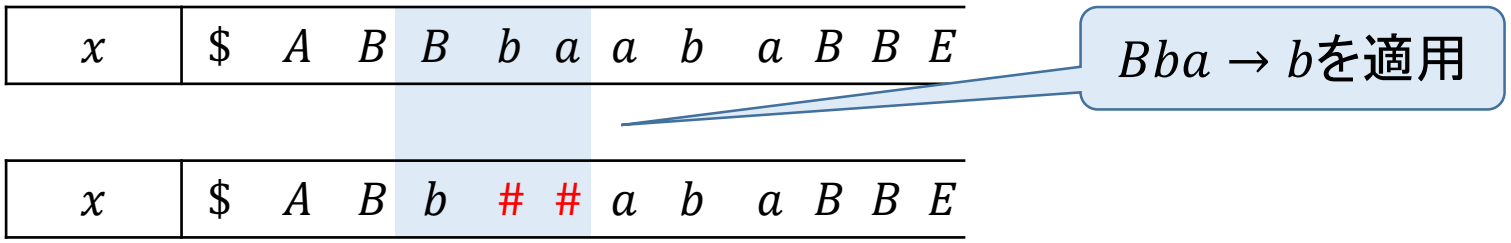
x	\$ A A b b b a B B E ⋯

(3)

x	\$ A B B b a a b a B B E

2. 0型文法 (6)

規則$\alpha \to \beta$でβの長さがαの長さよりも短いときは、
列は途中で短くなる。
この場合、特殊な記号#を使ってテープ上の列が短くならないようにする。#は常に読み飛ばすとする。



上記の動作を続けるとやがて#を除いて$の右側が終端記号だけになる。このときテープが#を除いて$x\$x$の形をしていれば受理、そうでないなら非受理とする。

Gがxを生成するなら正しいguessの系列が存在するので受理される。逆にMがxを受理するなら、Gがxを生成するのも明らか。

2. 0型文法 (7)

次に与えられたTM Mを模倣する文法Gを構成する。

具体例として、
Mの状態は$\{s_0, s_1, s_2, s_3\}$でs_0が初期状態、s_3が受理状態。
入力記号は$\{a, b\}$
入力記号以外のテープ記号は$\{X, Y, Z, E\}$で、Eは空白記号。
また、
Mは入力記号a, bを読むと入力記号以外に書き変えるとする。

構成するGの終端記号は$\{a, b\}$、変数は
$S, S_1, A, B, A_X, A_Y, A_Z, B_X, B_Y, B_Z, E_X, E_Y, E_Z, \$, \&, H_0, H_1, H_2, H_3, T$
とする。
A, BはMの入力記号a, bに対応。
H_0, H_1, H_2, H_3はヘッドの位置と状態を表すための変数。

2. 0型文法 (8)

TMは入力列が与えられてから動作を開始する。
一方、文法Gに入力列は最初に与えられはしない。
よって、Gは入力列を自分で作る。

①指定されたアルファベット上の任意の列を変数の形で作る。

②Mを模倣

③Mが受理するなら、最初の列を終端記号列の形で生成する。
　Mが受理しないなら、どのようにしても変数が残るようにする。

・・・のように進める。

2. 0型文法 (9)

Gの生成規則の第1グループ:

$$S \to \$H_0 S_1 \&, \qquad S_1 \to A S_1, \qquad S_1 \to B S_1, \qquad S_1 \to \epsilon$$

これを使えば、任意の$\mathcal{X} \in \{A, B\}^*$に対して,

$$S \overset{*}{\Rightarrow} \$H_0 \mathcal{X} \&$$

となることが分かる。

(例) $S \Rightarrow \$H_0 S_1 \& \Rightarrow \$H_0 A S_1 \& \Rightarrow \$H_0 A B S_1 \& \overset{*}{\Rightarrow} \$H_0 ABBAAA\&$

$\mathcal{X}$をMの入力列xと考える。 Aをa, Bをbと考える。

ここで、H_0はヘッドが入力の左端の文字の上にあって、
状態が初期状態s_0であることを示す。

以上で、Mの初期様相を作りあげた。

2. 0型文法 (10)

Gの生成規則の第2グループの導入:
初期様相(1)から、Mは$\delta(s_0, a) = (s_2, X, R)$に従って、1ステップの動作を行ったとする。
すると対応する文法での導出結果は(2)のように、

$$\$A_X H_2 BBAAA\&$$

となる。
ここで単純にAをXで書き変えずA_Xとしたのは、
<u>最初の列$ABBAAA$の情報を最後まで残しておく</u>ため

(1) $\$H_0 ABBAAA\&$

a	b	b	a	a	a	E	$\cdots$

$\Uparrow s_0$

(2) $\$A_X H_2 BBAAA\&$

X	b	b	a	a	a	E	$\cdots$

$\Uparrow s_2$

このステップでの生成規則は$\$H_0 A \rightarrow \$A_X H_2$を用意すればよい。

2. 0型文法 (11)

$$\$H_0A \rightarrow \$A_XH_2$$

左辺の左端が$になっているのは、
ヘッドがテープの左端にあるという特別な事情による。

一般的には、

$$A_XH_1A_Y \rightarrow H_2A_XA_X \Leftrightarrow \delta(s_1, Y) = (s_2, X, L)$$

のように、ヘッドの変数の両側はAやA_Xなどのテープ記号に対応する変数となる。

上記はヘッドが左に動く場合で、ヘッドが右に動く場合は、

$$A_XH_2B \rightarrow A_XB_YH_2 \Leftrightarrow \delta(s_2, b) = (s_2, Y, R)$$

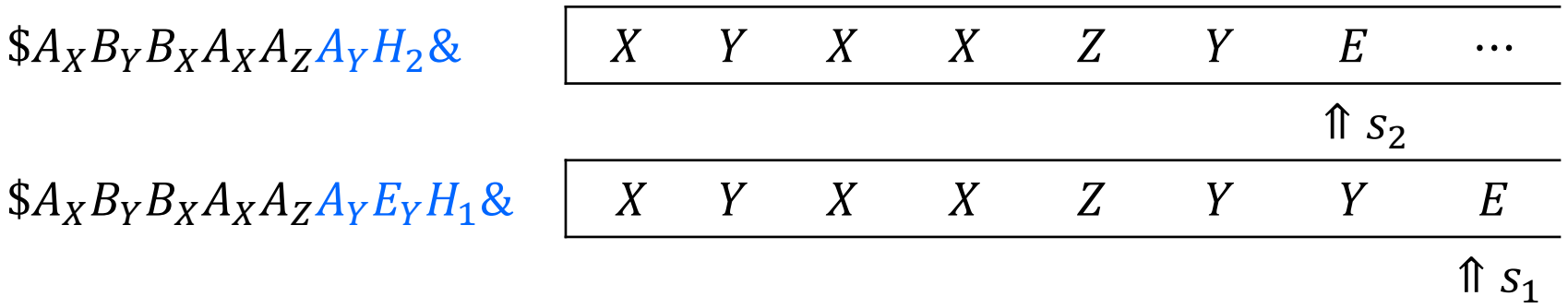
のようになる。

2. 0型文法 (12)

ヘッドが右端にある場合は、

$$A_Y H_2 \& \rightarrow A_Y E_Y H_1 \& \Leftrightarrow \delta(s_2, E) = (s_2, Y, R)$$

のようになる。この場合は、導出される列の長さが1増加する。



(注意)上記の規則でA_Yを付けておく必要があるのは、左へ進むMの動作へ対応するため。

2. 0型文法 (13)

Mが受理状態に入ったとする。この後は、現在の列を元の入力列xに戻す作業を行う。
最初に生成規則

$$H_3 \to TT$$

を導入する。(H_3に対応するs_3は受理状態)

次に2個のTを左と右に動かしていくことで、A_Xなどの記号を元の終端記号に変えていく。
必要な規則は、

$TA \to aT, TA_X \to aT, TA_Y \to aT, TA_Z \to aT, TB \to bT, \cdots, TE_X \to T,$
$AT \to Ta, A_XT \to Ta, A_YT \to Ta, A_ZT \to Ta, BT \to Tb, \cdots, E_XT \to T,$

である。(E_XはMでは元は空白のマス目だった)

最後に、$\$T \to \epsilon$と$T\& \to \epsilon$を使うと列$x$が導かれる。

2. 0型文法 (14)

受理状態に入ってから、入力列を作り直す例：

$$
\begin{aligned}
&\$A_X B_Y B_Y A_X A_X H_3 A_X E_X E_Y \& \\
\Rightarrow\ &\$A_X B_Y B_Y A_X A_X T T A_X E_X E_Y \& \\
\Rightarrow\ &\$A_X B_Y B_Y A_X T a T A_X E_X E_Y \& \\
\Rightarrow\ &\$A_X B_Y B_Y T a a T A_X E_X E_Y \& \\
&\Rightarrow \cdots \Rightarrow \cdots \\
\Rightarrow\ &\$T a b b a a T A_X E_X E_Y \& \\
&\Rightarrow \cdots \Rightarrow \cdots \\
\Rightarrow\ &\$T a b b a a a T \& \\
\Rightarrow\ &a b b a a a
\end{aligned}
$$

以上の方法で、Mを模倣する文法を作れる。

3. チューリング機械によっても認識できない言語 (1)

最も強力なオートマトンであるチューリング機械(TM)でも認識できない言語が存在する。

その存在を証明するために、『TMのTMによる模倣』を説明する。

状態遷移図は、本質的にはプログラムそのものであった。
プログラムは通常、『図』ではなく『文章』で書くものである。

よって、ここではまず状態遷移図を含めてTMを記号列で表現する方法を与える。
⇒　チューリング機械の符号化　とよぶ。

3. チューリング機械によっても認識できない言語 (2)

チューリング機械の符号化

特定のTMは、状態、テープ記号、状態遷移関数などで定まる。

状態は$s_0, s_1, \cdots$だとすると、これは、$q_1, q_2, \cdots$としても、一般性は失わない。

よって、この方針で記法を標準化する:

l個の状態: $q_1, q_2, \cdots, q_l$

m個の入力記号: $a_1, a_2, \cdots, a_m$

テープ記号: $a_1, a_2, \cdots, a_m, a_{m+1}, \cdots, a_n$

初期状態: q_1

空白記号: a_n

受理状態(k個): $q_{i_1}, q_{i_2}, \cdots q_{i_k}$

3. チューリング機械によっても認識できない言語 (3)

l個の状態 : $q_1, q_2, \cdots, q_l$

初期状態 : q_1　　受理状態(k個) : $q_{i_1}, q_{i_2}, \cdots q_{i_k}$

m個の入力記号 : $a_1, a_2, \cdots, a_m$　　空白記号 : a_n

テープ記号 : $a_1, a_2, \cdots, a_m, a_{m+1}, \cdots, a_n$

以上の制限を満たすTMを0, 1の列で符号化する :

$$11110^{l}110^{m}110^{n}1110^{i_1}110^{i_2}\cdots110^{i_k}111c_1 11c_2 11\cdots 11c_h 1111$$

- 重要な情報は0の数に符号化してある。
- 使う状態の名前は、既に決まっているので個数だけ示せばよい。
- $c_1, \cdots, c_h$は状態遷移関数を示した部分。詳細は後述。

3. チューリング機械によっても認識できない言語 (4)

$$11110^{l}110^{m}110^{n}1110^{i_1}110^{i_2}\cdots 110^{i_k}111c_1 11c_2 11\cdots 11c_h 1111$$

(q_{j_1}, a_{j_2})に対する状態遷移関数δの値が下記とする:

$$\delta(q_{j_1}, a_{j_2}) = (q_{j_3}, a_{j_4}, D)$$

これを一つのc_iで表す:

$$c_i = 0^{j_1}10^{j_2}10^{j_3}10^{j_4}10^{j_5}$$

ここで、

$$j_5 = 1 \Leftrightarrow D = L$$
$$j_5 = 2 \Leftrightarrow D = R$$

とする。

以上で、どのようなTMも0と1の列で表現できるようになった。
TM Mを上記のように符号化した列を$E(M)$と表記する。

3. チューリング機械によっても認識できない言語 (5)

次に入力列を符号化することを考える。
$x = a_{i_1} a_{i_2} \cdots a_{i_p}$のとき、符号化した列$E(x)$を以下で定義:

$$E(x) = 10^{i_1} 10^{i_2} \cdots 10^{i_p}$$

シミュレータとなるTM Tには $E(M)E(x)$として入力が与えれるとする。

定義:
Mがxを受理するとき、またその時に限ってTが$E(M)E(x)$を受理するならば、TM Tが、入力xが与えられたTM Mを模倣すると呼ぶ。

なお、$E(M)E(x)$以外の形の入力列はTは全て受理しないこととする。

3. チューリング機械によっても認識できない言語 (6)

定義
任意のTM Mを模倣するTM Tを万能チューリング機械と呼ぶ。

万能TMはプログラム内蔵方式の汎用計算機に対応する。

例題 5.5　万能チューリング機械を設計せよ。
解答:
ここで与えるTM Tは、具体的なTMで、状態遷移関数を実際に書き下すことができることに注意する。

3. チューリング機械によっても認識できない言語 (7)

Tは5本のテープを使用する。

1番目のテープ：入力$E(M)E(x)$　が与えられる。

2番目のテープ：模倣するTM Mのテープに対応
最初に$E(x) = 10^{i_1}10^{i_2}\cdots 10^{i_p}$をこのテープにコピー、
ヘッドを左端の1のところに戻す。

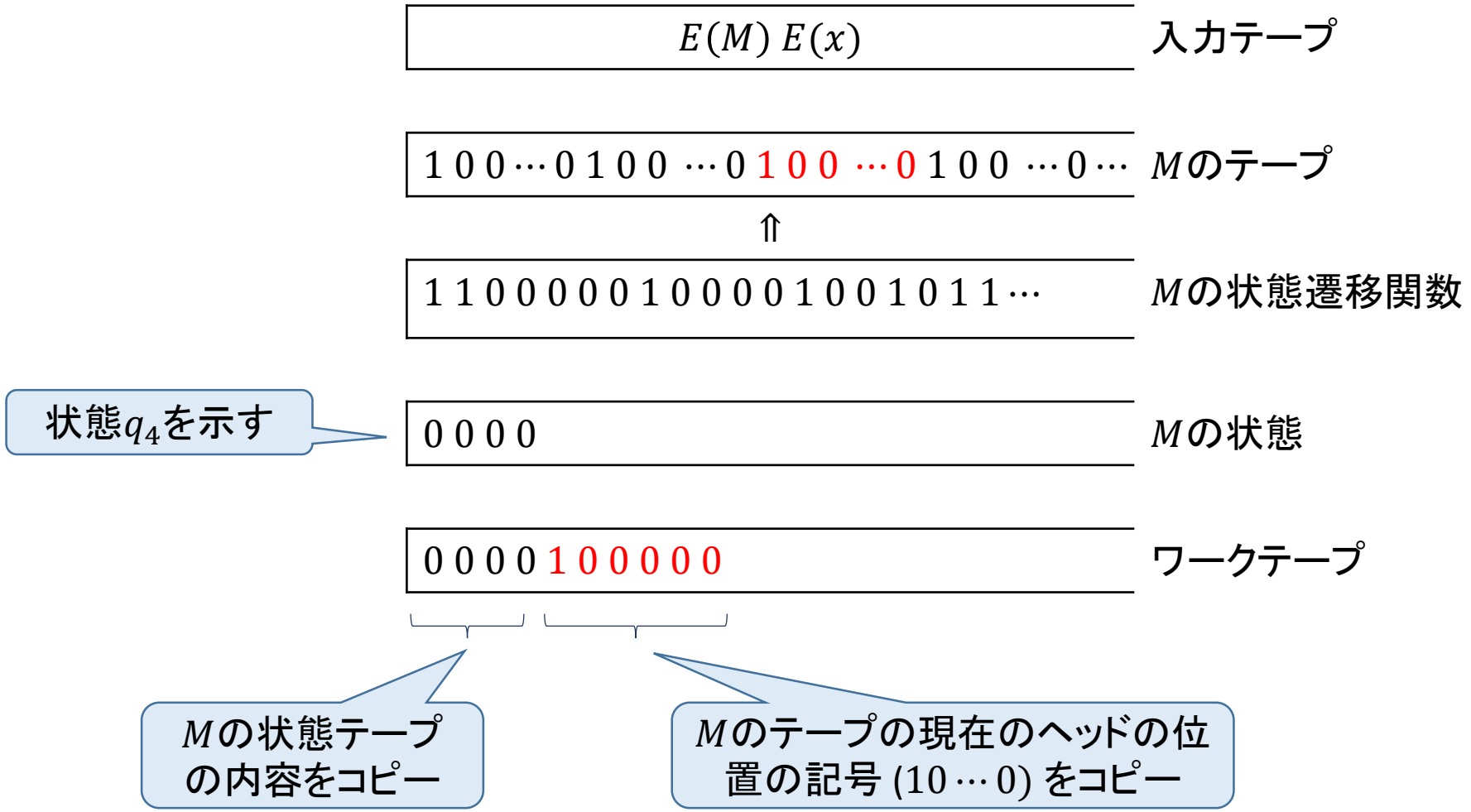
3番目のテープ：$E(M)$の後半の、
Mの状態遷移関数に対応する部分をコピーする。

4場目のテープ：Mの現在の状態を保持するために使用。
最初は0を1個入れる ⇔初期状態 q_1に対応

5番目のテープ：ワークスペース

3. チューリング機械によっても認識できない言語 (8)

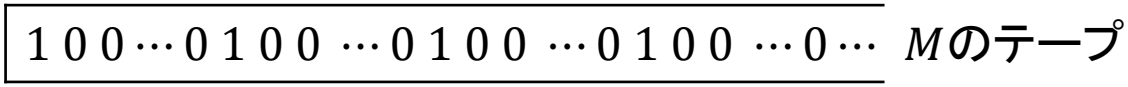
5つのテープを持つ万能TM



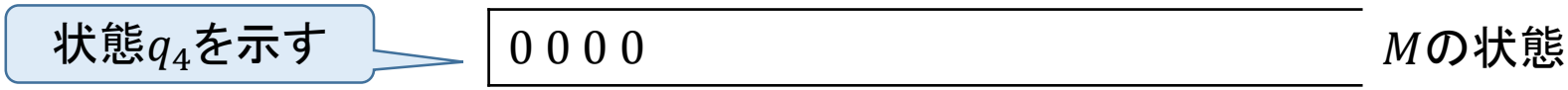
3. チューリング機械によっても認識できない言語 (9)

Mの1ステップの動作のTでの模倣の方法:
各ステップの模倣が始まる時点で、

- Tの第2のテープのヘッドの位置はMのヘッドの位置と合致している。
  (Mの一つのテープ記号は$10\cdots0$で符号化されているので、1の上にヘッドがある。)



- 第4のテープはMの現在の状態を正しく示している。
  (q_iならi個の0を保持している。)

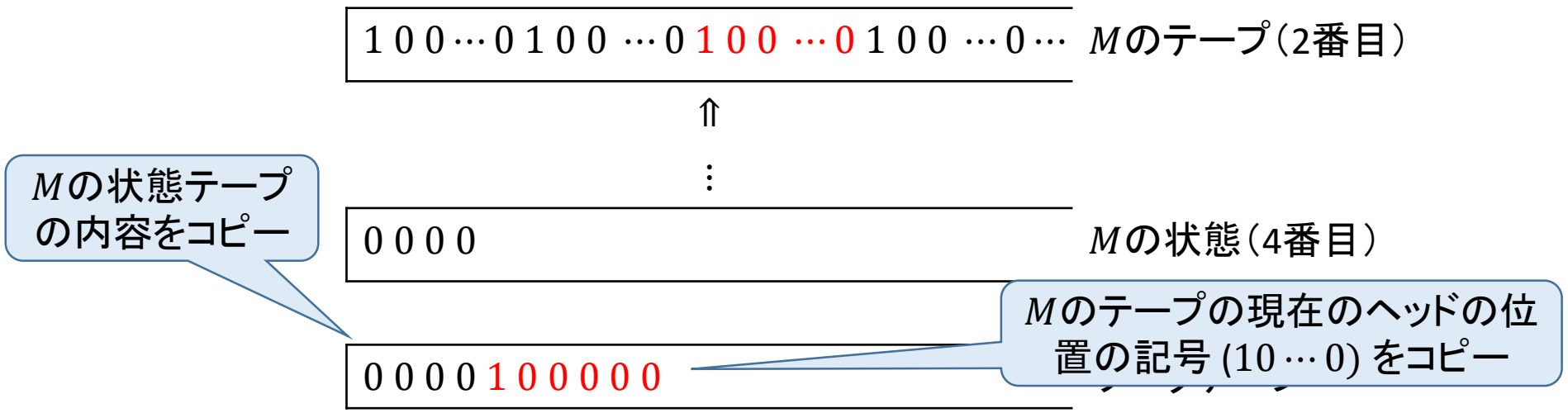


以上が成立しているとする。なお最初のステップでは成立している。

3. チューリング機械によっても認識できない言語 (10)

① Mの現在の状態が受理状態であるかをチェックする。
第4のテープの0の個数が$E(M)$で指定された受理状態の中に入っているかどうかを、第1のテープにアクセスしながらチェックすればよい。もし受理状態なら受理をして停止。

② 第4の状態テープの内容をワークスペースにコピーする。
更にワークテープにMのヘッドが現在読んでいる記号をコピーする。(単に第2のテープの現在のヘッド位置から右に0が続く限りコピーすればよい。)



3. チューリング機械によっても認識できない言語 (11)

以上でワークテープには、$0^{j_1}10^{j_2}$の形の列が出来上がっている。
これはδを表している$c_i = 0^{j_1}10^{j_2}10^{j_3}10^{j_4}10^{j_5}$の前半に対応

③ ワークテープの内容が前半と一致するc_iを第3のテープから探す。(パターンマッチングすればよい。)
見つからなかったら、非受理で停止する。
④ そのようなc_iが見つかったら、その後半部に従って第2のテープ(Mのテープ)の内容とヘッドの位置、第4のテープ(Mの状態)を更新する。
第2のテープの更新では、書き込むべき 0 の個数が前と異なるので、テープの右の部分全体をシフトする作業を行う。

以上の方針で万能TM Tが設計できる。

3. チューリング機械によっても認識できない言語 (12)

万能TMを用いて、どんなTMをもってきても認識できない$\{0,1\}$上の言語が存在することを示す。

$\{0,1\}$を入力とするTMの符号化を一つ固定する。
（万能TMのところで説明した符号化でもよいが、今は入力が$\{0,1\}$に限られているので、入力記号の符号化は省略できる。）

$\sigma_1 = 0, \sigma_2 = 1, \sigma_3 = 00, \sigma_4 = 01, \cdots$と
$\{0,1\}$上の列を短いものから順に、同じ長さの列は二進数読みで小さい数の順に並べる。
このときのi番目の列がσ_iとなるようにする。

縦にも横にも、$\sigma_1, \sigma_2, \sigma_3, \cdots$でラベル付けされた無限に大きい表を考える。

3. チューリング機械によっても認識できない言語 (13)

表の(i, j)要素に次のルールで〇と×を付ける:

① TM σ_iが列σ_jを受理すれば〇

② TM σ_iが列σ_jを受理しないとき、

もしくはσ_iがTMの符号語でないときは×

ここでTM σ_iは符号化するとσ_iになるTMのことである。

	σ_1	σ_2	σ_3	σ_4	…	σ_j	…
σ_1	×	×	×	×		×	
σ_2							
σ_3							
σ_4							
⋮							
σ_i	〇	〇	×	×		×	
⋮							

TM σ_iが列のσ_jを受理するなら〇，受理しないなら×を書く。

3. チューリング機械によっても認識できない言語 (14)

TM σ_iが列のσ_jを受理するなら〇，受理しないなら×を書く。

	σ_1	σ_2	σ_3	σ_4	…	σ_j	…
σ_1	×	×	×	×		×	
σ_2		〇					
σ_3			〇				
σ_4				×			
⋮							
σ_i	〇	〇	×	×		×	
⋮							

このように作成した表を基にして、次のように言語Lを定義：

$$L = \{\sigma_i \mid \text{TM } \sigma_i \text{が列} \sigma_i \text{を受理しない}\} \quad -(*)$$

つまり、i行i列のエントリーが×ならば列σ_iはLに入り、
〇ならば入らない。

定理 5.4 言語LはいかなるTMによっても認識されない。

3. チューリング機械によっても認識できない言語 (14')

$$L = \{\sigma_i \mid \text{TM } \sigma_i \text{が列} \sigma_i \text{を受理しない}\} \quad -(*)$$

定理 5.4 言語LはいかなるTMによっても認識されない。

証明: 対角線論法を用いる。
Lを認識するTMが存在するとしてそれをMとする:

$$L = \{\sigma_i \mid \text{TM } M \text{が列} \sigma_i \text{を受理する}\} \quad -(**)$$

Mを符号化した列をσ_Mとする。

① TM Mが列σ_Mを受理するとき: $(**)$より、$\sigma_M \in L$

一方、『TM σ_Mが列σ_Mを受理する』ので、
表のσ_M行σ_M列は〇となる。よって、$(*)$より$\sigma_M \notin L$で矛盾。

3. チューリング機械によっても認識できない言語 (14’’)

$$L = \{\sigma_i \mid \text{TM } \sigma_i \text{が列} \sigma_i \text{を受理しない}\} \quad -(*)$$

Lを認識するTMが存在するとしてそれをMとする:

$$L = \{\sigma_i \mid \text{TM } M \text{が列} \sigma_i \text{を受理する}\} \quad -(**)$$

Mを符号化した列をσ_Mとする。

② 同様に、TM Mが列σ_Mを受理しないとき: $(**)$より、$\sigma_M \notin L$

一方、『TM σ_Mが列σ_Mを受理しない』ので、
表のσ_M行σ_M列は×になる。よって、$(*)$より$\sigma_M \in L$で矛盾。

以上より、Lを認識するTMは存在しない。

３. チューリング機械によっても認識できない言語 (１５)

例題　$L = \{\sigma_i \mid \text{TM } \sigma_i \text{が列} \sigma_i \text{を受理しない}\}$ を認識するTM を万能TMを使って無理やり作ることを考える。

与えられた入力列$x \in \{0,1\}^*$に対して、
符号化がxになるTM M_xにxを入力したときの動作を、
万能TMを使って模倣する:
①M_xがxを受理すれば、Tはxを受理しない。
②M_xがxを受理しなければ、Tはxを受理する。

この議論のどこがおかしいのか？

3. チューリング機械によっても認識できない言語 (15')

符号化がxになるTM M_xにxを入力したときの動作を、
万能TMを使って模倣する:
①M_xがxを受理すれば、Tはxを受理しない。
②M_xがxを受理しなければ、Tはxを受理する。
この議論のどこがおかしいのか？

解答:①はできるが、②は一般的にできない。

M_xがxを受理しない場合には、M_xが永遠に止まらない場合が含まれる。
よって、M_xがxを受理しないことを、M_xの動作を模倣することで確かめることはできない。
（次の章で詳しく説明する。）

補遺　ランダムアクセス機械の模倣(1)

チューリング機械が、我々が通常使っているコンピュータを模倣できることを示す。

ランダムアクセス機械(RAM)を以下のように定義する。

- RAMはメモリ、プログラム、メモリアドレスレジスタ(MAR)、複数の汎用レジスタ($R_0, R_1, \cdots$)からなる。
- メモリはアドレスのついたワードに分かれているが、各ワードにはいくらでも大きな2進数が入り得る。また、メモリ全体のワード数にも制限はない。
- 命令は

  $ADD\ R_1\ (2000), \qquad L\ R_2\ (1800), \qquad ST\ R_1\ (1010)$

  $BRZ\ R_1\ 25$

  などを用いる。

補遺　ランダムアクセス機械の模倣(2)

可算命令：　$ADD\ R_1\ (2000)$
　2000番地の内容をMARにいれてメモリにアクセスする。
（間接アドレス方式）
もし、2000番地に12500が入っているなら、12500番地の値をレジスタR_1の値に足し込む。

ロード命令：　$L\ R_2\ (1800)$　　メモリの値をレジスタにロード

ストア命令：　$ST\ R_1\ (1010)$
　　ADDと同様だが、レジスタ値をメモリ書き込む

分岐命令：　$BRZ\ R_1\ 25$
R_1の値が0ならラベル25のついた命令に進み、0でないなら次の命令を実行する。

補遺　ランダムアクセス機械の模倣(3)

プログラムはROMに書かれていて変更不能とする。
⇔ 万能TMと同様に万能RAM (つまり汎用計算機)が存在する。

多テープTMを用いてRAMの動作を模倣する:

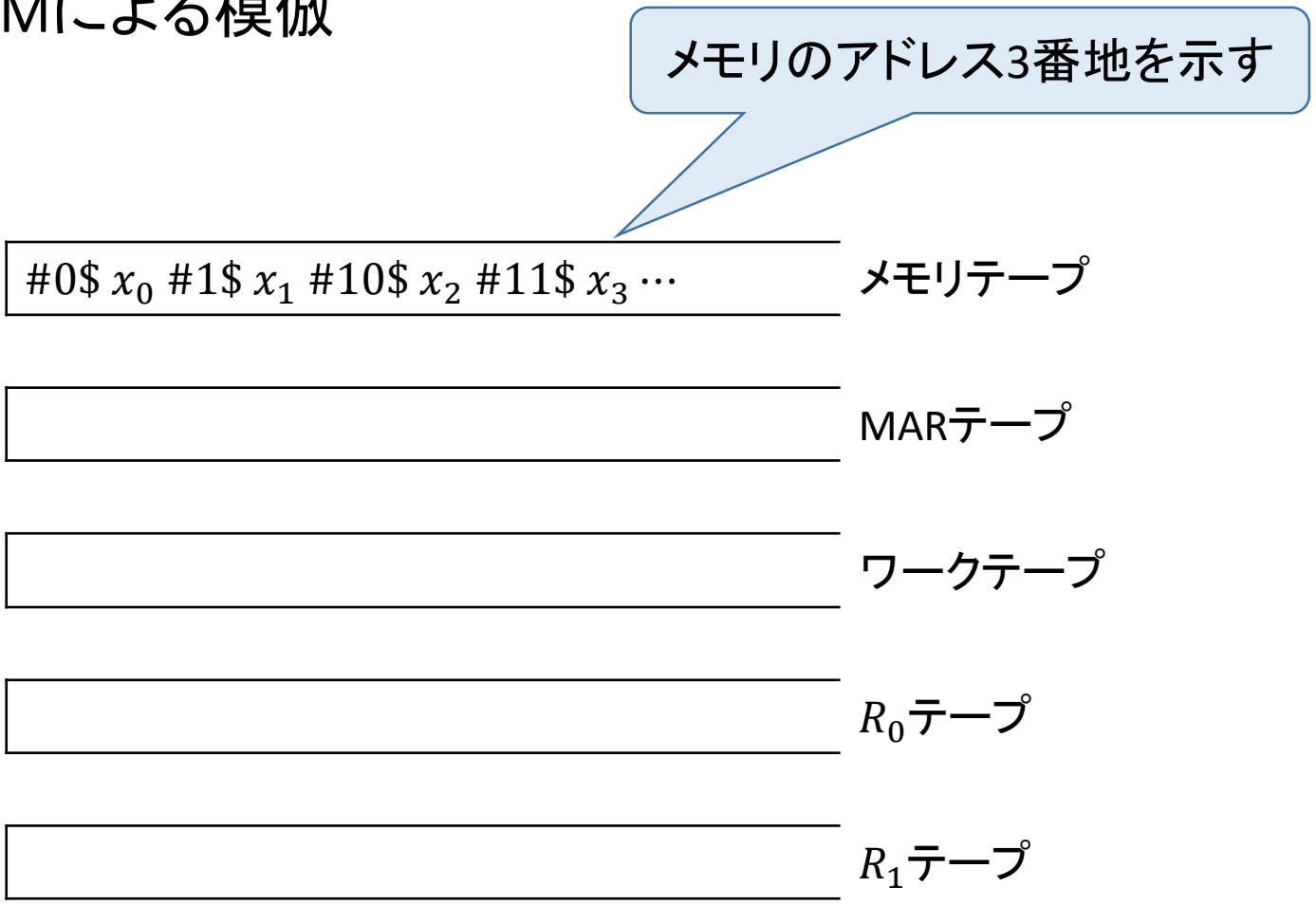
- 最初のテープ＝メモリテープ:RAMのメモリを模倣する。
  RAMの0番地、1番地、2番地・・・に対応して、
  $\#0\$, \#1\$, \cdots, \#10\$, \#11\$$,のようにテープに番地を振る。

- MARレジスタ、汎用レジスタ、などを模倣するテープを用意する。

- ワークテープをいくつか用意する。

補遺　ランダムアクセス機械の模倣(4)

RAMのTMによる模倣



補遺　ランダムアクセス機械の模倣(5)

$ADD\ R_1\ (2000)$の模倣:

① メモリテープのヘッドを2000番地に移動する。
(2000という数は、必ず具体的なある値が指定されているので、有限状態を用いて実行できる。)

② 2000番地にヘッドが到達したら、そこの値(いくらでも大きくなりうる)をMARテープにコピーする。

③ MARテープの内容の番地までテープを移動する。
・まず、ヘッドを左端に移動する。
・次に、最初の番地部(#0$右隣)にヘッドを移動させる。
・その内容とMARテープの値の一致検索を行う。
・一致しなければ、一つ右の番地部にヘッドを移動させて同じことを行う。

補遺　ランダムアクセス機械の模倣(6)

$ADD\ R_1\ (2000)$の模倣の続き:

④ いつかは一致するので、一致したらワードの内容をいったんワークテープにコピーする。

⑤ R_1の内容とワークテープの値を足してその値を別のワークテープに格納する。
この足し算は、普通の筆算と同様に下の桁から計算するとよい。

⑥ ワークテープの内容をR_1に写す。

他の命令も同様に模倣をすることができる。

このように、チューリング機械は、我々が通常使っているコンピュータを模倣できる。

第六章：チューリング機械の停止性と決定問題

『必ず停止する』チューリング機械と『計算機では解けない問題』の関係について

0. チューリング機械と計算機
1. チューリング機械の停止性
2. 非可解な（計算可能でない）問題
3. ポストの対応問題

0.チューリング機械と計算機 （1）

チューリング機械は、入力列$x \in \{0,1\}^*$を『受理する(yes)』か、『受理しない(no)』かを判定する機械として定義された。

TM Mが認識する言語を$L(M)$とすると、
Mは入力列xが$L(M)$に入るかを判定する機械といえる。

関数$f_M(x): \{0,1\}^* \to \{0,1\}$を、

$$f_M(x) = \begin{cases} 1 & \left(x \in L(M)\right) \\ 0 & \left(x \notin L(M)\right) \end{cases}$$

と定義すると、
Mは二値関数$f_M(x)$を計算する機械といえる。

なぜ、チューリング機械は計算機のモデルと言えるのだろうか？

0.チューリング機械と計算機 (2)

計算機の歴史:

- 紀元前数千年: 『そろばん』など
- 古代ギリシャ: 最古のアナログコンピュータ
  歯車式:天球上の天体の位置を計算
- 1642年:パスカルの機械式計算機(加算と減算)
- 1672年:ライプニッツの機械式計算機(加減乗除)
- 以降20世紀初頭まで機械式計算機の時代(量産・小型化)
- 19世紀:バベッジの階差機関(未完):多項式の自動計算
  解析機関(設計のみ):プログラム可能機械式計算機

- 1936年:チューリングが(万能)チューリング機械を論文発表
- 1938年:ツーゼ Z1: プログラム可能機械式計算機
- 1940年代:コンピュータ(デジタル計算機)の発明
  ABC(1942)、Colossus(1943)、ENIAC(1946)、The Baby(1948)

0.チューリング機械と計算機　(3)

『計算機(コンピュータ)』とは何か？

⇒入力された列$x \in \{0,1\}^*$に対し、(プログラムに従って計算し)列$y \in \{0,1\}^*$を出力する機械
⇔入力された列$x \in \{0,1\}^*$に対して、$\{0,1\}^*$上の関数$f(x)$を(プログラムに従って)計算する機械

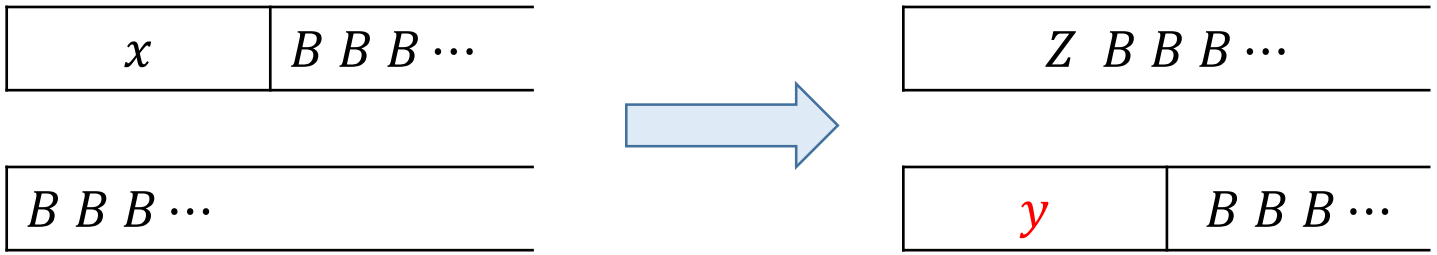
とするのが自然と思われる

一方、チューリング機械は二値$\{0,1\}$のみを出力する機械であった。

⇒入力列$x \in \{0,1\}^*$に対して列$y \in \{0,1\}^*$を出力するようなチューリング機械(チューリング変換機)を定義してみる。

0.チューリング機械と計算機　(4)

チューリング機械(チューリング変換機):
入力$x \in \Sigma^*$に対する出力$y \in \Sigma^*$を定義する。
定義1:　2テープチューリング機械を考える。
　受理状態に到達したときに、2テープ目に書かれている列$y \in \Sigma^*$を出力列であると定義する。



定義2:　1テープチューリング機械を考える。
　受理状態に到達したときに、テープの左端から最初に入力記号以外の記号が現れるまで列を出力列$y \in \Sigma^*$とする。

0.チューリング機械と計算機 (5)

今、$D \subseteq \{0,1\}^*$を定義域とする。
$\{0,1\}^*$上の部分関数$f: D \to \{0,1\}^*$ を計算することを考える。
入力$x \in \{0,1\}^*$に対して、
$x \in D$を受理して、$y = f(x) \in \{0,1\}^*$ を出力し、
$x \notin D$は受理しないチューリング変換機があれば、
この関数は『計算できる』といえると思える。

$\{0,1,\$\}^*$上の言語$L_f$を以下のように定義する:

$$L_f = \{x\$y \mid x \in D, y \in \{0,1\}^*, y = f(x)\}.$$

もし、L_fを認識するチューリング機械Tがあれば、計算したい$f(x)$に対して、色々な$y \in \{0,1\}^*$に対して、$x\$y$を$T$の入力列とすれば、そのうち受理される列$x\$y$が見つかり、$y = f(x)$が計算できる。

⇒通常のTMで関数の計算をすることができる。
(実は、この議論にはTMの停止性の話が抜けている。)

1.チューリング機械の停止性（1）

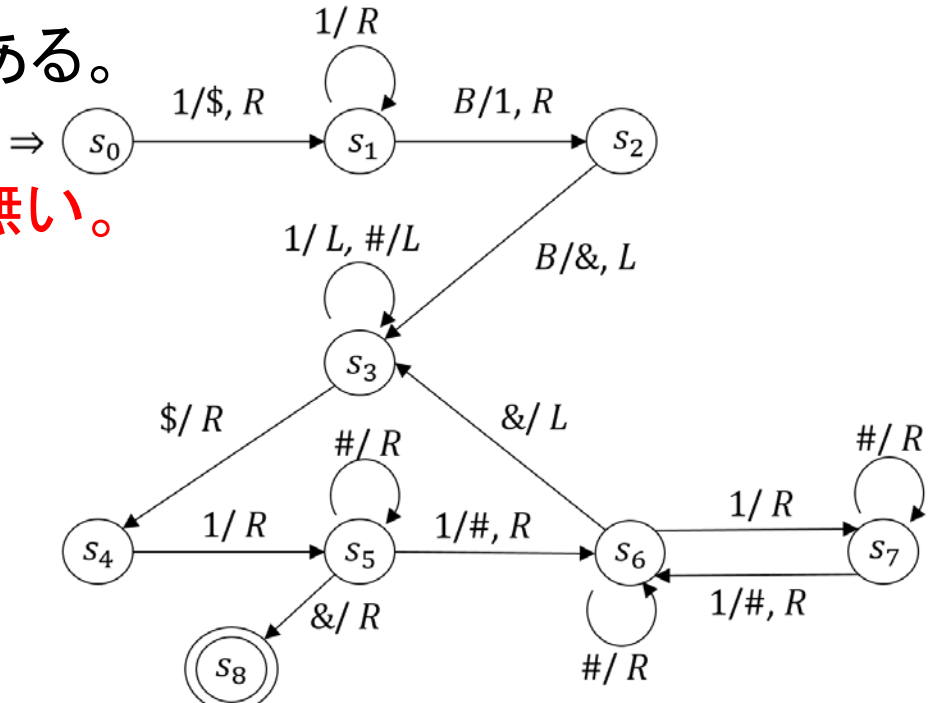
チューリング機械が列xを受理するとは、受理状態に遷移することであった。受理状態に遷移した後は、TMは停止するように設計するのが自然である。

一方で、非受理であるとは、受理状態に到達しないことであった。

以前の例なら、非受理には2通りある。

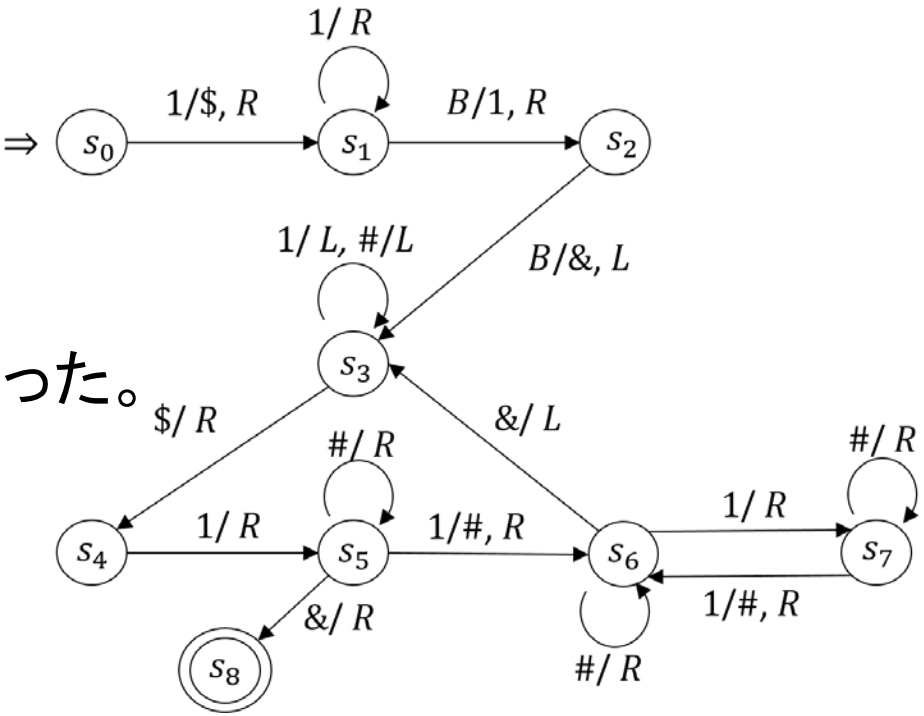
① 次の遷移先が状態遷移図に無い。
② 永遠に動き続ける。

厳密に話をするため、
TMを定義しなおす。



1.チューリング機械の停止性 (2)

左図の場合では、
読んだ記号と状態の組によっては
次の遷移が
状態遷移図に存在しない場合があった。



このことを反映させるため、
チューリング機械の中の状態遷移関数δを部分関数として再定義する。

状態遷移関数δは定義域$D \subseteq K \times \Gamma$において定義されるとする。

$$\delta: D \to K \times \Gamma \times \{L, R\}$$

1.チューリング機械の停止性 (3)

状態遷移関数の定義域$D \subseteq K \times \Gamma$は、
任意の受理状態$s_f \in F$、任意の記号$a \in \Gamma$に対して、
(s_f, a)を含まないとする。

$$\forall s_j \in F, \forall a \in \Gamma \left((s_f, a) \notin D \right)$$

つまり、受理状態からの遷移は定義されていないとする。

定義
チューリング機械は、現在の状態$s \in K$と呼んだ記号$a \in \Gamma$に対して、次の遷移が定義されていないとき、つまり$(s, a) \notin D$のとき停止するという。

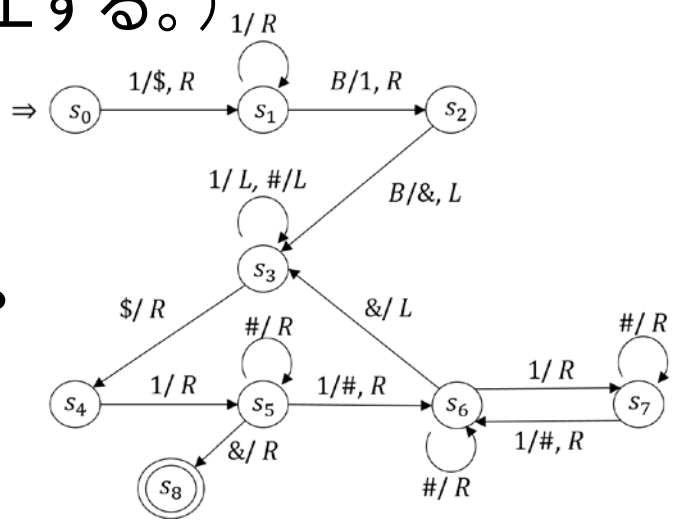
入力列$x \in \Sigma^*$に対して、受理状態以外の状態(非受理状態)で停止した場合、TMはxを拒否すると呼ぶ。

1.チューリング機械の停止性 (4)

TMが入力列xを非受理である場合には以下の2通りがある:

① TMはxを拒否する。(非受理状態で停止する。)
② それ以外(永遠に動き続ける。)

以前の例では、②の場合はなかった。
②の場合は、本当に存在するのだろうか？



疑問に答えるため、以下のようなTMのクラスを定義する

定義: 任意の入力$x \in \Sigma^*$に対して停止するTMを必ず停止するTMと呼ぶ。

1.チューリング機械の停止性 (5)

定理 6.1 必ず停止するTMでは認識できないが、そのような制限のないTMなら認識できる$\{0,1\}$上の言語が存在する。

（証明） 言語Lを以下で定義する:

$$L = \{\sigma_i \mid \text{TM}\sigma_i\text{は列}\sigma_i\text{を拒否する}\}$$

ここで、$\{0,1\}$入力のTMのみを考えて、入力列σ_iは符号化されていないと考えてもよいし、任意の入力記号に対するTMを考えて、σ_iは万能TMのときに使った符号化$E(M)$と$E(x)$であると考えてもよい。
(以降の同様の定理の証明では上記の説明を省略する)

Lを必ず停止するTMで認識できると仮定する。
Lを認識する必ず停止するTMの符号語をσ_Mとする。

1.チューリング機械の停止性 (6)

$$L = \{\sigma_i \mid \mathrm{TM}\sigma_i\text{は列}\sigma_i\text{を拒否する}\} \quad -(*)$$

Lを認識する必ず停止するTMの符号語をσ_Mとする:

$$L = \{\sigma_i \mid \mathrm{TM}\sigma_M\text{は列}\sigma_i\text{を受理する}\} \quad -(**)$$

TM σ_Mは列σ_Mに対して停止するので、以下の2通りが考えられる。

① TM σ_Mは列σ_Mを受理する場合:

TM σ_MはLを認識するので、$(**)$より $\sigma_M \in L$.

一方、Lの定義$(*)$より、$\sigma_M \notin L$となるので矛盾

② TM σ_Mは列σ_Mを拒否する場合:

TMσ_MはLを認識するので、$(**)$より$\sigma_M \notin L$.

一方、Lの定義$(*)$より、$\sigma_M \in L$となるので矛盾

よってLを認識する必ず停止するTMは存在しない。

1.チューリング機械の停止性　(6')

$$L = \{\sigma_i \mid \mathrm{TM}\sigma_i\text{は列}\sigma_i\text{を拒否する}\}$$

よってLを認識する必ず停止するTMは存在しない。

Lは必ず停止するとは限らないTM(受理するときは停止する)で認識できる。

実際に、万能TMを使って入力σ_iに対するTM σ_iの動作を模倣して、TM σ_iがσ_iを拒否したときのみ受理するとよい。　□

必ず停止するTMは、『文脈自由言語を全て認識する＝プッシュダウンオートマトンより強い』ことが証明できる。

2.非可解な問題 (1)

決定問題: 「与えられた列は何々の条件を満たすか？」というタイプの問題 (yes or no で答えれる問題)

Σ^*上の言語Lに、「与えられた列$x \in \Sigma^*$はLに入るか？」という決定問題を対応させれる。

Σ^*上の言語Lに対して、Lの特性関数$\chi_L : \Sigma^* \to \{0,1\}$を

$$\chi_L(x) = \begin{cases} 1 & (x \in L) \\ 0 & (x \notin L) \end{cases}$$

で定義する。

$1 = yes, 0 = no$とすると、特性関数$\chi_L(x)$はLに対応する決定問題の表現とみなせる。

2.非可解な問題 (2)

定義:
決定問題を解く必ず停止するTMが存在するとき、その決定問題は可解であると呼ぶ。
可解でない問題を非可解な問題と呼ぶ。

$$L = \{\sigma_i \mid \mathrm{TM}\sigma_i\text{は列}\sigma_i\text{を拒否する}\}$$

に対して、『与えられた$x \in \{0,1\}^*$は、Lに入るか？』は、
決定問題であり、Lを認識する必ず停止するTMが存在しないので、
非可解な問題である。

他の例を見てみる

2.非可解な問題 (3)

定理 6.2 (チューリング機械の停止問題)
チューリング機械Mに対して、Mが入力列$x \in \Sigma^*$に対して停止するかどうかの決定問題は非可解である。

（証明） まず、この問題が決定問題であることを確かめる。
チューリング機械は、符号化を使って$E(M) \in \{0,1\}^*$とできる。
同様にxも符号化により$E(x) \in \{0,1\}^*$となる。
列$E(M)E(x)$は正しい符号化になっていれば境界はわかる。

よって、『列$y \in \{0,1\}^*$が与えられたときに、yが$E(M)E(x)$の形をしており、かつ、TM Mが入力列xに対して停止するか？』
とすれば、決定問題になっている。

次に非可解性を示す。

2.非可解な問題 (4)

TMの停止問題を認識する必ず停止するTMが存在すると仮定する。

この仮定の下で、

$$L = \{\sigma_i \mid \mathrm{TM}\sigma_i\text{は列}\sigma_i\text{を拒否する}\}$$

を認識する必ず停止するTMが存在することを示すことを目指す。

上記のLを認識する必ず停止するTMは既に存在しないことを証明したので、矛盾が生じる。

2.非可解な問題 (4')

停止問題を認識する必ず停止するTM をM_hとする。

これを用いて

$$L = \{\sigma_i \mid \mathrm{TM}\sigma_i\text{は列}\sigma_i\text{を拒否する}\}$$

を認識するTM Tを以下のように構成する:

Tは与えられた入力xを右側にコピーして、
xxというテープを作る。

次にTは『入力xxに対するM_hの動作』を模倣する。
M_hは必ず停止するTMなので、xxに対して停止する。

① xxをM_hが拒否する場合: Tはxを非受理とする。
この場合、TM xは入力xに対して停止しないことが分かる。

2.非可解な問題 (5)

② M_hが入力xxを受理する場合:
　　TM xは入力xに対して停止することが分かる。

そこで、Tは入力xに対するTM xの動作を、万能TMの手法で模倣する。
TM xは入力xに対して停止するので、
TM x がxを拒否すれば、Tは受理状態で停止する。
TM x がxを受理すれば、Tは非受理状態で停止する。

2.非可解な問題 (5')

まとめると、TM Tの動作は以下になる:
TM xが入力xを拒否 ⇔ xを受理
そうでない ⇔ xを拒否

よって、TM Tは

$$L = \{\sigma_i \mid \mathrm{TM}\sigma_i\text{は列}\sigma_i\text{を拒否する}\}$$

を認識する必ず停止するTMである。

そのようなものは存在しないので、これは矛盾である。

2.非可解な問題 (6)

非可解な問題をTMで解こうとすると、答えがyesのときは停止するが、noのときは永遠に動き続ける可能性がある。

現実的には、我々が待つことができるのは有限の時間であるので、永遠に動き続けるのか、それとも長い時間の後、TMが停止するのか、我々には判断ができない。

⇒『非可解な問題は計算機では解けない』と考えるのが妥当。

可解な問題は、必ず停止するTMで認識できる言語に対応。
そこで、言語Lは、必ず停止するTMで認識できるとき計算可能であると呼ぶ。このようなLは帰納言語とも呼ばれる。

2.非可解な問題 (7)

チューリング変換機の話にもどる。

$f: D \to \{0,1\}^*$　を$D \subseteq \{0,1\}^*$を定義域とする$\{0,1\}^*$上の部分関数とする。
入力$x \in \{0,1\}^*$に対して、$x \in D$を受理して、$y = f(x) \in \{0,1\}^*$ を出力し、$x \notin D$を拒否するチューリング変換機があれば、
$f(x)$は計算可能であると呼ぶ。

$\{0,1,\$\}^*$上の言語$L$を以下のように定義する:

$$L = \{x\$y \mid x \in D, y \in \{0,1\}^*, y = f(x)\}.$$

もし、Lが計算可能であるならば、つまりLを認識し必ず停止するチューリング機械Tがあれば、計算したい$f(x)$に対して、色々な$y \in \{0,1\}^*$に対して、$x\$y$を$T$の入力列とすれば、そのうち$T$が受理する列$x\$y$が見つかり、$y$を出力できる。
よって、言語Lが計算可能⇔関数fが計算可能

2.非可解な問題 (8)

チャーチ＝チューリングのテーゼ(Church-Turing thesis):
実際の計算機(メモリはいくらでも使えるとする)で計算できる関数は、チューリング機械（変換機）で計算できる（計算可能な）関数である。

どのような新しい仕組みの計算機を設計しても、その計算機で計算できる関数は、チューリング機械(変換機)で計算可能な関数になるという経験則もしくは予測。

様々な計算機のモデルが存在する:
例:帰納的関数、ラムダ計算、ポストタグ機械、
ランダムアクセス機械、量子チューリング機械(量子計算機)

このテーゼは、全ての汎用計算機のモデルが、計算可能性に関しては等価であることを主張している。(今のところ正しい。)

2.非可解な問題 (9)

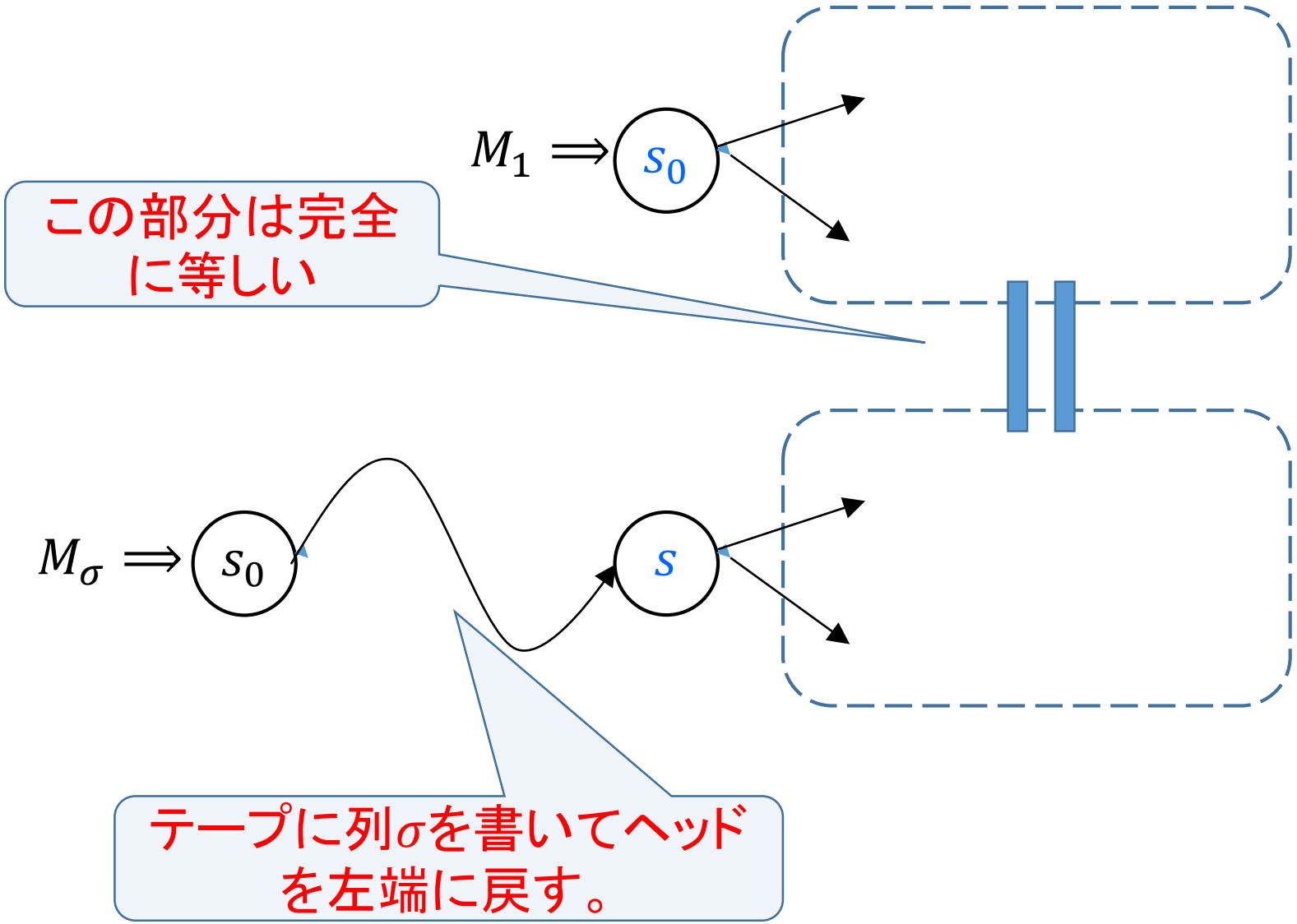
非可解な問題の例をいくつか見ていく。

定理 6.3 チューリング機械 Mが入力ϵに対して停止するかどうかの決定問題(ϵ停止問題)は非可解である。

証明:
TMの停止問題は非可解であった。
そこで、ϵ停止問題を解く必ず停止するTM T_ϵが存在すれば、
T_ϵ を使ってTMの停止問題を解く必ず停止するTMが作れることを示す。

任意のTM M_1と任意の列σに対して、TM M_σを以下で定義する:
M_σはϵテープが与えられると、まずテープにσを書き、ヘッドを左端に戻してからM_1の動作を始める。

2.非可解な問題 (10)


$M_1 \Longrightarrow s_0$
この部分は完全に等しい
$M_\sigma \Longrightarrow s_0$
s
テープに列σを書いてヘッドを左端に戻す。

２.非可解な問題 (１１)

TM M_1と列σ (の符号化)を入力として、TM M_σを出力するチューリング変換機Tを以下のように設計する。

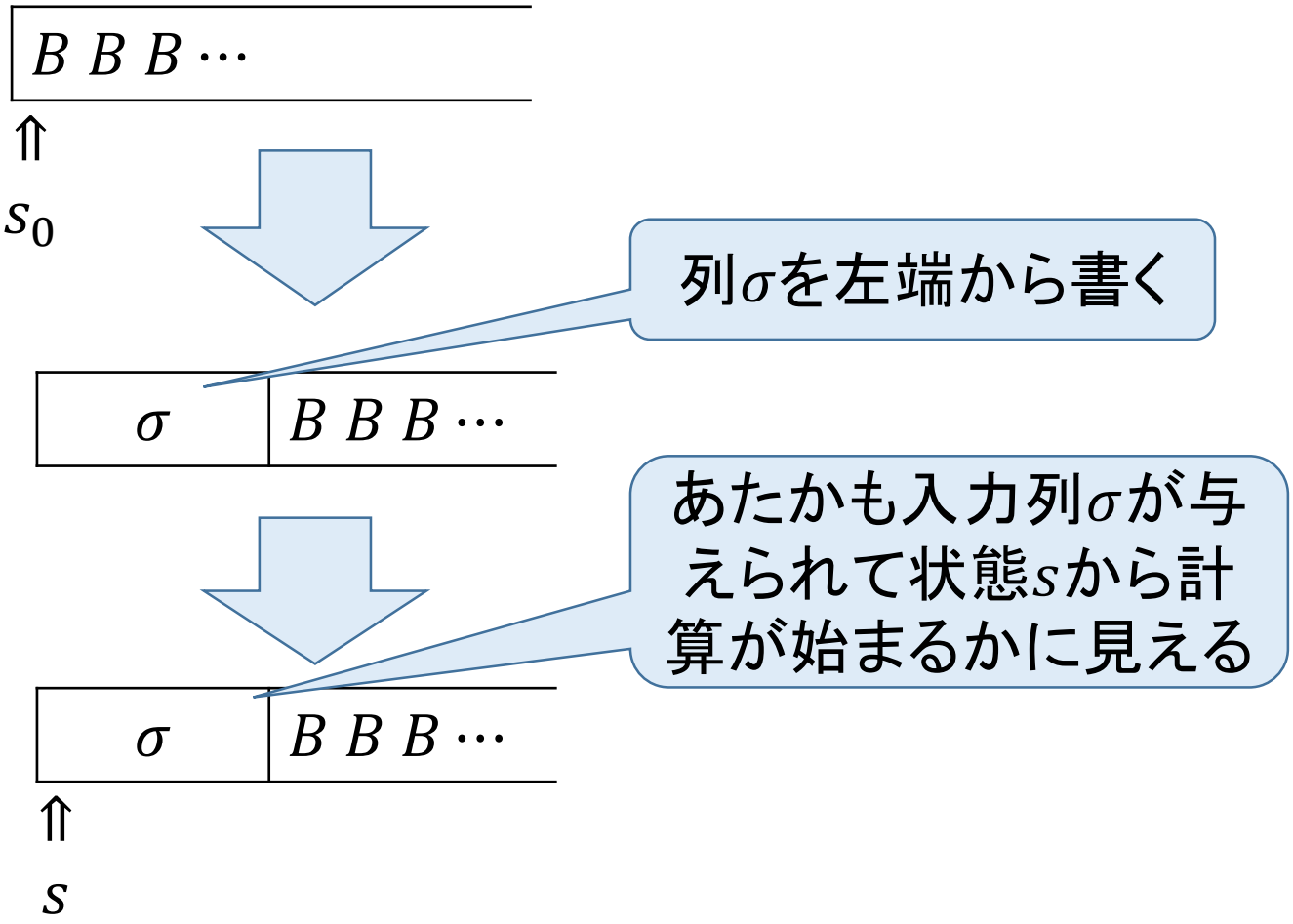
Tは以下の2ステップにより$E(M_\sigma)$を出力する:
① 入力のうち$E(M_1)$を書き変えて、M_1の初期状態をs_0ではなく、普通の状態sからはじまるようにする。

② 有限状態を利用することで、さらに、$E(M_1)$を、初期状態s_0からスタートして、テープに列σを書いて、ヘッドを左に戻し、その後にM_1を動作させるように書き変える。

Tは必ず停止するTM変換機となることが証明できる。
（CやJavaなどで必ず停止するプログラムを書けるような操作は、全て必ず停止するTMで実行できる。）

2.非可解な問題 (12)

M_σの動作



M_σがϵ入力に対して停止する必要十分条件は、
M_1がσに対して停止することである。

2.非可解な問題 (13)

ϵ停止問題を解くTM をT_ϵとして、
上記Tと TM T_ϵを連結させたTMを考える。
このTMは、入力(M_1, σ)に対して、まずTを動作して、M_σ(の符号化$E(M_\sigma)$)を得る。次にM_σに対してT_ϵを動作させる。

既に述べたように、M_σがϵ入力に対して停止する必要十分条件は、M_1がσに対して停止することである。

よって、このTMの答え(yes, no)が、(M_1, σ)の停止判定の正しい答えになっている。

Tは必ず停止するので、T_ϵが必ず停止すると仮定すると、連結したTMも必ず停止する。
これはTMの停止問題が可解であることを意味するので矛盾。 ■

2.非可解な問題 (14)

非可解性の標準的な証明法

P: 非可解性を証明したい問題(関数)
Q: 既に非可解であることが証明されている問題(関数)

『QをPに還元する』
$\Leftrightarrow$ Qへの入力列qに対してPへの入力列を対応させる写像
$t: q \mapsto p$で下記の性質を満たすものを見つける:

① $\forall q\ (Q(q) = yes \Leftrightarrow P(t(q)) = yes)$
qに対するQの答えがyesなら、対応するpに対するPの答えもyes

② qを入力として与えられたときに、
$t(q)$を出力する必ず停止するTM Tが存在する。

2.非可解な問題 (15)

Qへの入力列qに対してPへの入力列を対応させる写像$t: q \mapsto p$で下記の性質を満たすものを探す:

① $\forall q\ (Q(q) = yes \Leftrightarrow P(t(q)\ ) = yes)$
　qに対するQの答えがyesなら、対応するpに対するPの答えもyes

② qを入力として与えられたときに、
　$t(q)$を出力する必ず停止するTM Tが存在する。

上記の写像tが存在するならPも非可解である。

(証明)　Pを認識する必ず停止するTM T_Pの存在を仮定する。
TとT_Pを直列で接続したTM T_Qを考える。
T_Qはqを入力すると、まずTを回してpを作って、pをT_Pでyesかnoかを判定する。①②よりTM T_Qは問題Qを認識する必ず止まるTMとなっている。これはQの非可解性に矛盾する。

2.非可解な問題 (16)

例題 6.1　0型文法Gが列xを生成するかどうかの決定問題は非可解である。

（証明の概要）

最初にTMを符号化したように、0型文法も同様に符号化を行う。

使用する変数と終端記号を標準化する。

例えば、変数l個、終端記号m個ならば、

変数：$a_1, \cdots, a_l$　　(a_1は初期変数)

終端記号：$a_{l+1}, \cdots, a_{l+m}$　とする。

これらは $1110^{l}110^{m}111$ などで符号化できる。

変換規則は、$a_{i_1}a_{i_2}\cdots a_{i_p} \to a_{j_1}a_{j_2}\cdots a_{j_q}$と書けるので、

$$c = 110^{i_1}10^{i_2}1\cdots 10^{i_p}110^{j_1}10^{j_2}1\cdots 10^{j_q}11$$

というような形で書ける。結局、

$1110^{l}110^{m}111c_1 11c_2 11\cdots 11c_r 111$　などの符号化をすればよい。

2.非可解な問題 (17)

TMの停止問題をこの問題に還元する。
TMの停止問題の入力列(M, x)に対する写像$: t\ (M, x) \mapsto (G, y)$
を実現するチューリング変換機Tを以下のように構成する:

① $E(M)$を書き変えて、Mが入力列に対して停止するならば、必ずその列を受理するようなTM M'の符号化$E(M')$を得る。
② 定理5.3で説明した方法を使って、TM M'を模倣する0型文法をGとする。
③ $y = x$として、Gとxを符号化したものを出力する。

Mがxに対して停止 $\Leftrightarrow$ M'はxを受理　　(①より)
$\Leftrightarrow$ Gはxを生成可能　(②より)

また定理5.3の証明より、$TM\ T$は必ず停止するようにできることが分かる。
よって、この問題はTMの停止問題に還元することができたので、非可解である。

2.非可解な問題 (20)

例題6.3 文脈自由文法Gが列xを生成するかどうかの決定問題は可解である。

(証明の概要)
Gをチョムスキー標準形にする。

チョムスキー標準形の生成規則は下記のどちらかである:
- 右辺が1個の終端記号 （非膨張規則）
- 右辺が2個の変数 (膨張規則)

Gの導出　$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow y$
で、膨張規則をk回使用すると、yの長さは$k+1$になる。

2.非可解な問題 (21)

逆にGで長さnの列yが生成されたなら、$n-1$回膨張規則が使われなければならない。

また、長さnの列yが全て終端記号のみであるとすると、
導出では、非膨張規則をn回使わないといけない。

結局、長さn の列xがGで導出可能か調べるには、n回の膨張規則と$n-1$回の非膨張規則を総当たりで調べればよい。
(具体的なやり方は省略)

(0型文法の場合は、導出中に列の長さが減ることがあるので、有限の時間で総当たりすることができない。)

3.ポストの対応問題 (1)

できるだけ簡単な構造を持った非可解問題があれば、非可解性の証明のための還元の作業が楽になる。

ポストの対応問題:

入力: 列の対の集合 $\{(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)\}$

$x_i, y_i \in \Gamma^*$

問題: $x_{i_1} x_{i_2} \cdots x_{i_n} = y_{i_1} y_{i_2} \cdots y_{i_n}$

を満たす整数列$i_1 i_2 \cdots i_n$が存在するか？

ただし、$i_l \in \{1, \cdots, k\}$とする。

同じ列の対、例えば(x_2, y_2)、は何度使ってもよい。

3.ポストの対応問題 (2)

例えば、入力列を

$$\{(1,111),(10111,10),(10,0)\}$$

とする。
つまり、

$$x_1 = 1, x_2 = 10111, x_3 = 10,$$
$$y_1 = 111, y_2 = 10, y_3 = 0$$

である。

この入力に対する答えはyesである。
実際、2113という整数列に対して、

$$x_2 x_1 x_1 x_3 = 101111110$$
$$y_2 y_1 y_1 y_3 = 101111110$$

となる。

3.ポストの対応問題 (3)

例題 6.4 以下の入力列に対する、ポストの対応問題の答えがnoであることを証明せよ:

$$\left\{\begin{pmatrix}10\\101\end{pmatrix},\begin{pmatrix}011\\11\end{pmatrix},\begin{pmatrix}101\\011\end{pmatrix}\right\}.$$

(解答)
答えをyesとするような整数列Xが存在したとする。するとXは2または3で始まることはあり得ない。実際、x_2とy_2、x_3とy_3の最初の記号は異なっている。

Xは1で始まる。すると、x列とy列は、

$$\begin{matrix}10\\101\end{matrix}$$

となり、x列の1が1個足りないので、次のx列の記号は1でないといけない。よって、x_1かx_3が次に来る。

3.ポストの対応問題 (3)

$$\left\{\begin{pmatrix}10\\101\end{pmatrix},\begin{pmatrix}011\\11\end{pmatrix},\begin{pmatrix}101\\011\end{pmatrix}\right\}.$$

x_1の場合、列は、

$$1010$$
$$101101$$

となり、失敗である。
x_3の場合、

$$10101$$
$$101011$$

となる。しかし、x列が1が1個足りない状況になっている。
これは以前の状況のくり返しであり、この状況を解消することは永遠にできない。
よって、この問題の答えはnoである。

3.ポストの対応問題 (4)

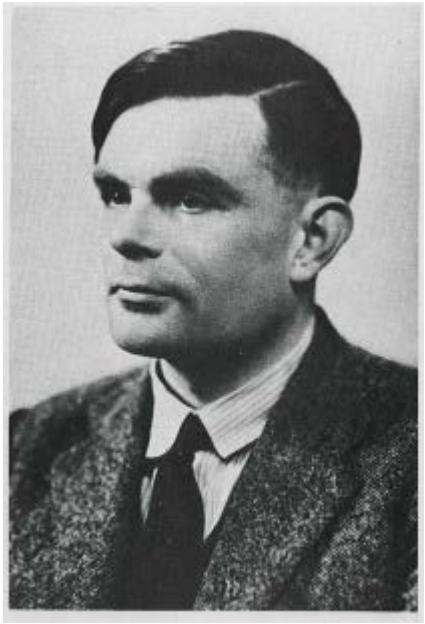
定理 6.4 ポストの対応問題は非可解である。

(証明のアイデア)
『0型文法Gが列xを生成するかどうかの問題』を
『0型文法が列0を生成するかどうか』の問題に還元する。

さらに、『0型文法が列0を生成するかどうか』の問題をポストの対応問題に還元する。

証明は複雑なので省略(参考書には書いてある。)

補遺．チューリングとチューリング機械



アラン・マシスン・チューリング
（Alan Mathison Turing)

- 1912年6月23日生まれ
- 1931: ケンブリッジ大学入学、数学を専攻。
- 1936: 論文『計算可能数、ならびに
  そのヒルベルトの決定問題への応用』にて、
  (万能)チューリング機械の理論を確立
- 1939-1945：ドイツの暗号エニグマの解読の自動化に成功
- 1945-1947: 英国初の万能計算機ACEの設計を行う。
- 1950: 論文『計算する機械と知性』
  人口知能の問題を提起、『チューリングテスト』の提案
- 1954年6月8日、死亡(自殺)

補遺．チューリングとチューリング機械

『計算可能数、ならびにそのヒルベルトの決定問題への応用』

論文の主題は、『一階述語論理において与えられた、論理式が証明可能かどうかを(有限の方法で)決定することはできるか？』という問題(一階述語論理の決定問題)であった。

そもそも、『(有限の方法で)決定する=計算可能である＝アルゴリズムが存在する』ということを数学的に定義されていなかった。そのため、『計算可能性』の数学的な定義が必要となり、そのためにチューリング機械を導入した。

万能チューリング機械を導入し、チューリング機械の停止問題が非可解性を証明し、それを用いて一階述語論理の決定問題も非可解であることを証明した。

補遺．チューリングとチューリング機械

実は、一階述語論理の決定問題はチューリングよりも少し早く、
アロゾン＝チャーチ(1903-1995)が解いていた。
“An unsolvable problem in elementary number theory(初等数論における非可解な問題について)” (1936)

この論文の中で、チューリング機械と後に同等であると証明される計算モデルである『ラムダ計算』を創案している。
ラムダ計算を用いて、一階述語論理の決定問題の非可解性を証明した。

関数型言語(Lispなど)はラムダ計算の実装となっている。

第七章：計算複雑性理論

計算量クラスNPを中心に計算量の理論について学ぶ。

0. 時間計算量と計算困難な問題
1. クラスPとクラスNP
2. NP完全性
3. その他の計算量クラス

O. 時間計算量と計算困難な問題 (1)

可解な問題は、原理的には計算機で解くことができる。
しかし、全ての可解な問題が計算機で容易に解けるとは限らない。

定義
$TM\ M$の(最悪)時間計算量が$t(n)$であるとは、任意の入力列wに対して、Mが$t(|w|)$ステップ以下で停止することをいう。
このようなTMを$t(n)$時間限定TMと呼ぶ。

$|w|$: 入力列wの長さ

この講義では、平均時間計算量は考えない。

0. 時間計算量と計算困難な問題 (2)

定義より、

$\log(n)$時間限定TMで解ける問題は、
n時間限定TMで解ける問題より計算が容易

n時間限定TMで解ける問題は、
n^2時間限定TMで解ける問題より計算が容易

n^2時間限定TMで解ける問題は、
2^n時間限定TMで解ける問題より計算が容易

相対的な計算の困難さ（容易さ）の比較はできる。

O. 時間計算量と計算困難な問題 (3)

原理的には計算可能(可解)だが、“絶対的に”計算が困難な問題:

決定問題Dを解くTM(≃アルゴリズム)の計算量が2^nだとする。

ある計算機は、このアルゴリズムを使ってlビットの入力に対してDを1秒で解いたとする。

入力ビット数	解くのにかかる時間
l	1秒
$l+10$	約20分
$l+20$	約10日
$l+27$	約3年
$l+33$	約300年
$l+60$	約300億年

数十ビット程度の長さで、
現実的には解けなくなる。

太陽の寿命は50億年
宇宙の年齢は150億年

O. 時間計算量と計算困難な問題 (4)

スーパーコンピュータを使うとどうか？

先ほどの計算機を、100万個の並列に繋いだスーパーコンピュータの場合（京は70万コア）、
アルゴリズムの完全な並列化が可能だとしても:

100万=$1 \times 10^6 \approx 1 \times 2^{20}$ より

入力ビット数	解くのにかかる時間
l	10^{-6}秒
$l+20$	約1秒
$l+40$	約10日
$l+47$	約3年
$l+53$	約300年
$l+80$	約300億年

太陽の寿命は50億年
宇宙の年齢は150億年

スーパーコンピュータを使っても、計算量が2^nのアルゴリズムは、大きな入力に対する計算ができない。

1. クラスPとクラスNP (1)

指数関数(2^n)の時間計算量のアルゴリズムは、大きな入力に対しては計算が不可能になる。

そうでないアルゴリズムは計算が容易であると言えるかもしれない。

多項式時間限定TM：ある多項式$p(n)$に対して$p(n)$時間限定TM

定義(計算量クラスP)： 決定問題(言語)Qに対して、多項式時間限定TMでQを解くことができるとき、『QはクラスPに属す』という。

n時間限定TMやn^2時間限定TMで解ける問題は、計算容易であると言ってよいと思われる。

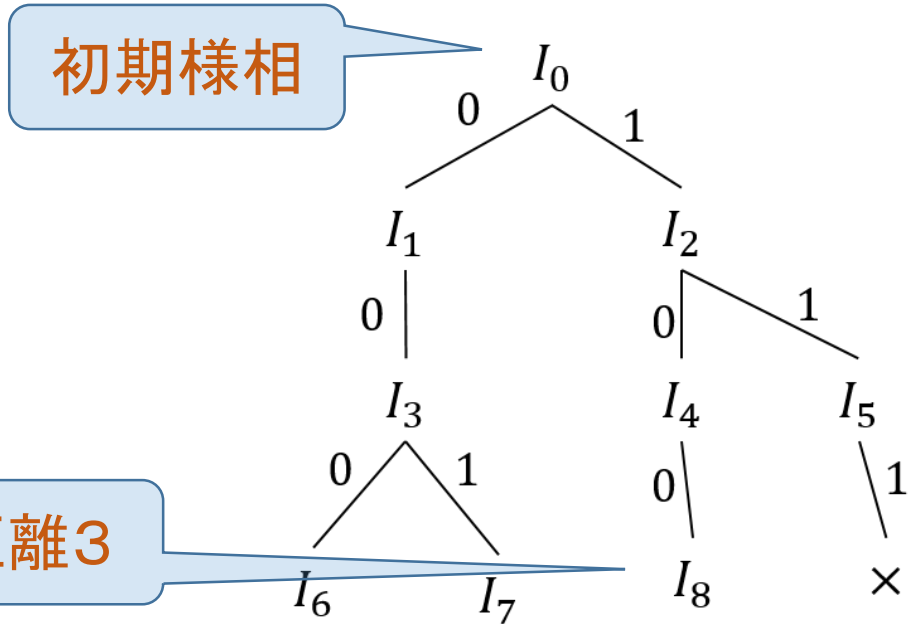
しかし、n^{100}時間限定TMでしか解けない問題は、計算容易だろうか？

1．クラスPとクラスNP　(4)

クラスPよりも『少しだけ難しい』決定問題のクラスとしてNPを考える。そのために$t(n)$時間限定非決定性TMを定義する。

定義：　任意の入力列wに対して、wが受理されるならば、初期様相から距離$t(|w|)$以内に受理様相が存在する非決定チューリング機械を$t(n)$時間限定非決定性TMと呼ぶ。

多項式時間限定非決定TM
＝ある多項式$p(n)$に対して
　　$p(n)$時間限定非決定性TM



1. クラスPとクラスNP (5)

定義(計算量クラスNP): 決定問題(言語)Qに対して、多項式時間限定非決定性TMでQを解くことができるとき、『QはクラスNPに属す』という。

定義より$P \subset NP$

NPは検算が容易な問題からなる計算量クラスということができる。

定理7.0
与えられた言語Q がNPに属するとき、又その時に限り、
以下の性質を持つPに属する言語Lと多項式qが存在する:
① 列xがQに属するならば、ある列wが存在して、
$|w| \leq q(|x|)$かつ$(x, w) \in L$
② 列xがQに属さないならば、任意の列wに対して$(x, w) \notin L$

1．クラスPとクラスNP　(6)

定理を言い換えると、
言語QがNPに属する必要十分条件は:

入力列xがQに属するならば、その証拠w (多項式の長さの列) が存在して、xとwを入力として与えるとyesを返すアルゴリズム(多項式時間TM)が存在。

入力列xがQに属さないならば、任意の証拠wに対してxとwをそのアルゴリズム(TM)に入力するとNoと答える。

つまり、入力列xがQに入っていることを効率よくチェックできる証拠が存在する。

xとwを入力してYESが出れば、必ず$x \in Q$

1. クラスPとクラスNP (7)

定理の証明で、に関する以下の補題を使う:

$t(n)$時間限定k-テープTM: $t(n)$時間以内で必ず止まるk本のテープを持つTM

補題:任意の言語Lに対して、Lを認識する$t(n)$時間限定k-テープTMが存在するならば、Lを 認識する $t(n)^2$ 時間限定TMが存在する。

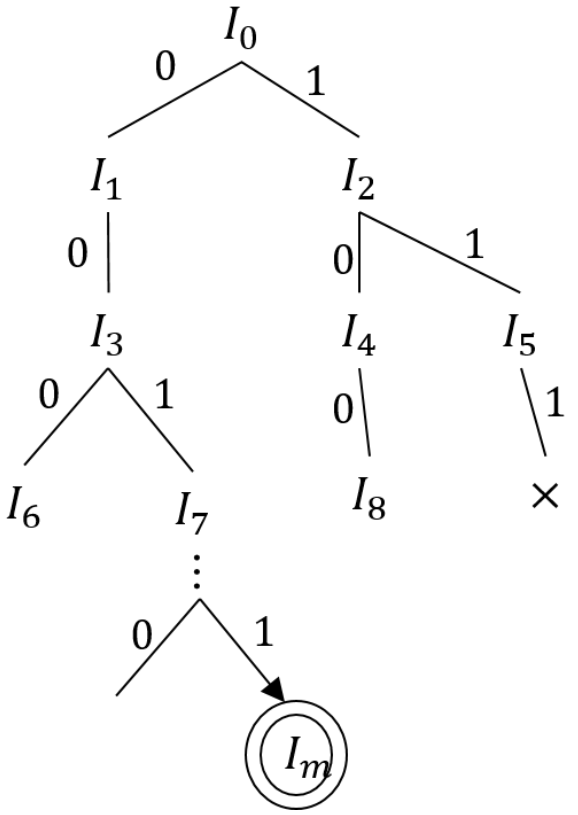
つまり、クラスPに関しては、何本のテープのTMで考えても同じ。

1. クラスPとクラスNP (8)

(定理7.0の証明)
まず、与えられた言語Q がNPに属することを仮定して、条件を満たすクラスPに属する言語が存在することを証明する。

Qを認識する非決定性TMの非決定的選択の上限をkとする。
すると距離cの様相は、
$\{1, \cdots, k\}$上の列$i_1 i_2 \cdots i_c$で指定できる。
つまり、様相を$I(i_1 i_2 \cdots i_c)$と書ける。

例えば、$k = 2$の場合、
左図のように$I_1 = I(0), I_5 = I(11),$
$I_8 = I(100)$
などと書くことができる。

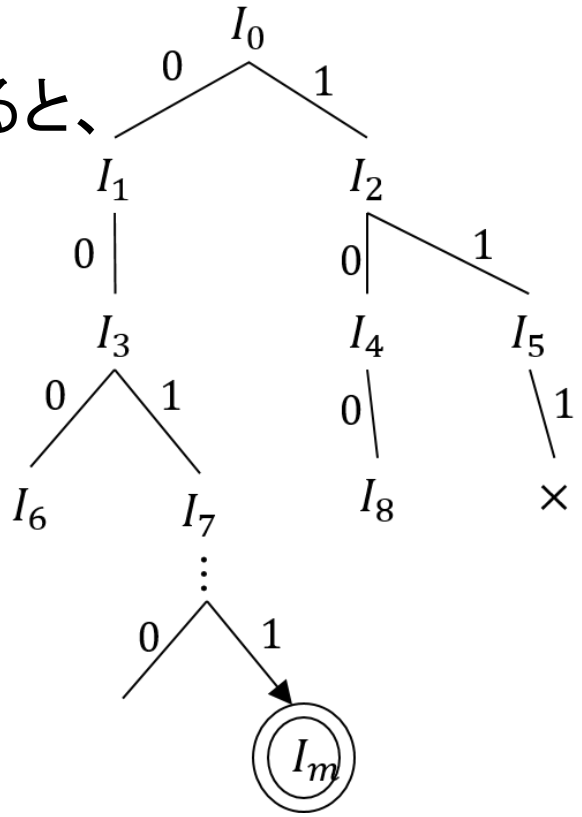


1. クラスPとクラスNP (9)

2テープTM Mは、入力xと$i_1 i_2 \cdots i_c$が与えられると、
2テープ目に$i_1 i_2 \cdots i_c$を書き、
1テープ目をxのみにする。

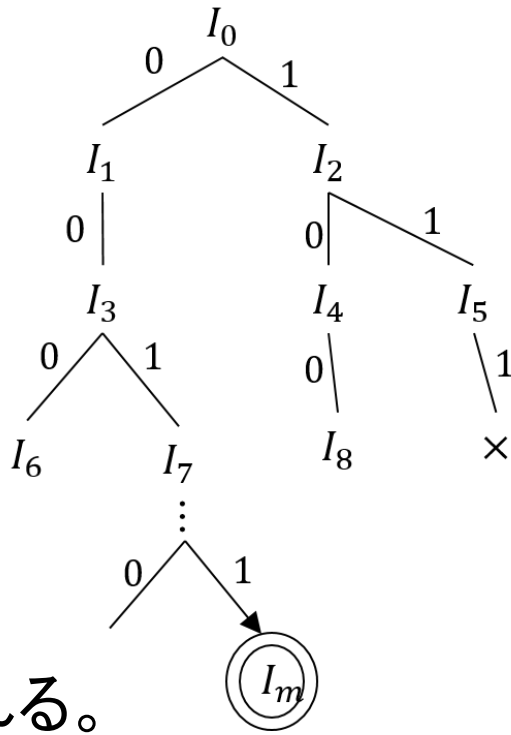
2番目のテープを読みながら、
lステップ目には、i_l番目の選択をするように、
非決定TMを1番目のテープで模倣していく。

Mは$i_1 i_2 \cdots i_c$を読み切ると停止するとしておく。



TM Mは過去の授業で非決定性TMを模倣するために、
使った決定性TMと2テープ目を書き変えないことを除いては同じ。

1. クラスPとクラスNP (10)



今、$x \in Q$とすると、ある多項式$p(n)$に対して、
$c \leq p(n)$かつ$I(i_1 i_2 \cdots i_c)$が受理様相である
列$i_1 i_2 \cdots i_c$が存在する。

よって、xと$i_1 i_2 \cdots i_c$をMに入力すると、
$c \leq p(n)$ステップ後に受理様相に到達し、受理される。

逆に、xがQに属さないとすると、任意の列$i_1 i_2 \cdots i_c$に対して、
$I(i_1 i_2 \cdots i_c)$は受理様相ではない。

よって、ある多項式$p(n)$に対して$c \leq p(n)$であるとき、
入力列xと$i_1 i_2 \cdots i_c$に対して、cステップ後に、TM Mは受理様相には到達しせずに停止する。

1. クラスPとクラスNP (11)

逆に、以下の条件を満たす言語Lと多項式qが存在するとする:
① 列xがQに属するならば、ある列wが存在して、
$|w| \leq q(|x|)$かつ$(x, w) \in L$
② 列xがQに属さないならば、任意の列wに対して$(x, w) \notin L$

この言語Lを認識する多項式時間TMをMとする。

以下のように非決定性TM Nを定義する:
入力列xが与えられると、Nは$q(|x|)$以下の長さの列wを非決定的に生成する。
次にNは(x, w)に対するLを認識するTM Mと同じ動作を行う。
Mが受理状態に達すると、Nも受理状態に遷移する。

1. クラスPとクラスNP (12)

$x \in Q$のときは、①の条件を満たすwを非決定的に選択すれば、非決定性TM Nは受理様相に到達する。この受理様相の初期様相からの距離は、Mが停止するステップ数程度なので多項式以下である。

$x \notin Q$のときは、②よりどのような列wに対しても、TM Mは(x, w)を受理することがない。よって、非決定性TM Nはどのような非決定的な選択を行っても受理様相には到達しない。

以上により、Qは多項式時間限定非決定性TMで認識できるので、NPに属している。

1. クラスPとクラスNP (13)

クラスNPに入る決定問題の例を色々見ていく:

CNF論理式の充足可能性問題(CNF-SAT)
入力:乗法標準形(Conjunctive Normal Form)論理式
出力:充足可能であればyes (1)、充足不可能ならno (0)

乗法標準形の論理式 $f(x_1, \cdots, x_n)$
例えば、 ($\neg x$ を $\bar{x}$ と書く)

$$f(x_1, x_2, x_3, x_4)$$
$$= (\bar{x}_1 \vee x_3 \vee \bar{x}_4) \wedge (x_2) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3 \vee \bar{x}_4)$$

$x_1 = 0\ (no), x_2 = x_3 = x_4 = 1\ (yes)$を割りてると、

$$\bar{x}_1 \vee x_3 \vee \bar{x}_4 = x_2 = x_1 \vee x_4 = x_2 \vee x_3 \vee \bar{x}_4 = 1$$

と、すべての節が1になるので、式全体が1になり、答えはyes。

1. クラスPとクラスNP (14)

例題7.1 CNF-SATがクラスNPに入ることを示せ。

解答:
各変数を$x0, x1, x01, x10, \cdots$とビット列で番号付けすることにする。
入力列は、

$$\left(\overline{x\vec{l}_1} \vee x\vec{l}_2 \vee \cdots\right) \wedge \cdots \wedge \left(x\vec{l}_c \vee \overline{x\vec{l}_{c+1}} \vee \cdots\right)$$

のような形をしている。

入力列の長さをnとすると、論理式に含まれる変数の数kは$k \leq n$を満たす。$x\vec{l}_1, \cdots, x\vec{l}_k$が論理式に含まれるとする。

長さkのビット列$w = j_1 \cdots j_k$に対して、$x\vec{l}_1 = j_1, \cdots, x\vec{l}_k = j_k$という割り当てを考える。（ e.g. $x\vec{l}_1 = j_1$ =1 なら、$x\vec{l}_1$ はyes）

１．クラスPとクラスNP （15）

長さkのビット列$w = j_1 \cdots j_k$に対して、$x\vec{l}_1 = j_1, \cdots, x\vec{l}_k = j_k$という割り当てを考える。

もし、論理式が充足可能であるならば、
長さkのビット列$w = j_1 \cdots j_k$が存在して、$x\vec{l}_1 = j_1, \cdots, x\vec{l}_k = j_k$という割り当てで、論理式が真 (1)になるものが存在する。

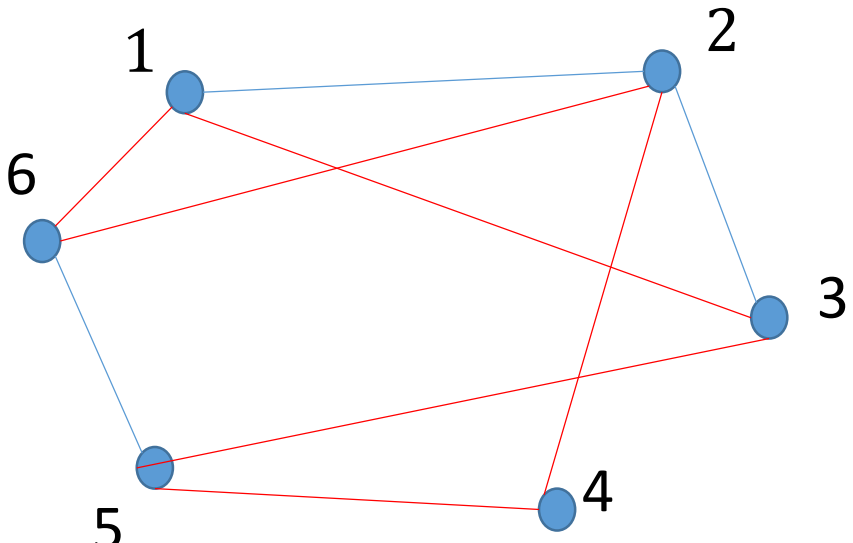
一方、論理式が充足可能でなければ、
任意の長さkのビット列$w = j_1 \cdots j_k$に対して、$x\vec{l}_1 = j_1, \cdots, x\vec{l}_k = j_k$という割り当では、論理式は真 (1)にならない。

wが与えられたときの論理式の値の計算は、すべての節に1となるリテラルが含まれていることをチェックするだけなので、明らかに多項式時間で終わる。よって定理7.0よりCNF-SATはNP

1. クラスPとクラスNP (16)

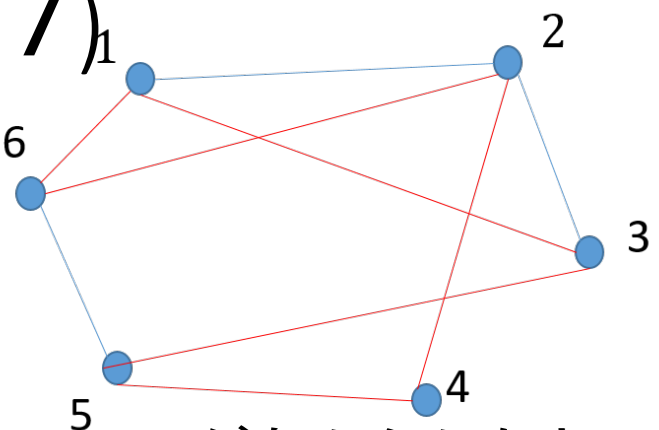
グラフのハミルトン閉路問題
入力: グラフ$G = (V, E)$
出力: Gにハミルトン閉路が含まれればyes、
そうでないならno.

ハミルトン閉路:グラフの全ての頂点をちょうど一回通る閉路



左図の場合、
閉路1-3-5-4-2-6-1は
ハミルトン閉路である。

1. クラスPとクラスNP (17)



グラフのハミルトン閉路問題はNPに属する。

証明:
長さnの入力 $(\{1,\cdots,k\},\{(v_1,v_2),\cdots,(v_{2h-1},v_{2h})\})$ が与えらえたとする。
$k \leq n$なので、各頂点vは、およそ $\log_2 k$の長さのビット列$E(v)$で指定できる。

グラフとビット列wを入力とするTM Mを以下のように定義する。
① Mは、まず$w = E(u_1)\cdots E(u_k)$となるk個の頂点からなる列 $u_1\cdots u_k$が存在するかどうかを確かめる。もし、対応する頂点が存在しなければ、入力を拒否する。

1. クラスPとクラスNP (17)

② そのような頂点列が存在すれば、次に$(u_1, \cdots, u_k, u_1)$ が閉路となっているかをチェックする。このためには、$1 \leq \forall b \leq k$に対して(u_b, u_{b+1})と一致する辺が入力グラフに含まれているかをチェックする。ただし、$u_{k+1} = u_1$ とする。閉路でなければ、入力を拒否する。

③ 最後に、閉路 $(u_1, \cdots, u_k, u_1)$がハミルトン閉路であるかチェックする。これは、頂点の列$\{u_1, \cdots, u_k\}$をソートして$\{1, \cdots, k\}$になるかをチェックするとよい。

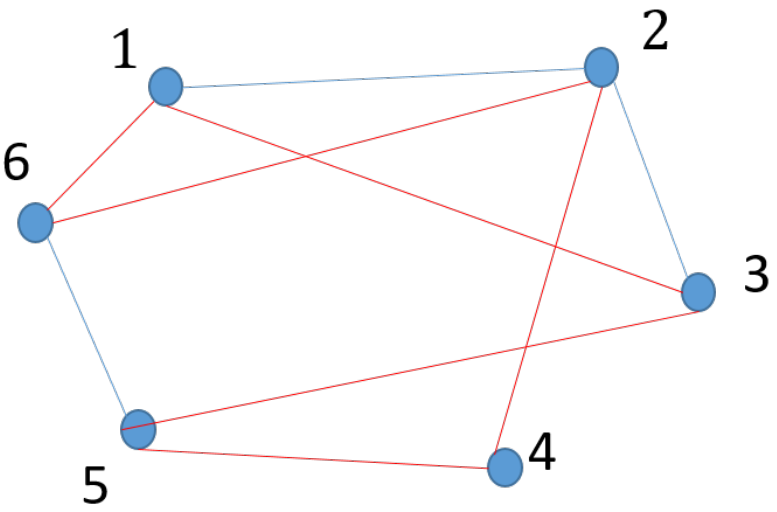
①②③は全て多項式時間で終えることができる。

1. クラスPとクラスNP (18)

以上の方法で、
もし入力グラフにハミルトン閉路$(u_1, \cdots, u_k, u_1)$が含まれる場合は、グラフとビット列$w = u_1 \cdots u_k$をTM Mに与えると、多項式時間でこれらを受理する。

逆に入力グラフにハミルトン閉路が含まれない場合は、どのようなビット列wに対しても、上記のTM Mは多項式時間で入力グラフとwを拒否をする。

よって、ハミルトン閉路問題はNPである。

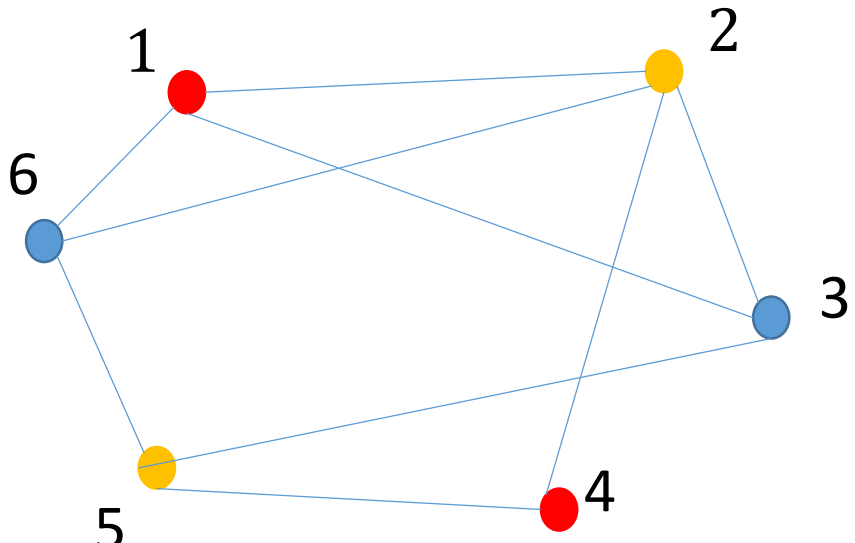


1．クラスPとクラスNP （19）

グラフのk彩色問題

入力　グラフ$G = (V, E)$

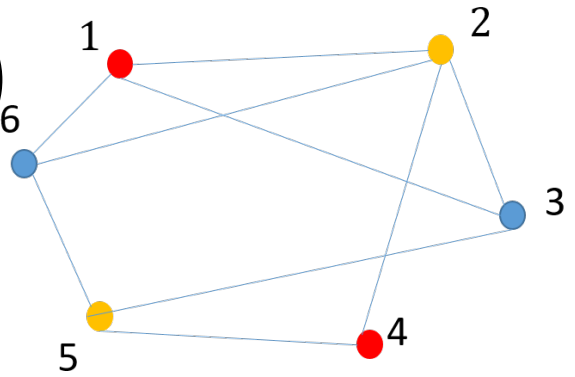
出力　Gの全ての頂点が、
辺で結ばれた二つの頂点を同じ色で塗ることなく、
k色で塗り分けることができればyes,
そうでないならno



左図の場合、3色で塗り分け可能
よってグラフの3彩色問題に対する
出力はyesである。

☆kは入力によらない定数である。

1. クラスPとクラスNP (20)$_6$

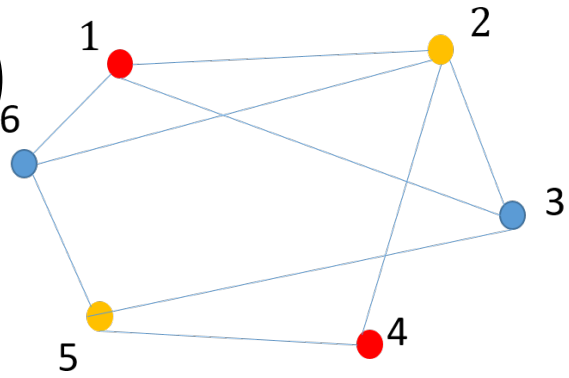


グラフのk彩色問題はNPに属する
証明:
長さnの入力 $(\{1,\cdots,l\},\{(v_1,v_2),\cdots,(v_{2h-1},v_{2h})\})$ が与えられたとする。
ここで、$v_i \in \{1,\cdots,l\}$とする。(頂点を$\{1,\cdots,l\}$で符号化している)

$\{1,\cdots,k\}$上の長さlの列$i_1\cdots i_l$に対して、 $(i_c \in \{1,\cdots,k\})$
頂点cにi_c番目の色を割り当てることを考える。

このために、
$i_1\cdots i_l$ を用いて関数gを$g(c)=i_c$で定義する。

1．クラスPとクラスNP (21)



入力グラフがk彩色可能であるならば、
ある列$i_1 \cdots i_l$が存在して、
任意のjに対して、$g(v_{2j-1}) \neq g(v_{2j})$が成立する。

これは、$i_1 \cdots i_l$が与えられれば多項式時間で確認できる。

入力グラフがk彩色可能でないならば、
任意の列$i_1 \cdots i_l$に対して、
あるjが存在して$g(v_{2j-1}) = g(v_{2j})$が成立する。

これも$i_1 \cdots i_l$が与えられれば多項式時間で確認できる。
よって定理7.0よりこの問題はNPである。

3. NP完全問題 (1)

NP問題には応用上重要な問題が多数存在する。

SATやハミルトン閉路問題は本当に多項式時間TMで解くことはできないのか？

もしNP問題が全て多項式時間TMで解けるなら、

$$P = NP$$

となる。
もし、多項式時間TMで解けないNP問題が存在すれば、

$$P \neq NP$$

である。

$P = NP$か$P \neq NP$かは、まだ未解決である。
ミレニアム懸賞問題：解くと100万ドルがもらえる。

3. NP完全問題 (2)

$P = NP$を証明するのに、すべてのNP問題が多項式時間TMで解けるかを調べるのは難しい。

最も難しいNP問題から研究すればよい。

NP完全問題：その問題が多項式時間TMで解ければ、すべてのNP問題が多項式時間TMで解ける($P = NP$)ような問題。

多くの学者は$P \neq NP$だと考えている。
よって、NP完全問題は、多項式時間TMを持たないと考えられる。

3. NP完全問題 (3)

定義
『Q_1はQ_2に多項式時間で還元できる』
$\Leftrightarrow$ Q_1への入力列qに対してQ_2への入力列を対応させる写像
$t: q_1 \mapsto q_2$で下記の性質を満たすものが存在する:

① $\forall q_1\ (Q_1(q_1) = yes \Leftrightarrow Q_2(t(q_1)) = yes)$
　q_1に対するQ_1の答えがyesなら、対応するQ_2の答えもyes
② q_1を入力として与えられたときに、
　$t(q_1)$を出力する多項式時間決定性チューリング変換機 Tが存在する。

このとき、$Q_2 \in P$なら、$Q_1 \in P$である。

実際、Q_2を解く多項式時間TMを Mとすると、TとMを連結させたTMはQ_1を多項式時間で解くことができる

3. NP完全問題 (4)

定義
NP問題Qは、すべてのNP問題がQに多項式時間で還元可能であるときに、NP完全であるという。

よって、あるQがNP完全であるならば、
$Q \in P$ならば、$P = NP$である。

定理
CNF-SATは、NP完全である。

歴史的には、CNF(乗法標準形)の制限のない命題論理式の充足可能性問題(SAT)が、最初にNP完全であることが証明された。
(Cook-Levinの定理)

3. NP完全問題 (5)

（CNF-SATがNP完全である証明のあらすじ）

任意のNP問題Qを一つ固定する。

Qへの入力列xをCNF fに変換するチューリング変換機で、

$$Q(x) = yes \Leftrightarrow f\text{は充足可能}$$

とあるものを作ればよい。
(☆チューリング変換機は、存在を示せるだけでよい。)

Qを解く非決定多項式時間TM をMとする。

まず、入力列xに対するMの動作を模倣する0型文法$G(M)$を構成して、更に、$G(M)$の動作を模倣するCNF fを作成する。

3. NP完全問題 (6)

CNF-SATはNP完全であることが、証明できたとする。
その他のNP問題がNP完全であることを証明したいなら、
CNF-SATが、その問題に多項式時間で還元可能であることを示せばよい。

この方法で、
- グラフのハミルトン閉路問題
- $k \geq 3$に対するグラフのk彩色問題

もNP完全であることが証明できる。

実は、$P \neq NP$が正しいと仮定して、
Pに入っていないと思われているNP問題の多くは、NP完全であることが証明されている。

3. NP完全問題 (7)

Pに入ることもNP完全であることも証明されていない問題
＝PとNP完全の間くらいのむつかしさの問題

素因数分解問題
入力:(x, y)
xはある素数pとqに対して$x = p \cdot q$を満たす。

出力:
pかqのどちらかが与えられた自然数yよりも小さいかどうか？
つまり、$(p \leq y) \vee (q \leq y)$の値(真偽値)。

素因数分解問題はRSA暗号という公開鍵暗号で使われる。

PでもNP完全でもない問題には、公開鍵暗号に使用可能なものが多い。

4. その他の計算量クラス(1)

言語：$Q \subset \{0,1\}^*$ に対する補集合Q^cを定義：

$$Q^c := \{0,1\}^* - Q$$

決定問題と対応する言語Qの関係：

決定問題：『$x \in \{0,1\}^*$は〇〇か？』

Yes: $x \in Q$ ⇔ No:$x \in Q^c$

Q^cに対応する決定問題を補問題という。

補問題：『$x \in \{0,1\}^*$は〇〇ではないか？』

Yes: $x \in Q^c$ ⇔ No:$x \in Q$

例）『xは素数か？』の補問題は『xは合成数か？』

4. その他の計算量クラス(2)

PやNPのような言語の集合を一般に計算量クラスという。

計算量クラスCに対して、計算量クラス$co-C$を
Cに入る決定問題の補問題の集合として定義：

$$co-C := \{Q \subset \{0,1\}^* \mid Q^c \in C\}$$

これにより$co-P$や$co-NP$が定義される。

$$co-P := \{Q \subset \{0,1\}^* \mid Q^c \in P\}$$
$$co-NP := \{Q \subset \{0,1\}^* \mid Q^c \in NP\}$$

4. その他の計算量クラス(3)

$$co-C := \{Q \subset \{0,1\}^* \mid Q^c \in C\}$$

定理($co-C$の性質)

① $Q \in C \Leftrightarrow Q^c \in co-C$

② $co-(co-C)=C$

③ $C \subset D \Longrightarrow co-C \subset co-D$

④ $C \subset co-C \Longrightarrow C = co-C$. また $co-C \subset C \Longrightarrow C = co-C$

(①の証明)

$co-C$の定義より任意の言語Q'に対して、

$$Q'^c \in C \Leftrightarrow Q' \in co-C$$

$Q := Q'^c$と定義すると、$Q^c = Q'^{cc} = Q'$ なので、

$$Q \in C \Leftrightarrow Q^c \in co-C$$

■

4. その他の計算量クラス(4)

定理($co-C$の性質)

① $Q \in C \Leftrightarrow Q^c \in co-C$

② $co-(co-C)=C$

③ $C \subset D \Rightarrow co-C \subset co-D$

④ $C \subset co-C \Rightarrow C = co-C$. また $co-C \subset C \Rightarrow C = co-C$

(②の証明)

$$co-(co-C) = \{Q \subset \{0,1\}^* \mid Q^c \in co-C\}$$
$$= \{Q \subset \{0,1\}^* \mid Q \in C\} = C$$

■

(③の証明)

$$Q \in co-C \Rightarrow Q^c \in C \subset D$$
$$\Rightarrow Q \in co-D$$

■

4. その他の計算量クラス(5)

$C \subset co-C \Longrightarrow C = co-C$. また $co-C \subset C \Longrightarrow C = co-C$

(④の証明)

$$C \subset co-C$$
$$\Longrightarrow co-C \subset co-(co-C)$$
$$\Longrightarrow co-C \subset C$$

よって、$C \subset co-C$かつ$co-C \subset C$より、$C = co-C$

$$co-C \subset C \Longrightarrow co-(co-C) \subset co-C$$
$$\Longrightarrow C \subset co-C$$

よって、$C \subset co-C$かつ$co-C \subset C$より、$C = co-C$ ■

4. その他の計算量クラス(6)

$$co-P := \{Q \subset \{0,1\}^* \mid Q^c \in P\}$$

Pと$co-P$は一致する:

定理: $P = co-P$

(証明) $Q \in co-P$とすると$Q^c \in P$.
よって、Q^cを認識する多項式時間TM Mが存在。
つまり、$x \in \{0,1\}^*$をMに入力すると、多項式時間で止まり、
受理状態で止まれば$x \in Q^c$、非受理状態で止まれば$x \in Q$.

Mの受理状態と非受理状態を逆転させたTMをM^cとする。
M^cは明らかにQを受理する。
Mが多項式時間で止まるので、M^cも多項式時間で止まる。
よって、$co-P \subset P$ なので、$P = co-P$ ■

4. その他の計算量クラス(7)

$$Q \in co-NP \iff Q^c \in NP$$

$Q \in NP$の定義は以下であった:
$x \in Q \Longrightarrow$ 距離$c < poly(|x|)$の受理様相が存在

これから、$Q \in co-NP$の必要十分条件は以下になる:
$x \notin Q \Longrightarrow$ 距離$c < poly(|x|)$の拒否様相が存在

拒否様相:非受理状態で次の遷移が存在しない様相

$Q \in NP$の以下の必要十分条件を使う:

$\exists L \in P, \exists$多項式q
$s.t.$
$$\begin{cases} x \in Q \Longrightarrow \exists w \in \{0,1\}^*, s.t.\ |w| < q(|x|) \text{ かつ } (x,w) \in L \\ x \notin Q \Longrightarrow \forall w \in \{0,1\}^*, (x,w) \notin L \end{cases}$$

4. その他の計算量クラス(8)

$Q \in NP$の以下の必要十分条件を使う:

$\exists L \in P, \exists$多項式q

$s.t.$

$$\begin{cases} x \in Q \Longrightarrow \exists w \in \{0,1\}^*, s.t.\ |w| < q(|x|) \text{ かつ } (x,w) \in L \\ x \notin Q \Longrightarrow \forall w \in \{0,1\}^*, (x,w) \notin L \end{cases}$$

よって、

$Q \in co-NP \Leftrightarrow Q^c \in NP$を使うと、

$Q \in co-NP$の必要十分条件は以下:

$\exists L \in P, \exists$多項式q

$s.t.$

$$\begin{cases} x \notin Q \Longrightarrow \exists w \in \{0,1\}^*, s.t.\ |w| < q(|x|) \text{ かつ } (x,w) \in L \\ x \in Q \Longrightarrow \forall w \in \{0,1\}^*, (x,w) \notin L \end{cases}$$

4. その他の計算量クラス(9)

$Q \in co-NP$の必要十分条件は以下:

$\exists L \in P, \exists$多項式q

$s.t.$

$$\begin{cases} x \notin Q \Longrightarrow \exists w \in \{0,1\}^*, s.t.\ |w| < q(|x|) \text{ かつ } (x,w) \in L \\ x \in Q \Longrightarrow \forall w \in \{0,1\}^*, (x,w) \notin L \end{cases}$$

wは、xがQに入っていないことの証拠である。

$co-NP$は、
$x \notin Q$の検算が容易な問題Qからなる計算量クラス

NPは、
$x \in Q$の検算が容易な問題Qからなる計算量クラス

4. その他の計算量クラス(10)

クラスco $-$ NPに入る決定問題の例:

CNF論理式の充足可能性問題の補問題
入力:乗法標準形(Conjunctive Normal Form)論理式
出力:充足不可能ならyes (1)、充足可能であればno (0)

グラフのハミルトン閉路問題の補問題
入力: グラフ$G = (V, E)$
出力: Gにハミルトン閉路が含まれればno、
そうでないならyes.

NP問題のnoとyesを逆にすればよい。

4. その他の計算量クラス(11)

$$co - C \coloneqq \{L \subset \{0,1\}^* \mid L^c \in C\}$$

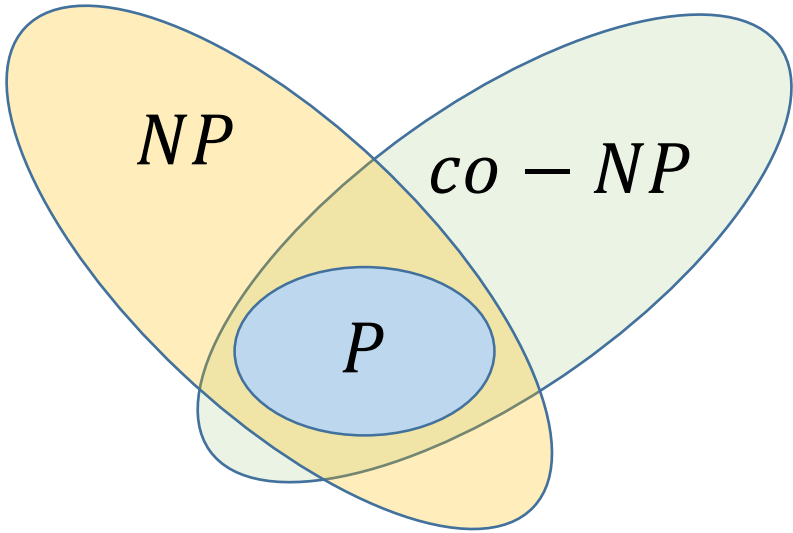
$$C \subset D \Longrightarrow co - C \subset co - D$$

を用いると、$P \subset NP$より、

$$P = co - P \subset co - NP$$

したがって、

$$P \subset NP \cap co - NP$$



4. その他の計算量クラス(12)

定理：$\mathrm{N}P \neq co-NP$ならば、$P \neq NP$

言い換えると：　$P = NP$ならば、$\mathrm{N}P = co-NP$

(証明)
$P = NP$とする。

$$
\begin{aligned}
& co-NP \\
&= \{L \in \{0,1\}^* \mid L^c \in NP\} \\
&= \{L \in \{0,1\}^* \mid L^c \in P\} \\
&= co-P = P = NP
\end{aligned}
$$

よって、$co-NP = NP$　■

4. その他の計算量クラス(13)

定理：N$P \neq co-NP$ならば、$P \neq NP$

言い換えると：　$P = NP$ならば、N$P = co-NP$

$co-NP$は、$P \neq NP$を証明するために考えだされた。

しかし、$NP \neq co-NP$を証明することは、
$P \neq NP$ を証明するのと同じか、それ以上に難しいことが分かった。

$P \neq NP$と予想する人が多いので、
同じように$NP \neq co-NP$と予想されている。

4. その他の計算量クラス(14)

定義
$co-NP$問題Qは、すべての$\mathrm{co}-NP$問題がQに多項式時間で還元可能であるときに、$\mathrm{co}-NP$完全であるという。

よって、あるQが$\mathrm{co}-NP$完全であるならば、
$Q \in P$ならば、$P = co-NP$である。

定理
NPに入る$co-NP$完全問題が存在すれば、$P = NP$

4. その他の計算量クラス(15)

定理
NPに入る$co-NP$完全問題が存在すれば、$P = NP$

(証明)
NPに入る$co-NP$完全問題をQ_2とすると、任意の$co-NP$問題Q_1がQ_2に多項式時間で還元可能。

よって、任意の$co-NP$問題Q_1に対して、
$t: q_1 \mapsto q_2$で下記の性質を満たすものが存在する:

① $\forall q_1 \, (q_1 \in Q_1 \Leftrightarrow t(q_1) \in Q_2)$
② q_1を入力として与えられたときに、
$t(q_1)$を出力する多項式時間決定性チューリング変換機 Tが存在。

4. その他の計算量クラス(16)

一方で、Q_2はNP問題でもあるので、以下の言語$L \in P$が存在:

① 列xがQ_2に属するならば、ある$poly(|x|)$以下の長さの列wが存在して、$(x, w) \in L$

② 列xがQ_2に属さないならば、任意の列wに対して$(x, w) \notin L$

チューリング変換機Tを使うことでQ_1に関して以下が言える:

① 列xがQ_1に属するならば、ある$poly(|x|)$以下の長さの列wが存在して、$(t(x), w) \in L$

② 列xがQ_1に属さないならば、任意の列wに対して$(t(x), w) \notin L$

4. その他の計算量クラス(17)

チューリング変換機Tを使うことでQ_1に関して以下が言える:

① 列xがQ_1に属するならば、ある$poly(|x|)$以下の長さの列wが存在して、$(t(x), w) \in L$

② 列xがQ_1に属さないならば、任意の列wに対して$(t(x), w) \notin L$

Lを認識するTMをMとする。
TとMを連結させたTM M'は多項式時間で止まり、
$(x, w) \rightarrow (t(x), w)$とした後、$(t(x), w) \in L$かどうかを判定する。

M'に対応する言語L'はPに属して以下をみたす:

① 列xがQ_1に属するならば、ある$poly(|x|)$以下の長さの列wが存在して、$(x, w) \in L'$

② 列xがQ_1に属さないならば、任意の列wに対して$(x, w) \notin L'$

4. その他の計算量クラス(18)

M'に対応する言語L'はPに属して以下をみたす:

① 列xがQ_1に属するならば、ある$poly(|x|)$以下の長さの列wが存在して、$(x, w) \in L'$

② 列xがQ_1に属さないならば、任意の列wに対して$(x, w) \notin L'$

よって、$Q_1 \in NP$である。

Q_1は任意の$co-NP$問題だったので、$co-NP \subset NP$

よって、$co-NP = NP$.

つまり、$P = NP$. ■

4. その他の計算量クラス(19)

定理
NP完全問題の補問題は、$co-NP$完全問題である。

（証明）
Q_2 はNP完全問題の補問題とする。
すると、$Q_2 \in co-NP$.

Q_1を任意の$co-NP$問題とすると、$Q_1^c \in NP$.
Q_2^cがNP完全問題であることより:

$$q \in Q_1^c \Leftrightarrow t(q) \in Q_2^c$$

となる $t: Q_1^c \to Q_2^c$ を実行する多項式時間で止まるチューリング変換機Tが存在。

4. その他の計算量クラス(20)

Q_1を任意の$co-NP$問題とすると、$Q_1^c \in NP$.
Q_2^cがNP完全問題であることより:

$$q \in Q_1^c \Leftrightarrow t(q) \in Q_2^c$$

となる $t: Q_1^c \to Q_2^c$ を実行する多項式時間で止まるチューリング変換機Tが存在。

$$q \in Q_1 \Leftrightarrow q \notin Q_1^c \Leftrightarrow t(q) \notin Q_2^c \Leftrightarrow t(q) \in Q_2$$

よって、Tは、

$$q \in Q_1 \Leftrightarrow t(q) \in Q_2$$

を満たすtを実行する多項式時間で止まるチューリング変換機。

よって、任意の$co-NP$問題 Q_1はQ_2に多項式時間で還元可能なので、Q_2は$co-NP$完全問題. ■

4. その他の計算量クラス(21)

co $-$ NP完全問題の例:

CNF論理式の充足可能性問題の補問題
入力:乗法標準形(Conjunctive Normal Form)論理式
出力:充足不可能ならyes (1)、充足可能であればno (0)

グラフのハミルトン閉路問題の補問題
入力: グラフ$G = (V, E)$
出力: Gにハミルトン閉路が含まれればno、
そうでないならyes.

NP完全問題のnoとyesを逆にすればよい。

4. その他の計算量クラス(22)

co − NP完全問題がNPに入るかどうかは不明 (多分入らない)

一方で、
素因数分解問題は $NP \cap co - NP$に入ることが分かっている。

素因数分解問題
入力:(x, y)
xはある素数pとqに対して$x = p \cdot q$を満たす。
出力:
pかqのどちらかが与えられた自然数yよりも小さいならyes

一般に、公開鍵暗号に役立つのは、$NP \cap co - NP$に入る問題。
NP完全問題や$co - NP$完全問題は公開鍵暗号に使えない。

4. その他の計算量クラス(23)

NPよりも大きいクラスで、有名なものとしては$PSPACE$がある。

定義（計算量クラス$PSPACE$）
決定問題(言語)Qに対して、多項式空間限定TMでQを解くことができるとき、『Qはクラス$PSPACE$に属す』という。

多項式空間限定TM:
ある多項式qが存在して、任意の入力列xに対して、
TMが止まるまでに訪れるテープのマス目の数cが$c < q(|x|)$.

多項式時間では、多項式回しかヘッドを右に動かせないので、

$$P \subset PSPACE$$

4. その他の計算量クラス(24)

定理
$NP \subset PSPACE$ かつ $co-NP \subset PSPACE$

(証明のあらすじ)

まず、以前の話からテープを何本にしても、$PSPACE$の定義は変わらないことが分かる。

2テープ決定性TMで非決定性TMを模倣するとき、nステップの非決定性TMの模倣に、2テープ決定性TMはn以下の長さのテープがあれば充分であった。

よって、NPや$co-NP$に入る問題は、多項式の長さのテープの決定性TMで、解くことができる。 ■

4. その他の計算量クラス(25)

定理
$NP \subset PSPACE$ かつ $co - NP \subset PSPACE$