情報数学
第5回　チューリング機械と計算可能性

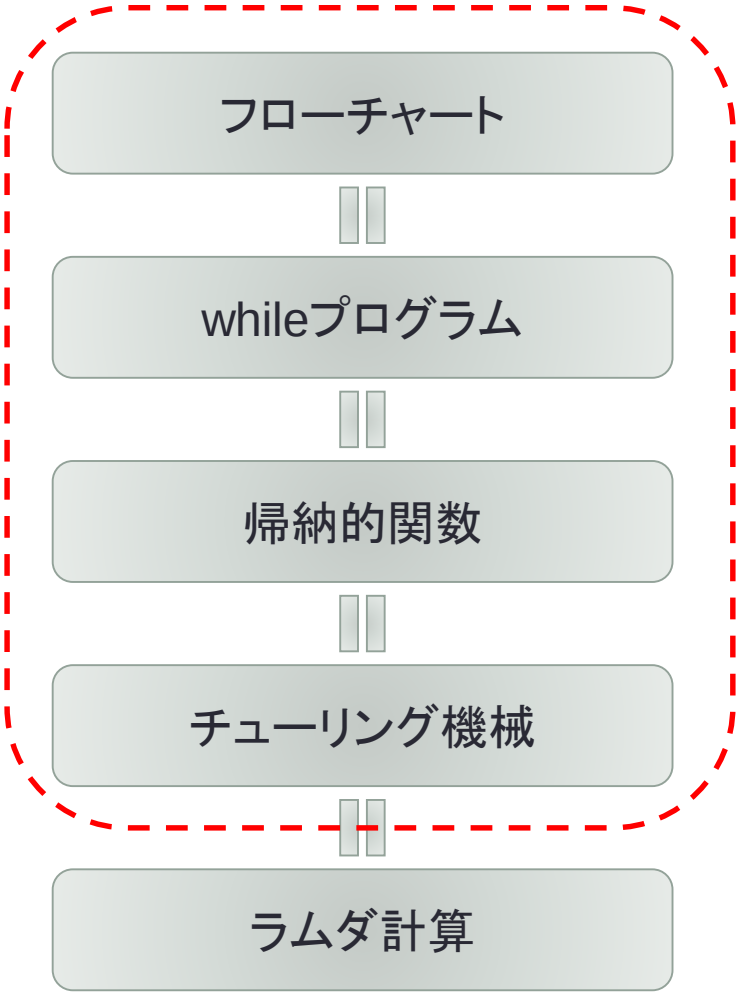
萩野　達也
hagino@sfc.keio.ac.jp

スライドURL

https://vu5.sfc.keio.ac.jp/slide/

これまで

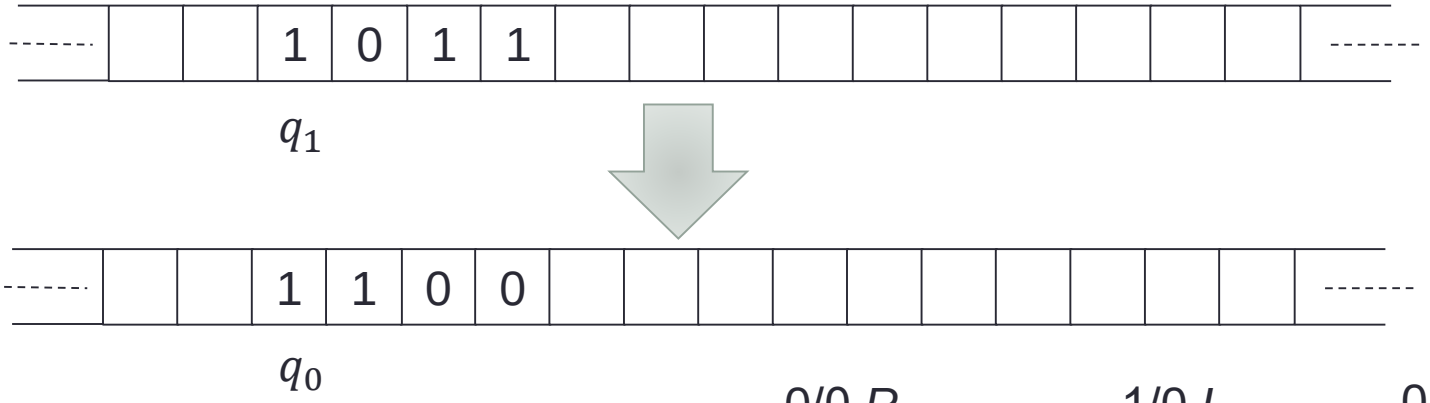
- 計算可能
  - フローチャート
  - whileプログラム
  - 帰納的関数
    - 原始帰納的関数
    - 最小解オペレータ
  - チューリング機械



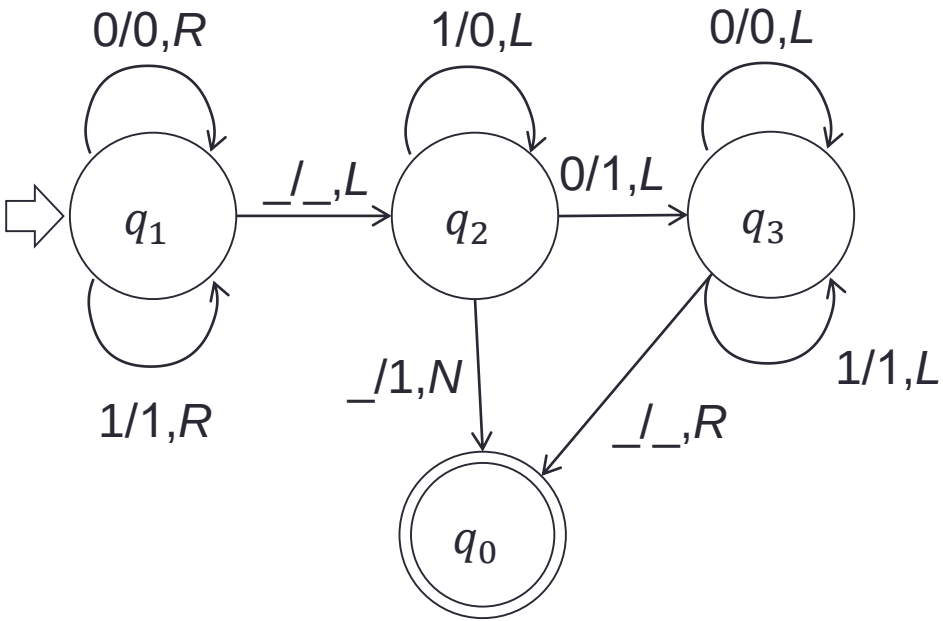
チューリング機械の例(3)

- テープ上の二進数の数字に1を加えるチューリング機械を作りなさい.

$$M_5 = (\{\_, 0,1\}, \{q_0, q_1, q_2, \cdots\}, T_5)$$



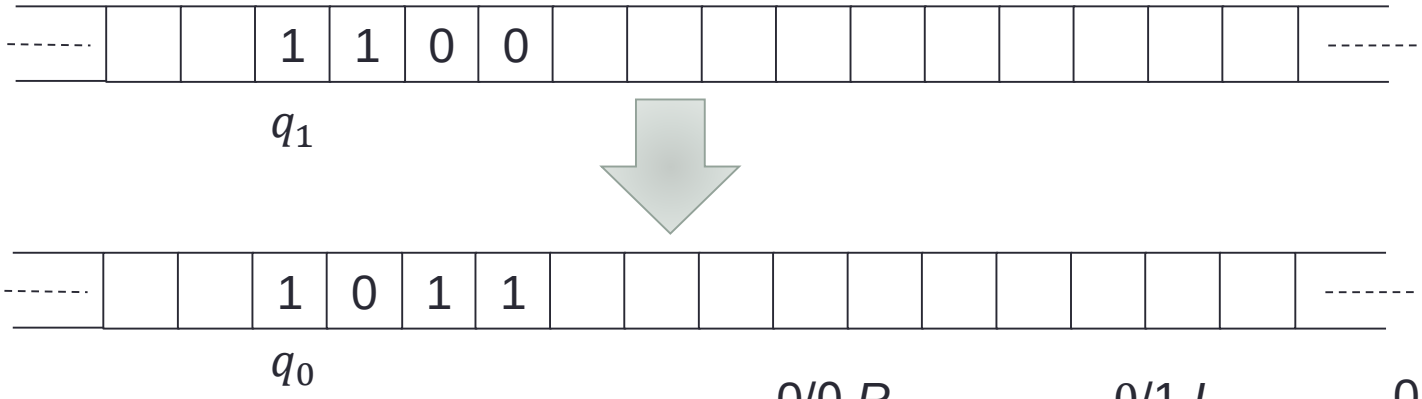
T_5	_	0	1
q_1	(q_2,_,L)	(q_1,0,R)	(q_1,1,R)
q_2	(q_0,1,N)	(q_3,1,L)	(q_2,0,L)
q_3	(q_0,_,R)	(q_3,0,L)	(q_3,1,L)



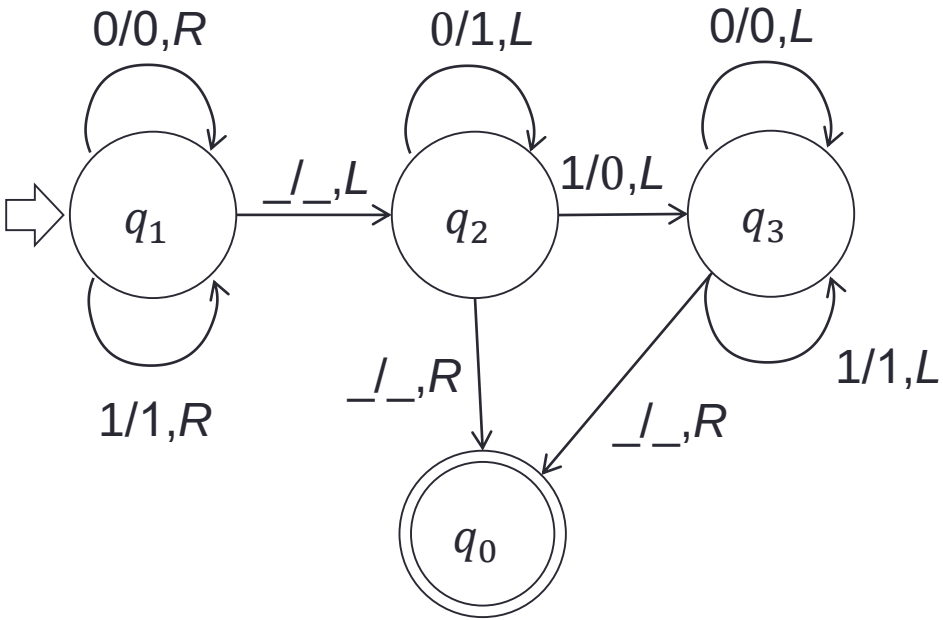
チューリング機械の例(4)

- テープ上の二進数の数字から1を引くチューリング機械を作りなさい.

$$M_6 = (\{\_, 0, 1\}, \{q_0, q_1, q_2, \cdots\}, T_6)$$



T_6	_	0	1
q_1	$(q_2,\_,L)$	$(q_1,0,R)$	$(q_1,1,R)$
q_2	$(q_0,1,N)$	$(q_2,1,L)$	$(q_3,0,L)$
q_3	$(q_0,\_,R)$	$(q_3,0,L)$	$(q_3,1,L)$

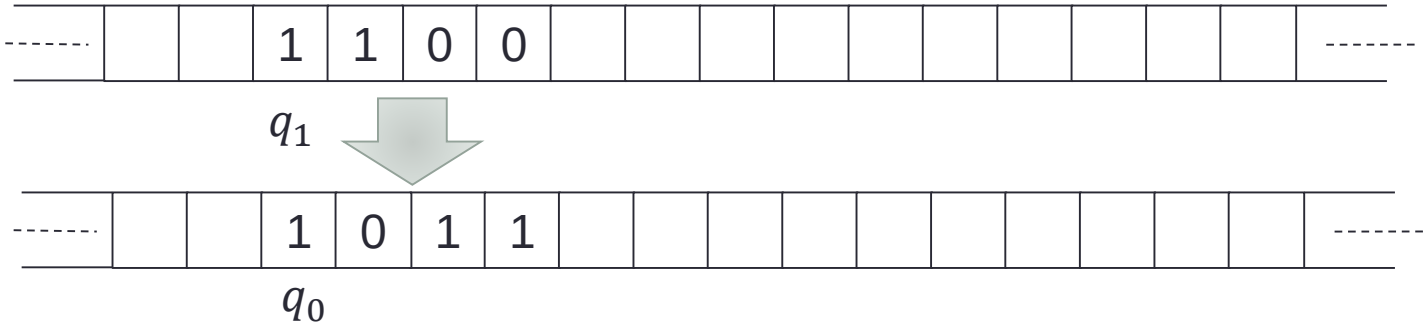


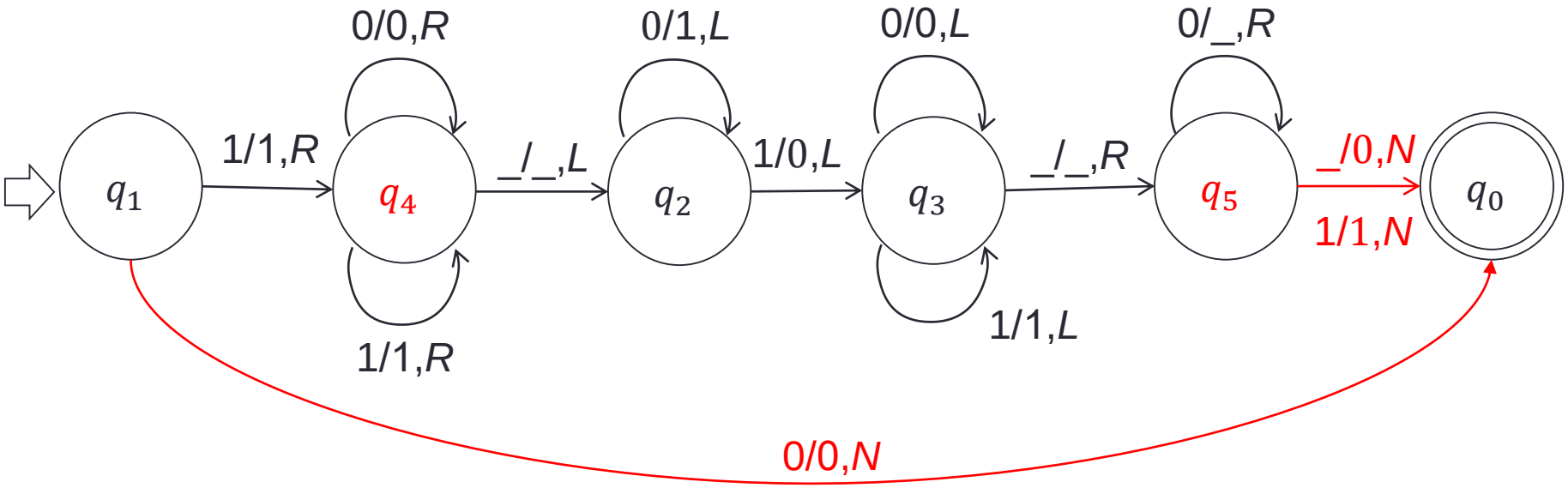
- 先頭の0を消していない
- 与えられた数字が0のときは引けない

チューリング機械の例(4)つづき

- テープ上の二進数の数字から1を引くチューリング機械を作りなさい.

$$M_6 = (\{\_, 0, 1\}, \{q_0, q_1, q_2, \cdots\}, T_6)$$

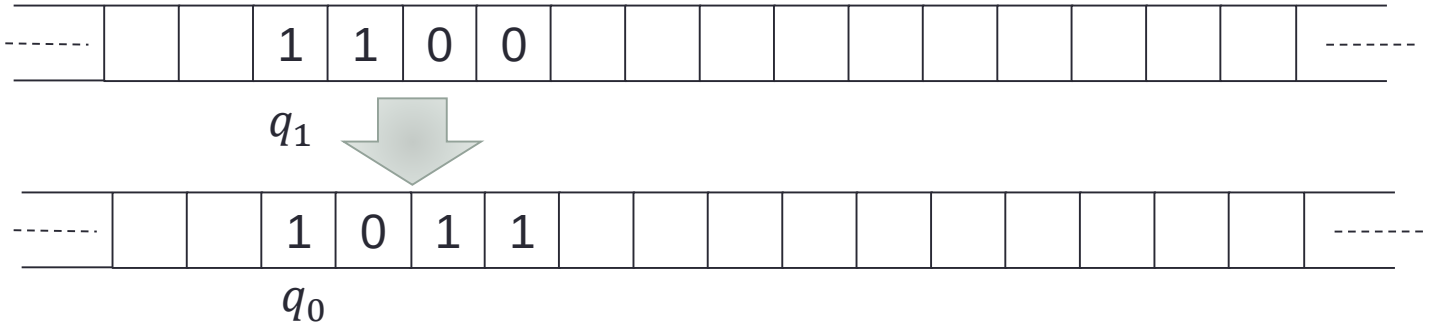




チューリング機械の例(4)つづき

- テープ上の二進数の数字から1を引くチューリング機械を作りなさい.

$$M_6 = (\{\_, 0, 1\}, \{q_0, q_1, q_2, \cdots\}, T_6)$$

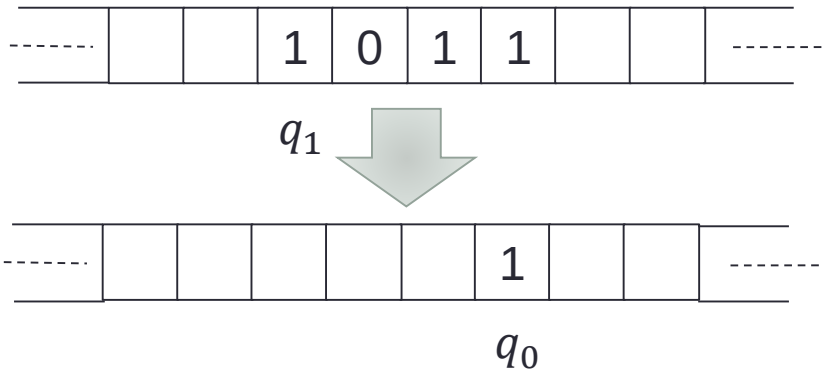


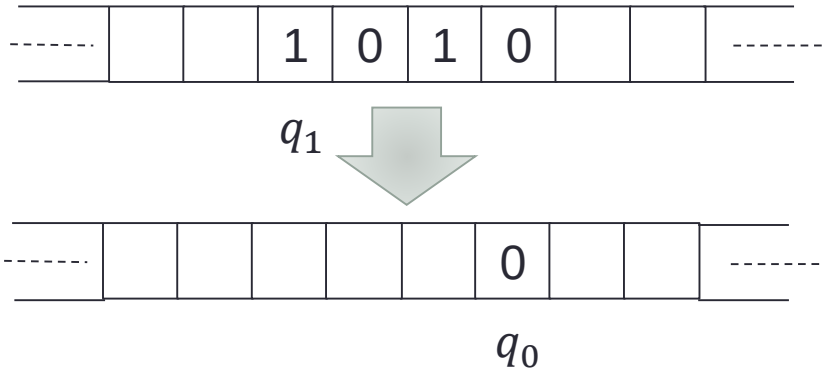
T_6	_	0	1
q_1	$(q_2,\_,L)$	$(q_1,0,R)$	$(q_1,1,R)$
q_2	$(q_0,1,N)$	$(q_2,1,L)$	$(q_3,0,L)$
q_3	$(q_0,\_,R)$	$(q_3,0,L)$	$(q_3,1,L)$

T_6	_	0	1
q_1		$(q_0,0,N)$	$(q_4,1,R)$
q_4	$(q_2,\_,L)$	$(q_4,0,R)$	$(q_4,1,R)$
q_2	$(q_0,1,N)$	$(q_2,1,L)$	$(q_3,0,L)$
q_3	$(q_5,\_,R)$	$(q_3,0,L)$	$(q_3,1,L)$
q_5	$(q_0,0,N)$	$(q_5,\_,R)$	$(q_0,1,N)$

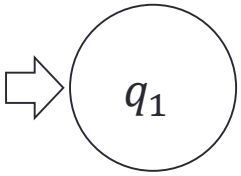
チューリング機械の例(5)演習問題

- テープ上の二進数の数字を2で割った余りを求めるチューリング機械を作りなさい.



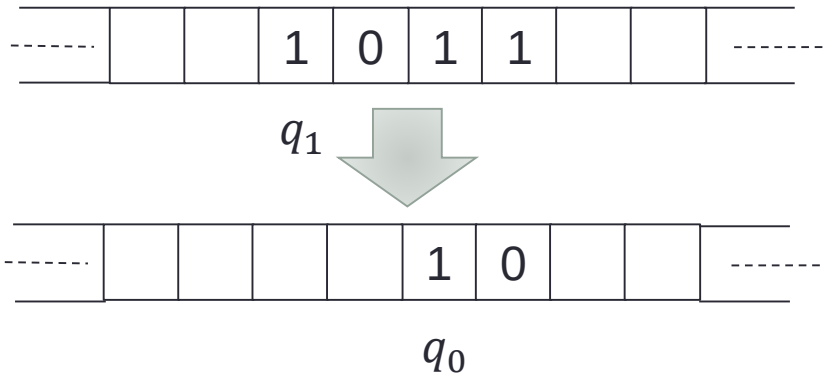


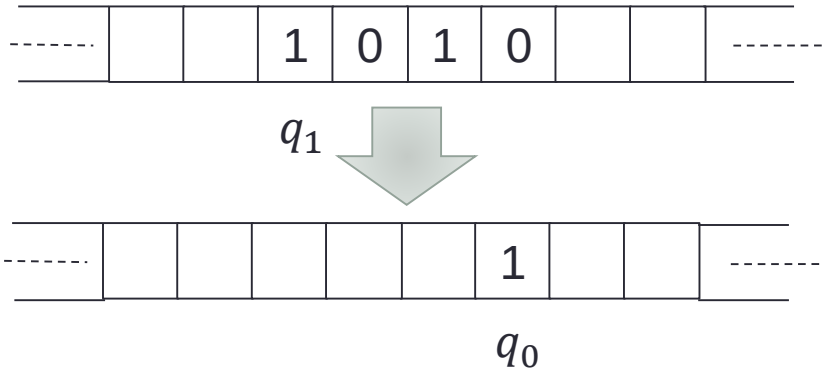
T	_	0	1
q_1			
q_2			



チューリング機械の例(6)演習問題

- テープ上の二進数の数字を3で割った余りを求めるチューリング機械を作りなさい.



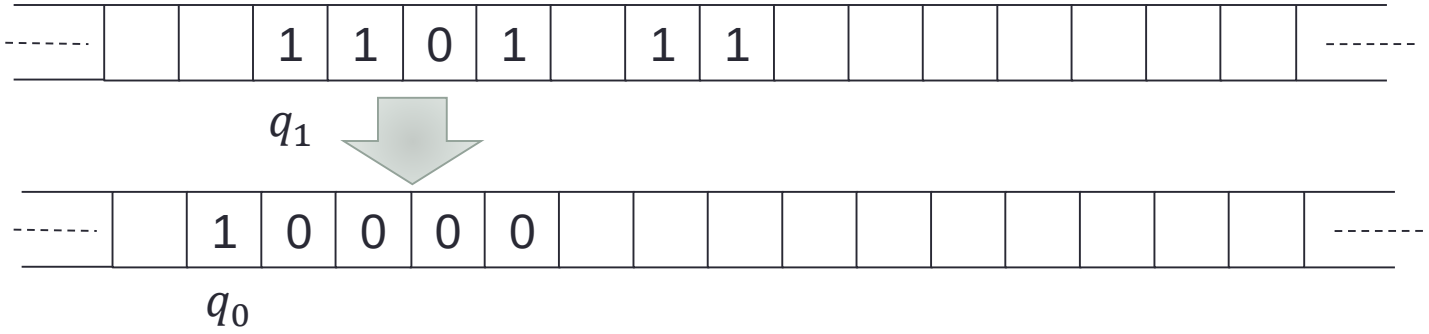


T	_	0	1
q_1			
q_2			
q_3			
q_4			

チューリング機械の例(7)演習問題

- テープ上の2つの二進数を加えるチューリング機械を作りなさい.

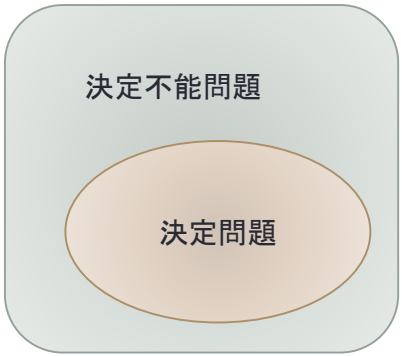
$$M_7 = (\{\_, 0, 1\}, \{q_0, q_1, q_2, \cdots\}, T_7)$$



- 1を加えるチューリング機械と1を引くチューリング機械を組み合わせる.
  - 2つ目の数が0でない場合
    - 2つ目の数から1を引く
    - 1つ目の数に1を加える
  - 2つ目の数が0になるまで繰り返す

決定可能問題

- **決定可能**問題
  - プログラムでyesかnoか判定できる問題
  - プログラムは停止する必要がある.
  - 対応する帰納的関数は全域的.



- **決定不能**問題
  - 決定可能でない問題
  - プログラムはyesとは答えるが，noが出せないかもしれない.
  - 対応する関数は帰納的ではあるが，全域的ではない.

- **停止問題**:
  - 任意のプログラムPとその入力$a_1, \ldots, a_n$が与えられた時に，プログラムPがこの入力に対して停止し計算結果を出すかどうかを調べるプログラムは存在するか?

プログラムのコード化

- プログラムを別のプログラムの入力として与えるために，プログラムを数字で表す（コード化する）．

- フローチャート
  - 箱が矢印で結ばれている
  - 箱に番号を付ける
  - それぞれの箱は次のどれか
    - $\mathrm{input}(x_1, x_2, \cdots, x_n)$
    - $x_i := m$
    - $x_i := x_j + x_k$
    - $x_i := x_j - x_k$
    - $x_i := x_j \times x_k$
    - $x_i := x_j \div x_k$
    - $\mathrm{if}\,(x_i = x_j)$
    - $\mathrm{output}(x_i)$

コード化

- P の入力変数を $x_1, \ldots, x_n$ としその他の変数を $x_{n+1}, x_{n+2}, \ldots, x_t$ とする.
- P の箱を $A_1, A_2, \ldots, A_l$ とする（A_1 は入力の箱，A_l は出力の箱）
- Gödel数を用いて箱をコード化する：$\#A$

A_a	$\#A_a$
$\mathrm{input}(x_1, x_2, \cdots, x_n)$	$\langle 1, n, a' \rangle$
$x_i := m$	$\langle 2, i, m, a' \rangle$
$x_i := x_j + x_k$	$\langle 3, i, j, k, a' \rangle$
$x_i := x_j - x_k$	$\langle 4, i, j, k, a' \rangle$
$x_i := x_j \times x_k$	$\langle 5, i, j, k, a' \rangle$
$x_i := x_j \div x_k$	$\langle 6, i, j, k, a' \rangle$
$\mathrm{if}\,(x_i = x_j)$	$\langle 7, i, j, a', a'' \rangle$
$\mathrm{output}(x_i)$	$\langle 8, i \rangle$

- プログラムのコード化は次のようになる：
  - $\#P = \langle \#A_1, \#A_2, \ldots, \#A_l \rangle$

P のインタープリタ

定理: 次の部分関数 $\mathrm{comp}_n : N^{n+1} \to N$ は帰納的関数である.

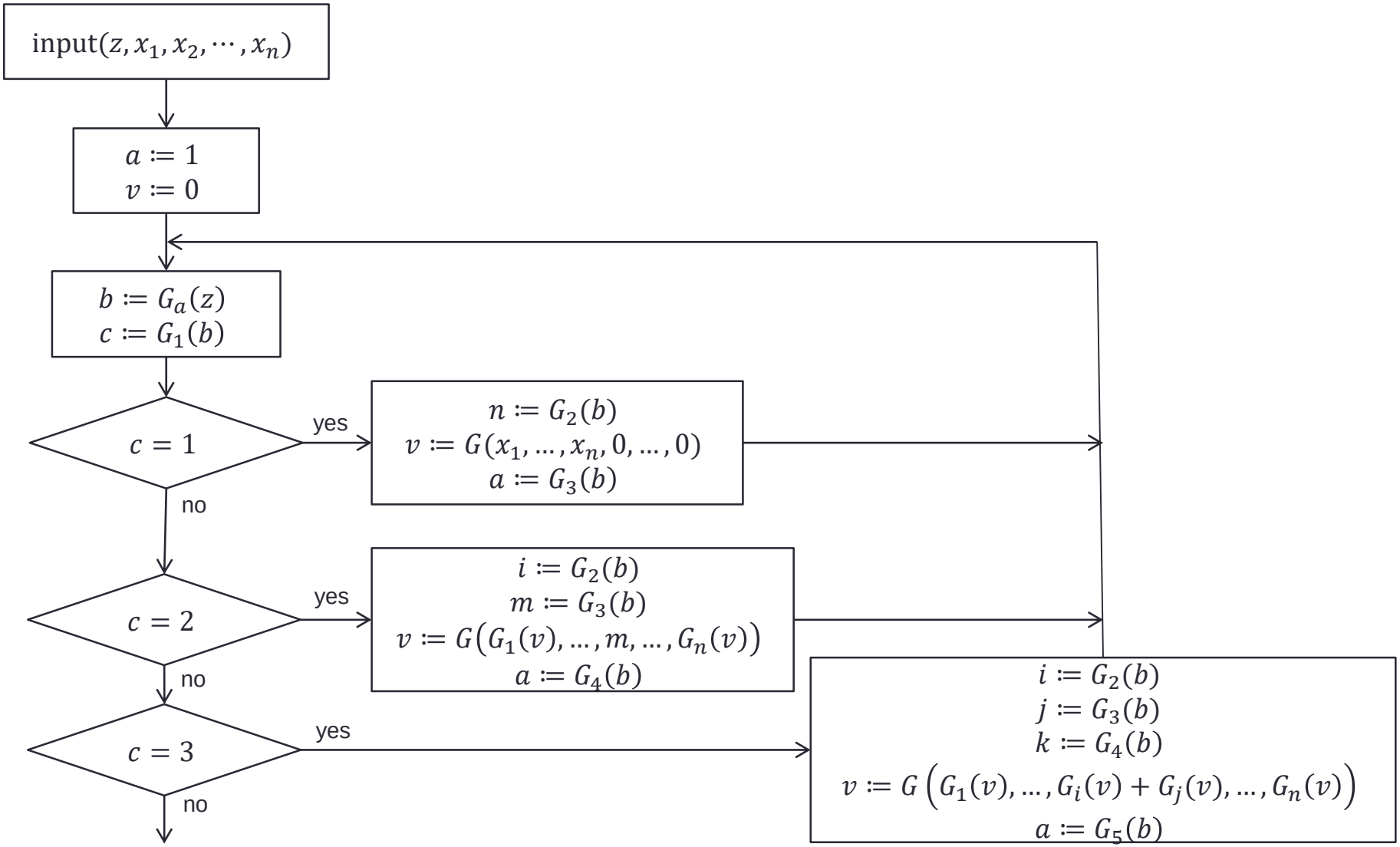
$$\mathrm{comp}_n(z, x_1, \ldots, x_n) = \begin{cases} y & z = \#P \text{ かつ } y = f_P(x_1, \ldots x_n) \\ \text{未定義} & \text{上記以外} \end{cases}$$

ここで f_P はプログラム P が計算する帰納的関数.

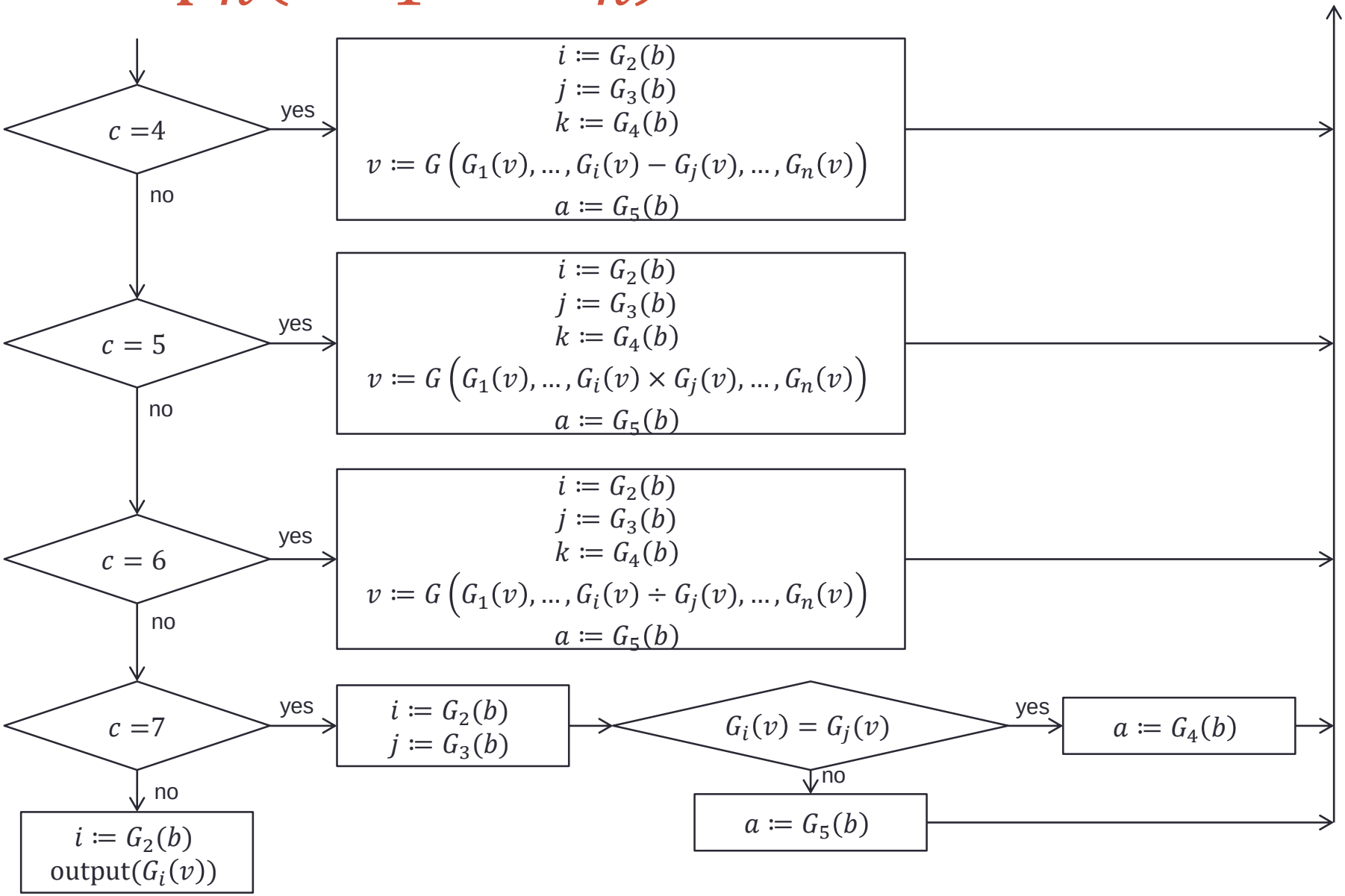
証明:

- comp_n を計算するプログラムを書けばよい.
- $\#P$ が表しているフローチャートのプログラムをシュミレーションする.

$\mathrm{comp}_n(z, x_1, \ldots, x_n)$



$\mathrm{comp}_n(z, x_1, \ldots, x_n)$つづき



compは全域的か？

定理: $\mathrm{comp}_n: N^{n+1} \to N$ を拡張して全域的関数 $g: N^{n+1} \to N$ を作ったとすると，g は帰納的ではない．

証明:

- $n = 1$ の場合を示す．
  - カントールの対角線論法を用いて，背理法により証明する．
  - $\mathrm{comp}_1(z, x) = g(z, x)$ となる全域的関数 $g: N^2 \to N$ があったとする．
  - $h(x) = g(x, x) + 1$ とすると，h は全域的な帰納的関数．
  - h を計算するプログラムのコードを c とする．
  - comp_1 の定義により，$h(x) = \mathrm{comp}_1(c, x)$．
  - h に入力として c を与えると，
  - $h(c) = \mathrm{comp}_1(c, c) = g(c, c)$
  - これは，$h(c) = g(c, c) + 1$ と矛盾する．
  - したがって，全域的な帰納的関数 g は存在しない．(QED)

帰納的述語

定義: 述語 $P: N^n \to \{T, F\}$ の特性関数 $C_P: N^n \to N$ が帰納的関数の時，P を帰納的述語という.

- C_P は全域的.
- P は決定可能.

- $P(x_1, \ldots, x_n)$, $Q(x_1, \ldots, x_n)$, $R(x_1, \ldots, x_n, y)$ が帰納的述語であれば，以下の述語も帰納的述語である.
  - $P(x_1, \ldots, x_n) \wedge Q(x_1, \ldots, x_n)$
  - $P(x_1, \ldots, x_n) \vee Q(x_1, \ldots, x_n)$
  - $\neg P(x_1, \ldots, x_n)$
  - $\forall z < y\big(R(x_1, \ldots, x_n, z)\big)$
  - $\exists z < y\big(R(x_1, \ldots, x_n, z)\big)$

停止問題は決定不能

- 述語 $\mathrm{halt_n}(z, x_1, \ldots, x_n)$： $N^{n+1} \to \{T, F\}$ を次のように定義する:

$$\mathrm{halt}_n(z, x_1, \ldots, x_n) = \begin{cases} T & \mathrm{comp}_n(z, x_1, \ldots, x_n) \text{ が定義されているとき} \\ F & \mathrm{comp}_n(z, x_1, \ldots, x_n) \text{ が定義されないとき} \end{cases}$$

定理: $\mathrm{halt_n}(z, x_1, \ldots, x_n)$ は帰納的述語ではない（決定不能である）

証明:

- $\mathrm{halt}_n(z, x_1, \ldots, x_n)$ が帰納的述語であれば，その特性関数 C_{halt_n} は帰納的な全域的関数である．すると，

$$g(z, x_1, \ldots, x_n) = C_{\mathrm{halt}_n}(z, x_1, \ldots, x_n) \times \mathrm{comp}_n(z, x_1, \ldots, x_n)$$

は帰納的な全域的関数であり，これは前の定理と矛盾する．(QED)

全域問題は決定不能

定理: $n > 0$ に対して，次の条件を満たす全域的な帰納的関数 $g: N^{n+1} \to N$ は存在しない．

$$\{ g(c, x_1, \ldots, x_n): N^{n+1} \to N \mid c \in N \} = \{ f: N^n \to N \mid f \text{ は全域的かつ帰納的} \}$$

- $\mathrm{comp}_n(z, x_1, \ldots, x_n): N^{n+1} \to N$ は帰納的関数に対する（帰納的）万能関数であるが，部分的関数である，全域的な（帰納的）万能関数は存在しない．

証明:

- $n = 1$ の場合，もし $g: N^2 \to N$ が存在すると仮定すると，$f(x) = g(x, x) + 1$ は全域的な帰納的関数．
- c を f のコードとすると，$g(c, x) = f(x) = g(x, x) + 1$ であり，これは $x = c$ の時に矛盾する．
- $n > 1$ の場合にも同様に証明することができる．（QED）

系: $\mathrm{total}_n(z) \equiv \forall x_1 \cdots \forall x_n \big(\mathrm{halt}_n(z, x_1, \ldots, x_n)\big)$ は帰納的述語ではない．すなわち，$\mathrm{total}_n(z)$ は決定不能．

証明: C_{total_n} を total_n の特性関数とすると，

$$g(z, x_1, \ldots, x_n) = C_{\mathrm{total}_n}(z) \times \mathrm{comp}_n(z, x_1, \ldots, x_n)$$

g は全域的な帰納的関数であるが，これは前の定理と矛盾する．（QED）

決定不能述語

- $\mathrm{halt}_n(z, x_1, \ldots, x_n)$
  - プログラム z が入力 $x_1, \ldots, x_n$ に対して停止するのかどうか.
- $\mathrm{total}_n(z)$
  - プログラム z が常に停止するかどうか.
- $\forall x_1 \cdots \forall x_n(\mathrm{comp}_n(z, x_1, \ldots, x_n) = 0)$
  - プログラム z がどんな入力に対しても常に0を出力するかどうか.
- $\exists x_1 \cdots \exists x_n(\mathrm{comp}_n(z, x_1, \ldots, x_n) = 0)$
  - プログラム z がある入力に対して0を出力することがあるかどうか.
- z に関して $\mathrm{comp}_n(z, x_1, \ldots, x_n)$ の定義域が有限集合である.
  - プログラム z が有限個の入力に対して定義されているかどうか.
- z に関して $\mathrm{comp}_n(z, x_1, \ldots, x_n)$ が定数関数である.
  - プログラム z が常に同じ値を出力するかどうか.
- z と z' に関して $\mathrm{comp}_n(z, x_1, \ldots, x_n) = \mathrm{comp}_n(z', x_1, \ldots, x_n)$
  - 2つのプログラム z と z' が同じかどうか.

s-m-n定理

定理: 自然数 m と n に対して，次の性質を満たす原始帰納的関数 $S_{m,n}: N^{m+1} \to N$ が存在する.

$$\mathrm{comp}_{m+n}(z, x_1, \ldots, x_n, y_1, \ldots, y_m) = \mathrm{comp}_n\big(S_{m,n}(z, y_1, \ldots, y_m), x_1, \ldots, x_n\big)$$

証明: $S_{m,n}(z, u_1, \ldots, u_m)$ は与えられたプログラム $z = \langle \#A_1, \#A_2, \ldots, \#A_l \rangle$ を

$$z' = \big\langle \#\big(\mathrm{input}(x_1, \ldots, x_n)\big), \#(y_1 := u_1), \ldots, \#(y_m := u_m), \#A_2, \ldots, \#A_l \big\rangle$$

に変換する. これは次のプログラムに対応する.

- $\mathrm{input}(x_1, \ldots, x_n)$
- $y_1 := u_1$
- $\ldots$
- $y_m := u_m$
- A_2
- $\ldots$
- A_l

ゲーデル関数を使って入力のプログラムを分解し，先頭にいくつかの数字をつけたものをゲーデル関数で変換しなおせばよいので，原始帰納的関数として書くことができる. (QED)

再帰定理

定理: 自然数 n と帰納的全域的関数 $f: N \to N$ に対して，次の式を満たす自然数 c が存在する.

$$\mathrm{comp}_n(f(c), x_1, \ldots, x_n) = \mathrm{comp}_n(c, x_1, \ldots, x_n)$$

証明:

- a を $\mathrm{comp}_{n+1}(y, x_1, \ldots, x_n, y)$ のコードとする.
- $\mathrm{comp}_{n+1}(y, x_1, \ldots, x_n, y) = \mathrm{comp}_{n+1}(a, x_1, \ldots, x_n, y) = \mathrm{comp}_n\left(S_{1,n}(a, y), x_1, \ldots, x_n\right)$
- b を $\mathrm{comp}_n\left(f\left(S_{1,n}(a, y)\right), x_1, \ldots, x_n\right)$ のコードとする.
- $\mathrm{comp}_n\left(f\left(S_{1,n}(a, y)\right), x_1, \ldots, x_n\right) = \mathrm{comp}_{n+1}(b, x_1, \ldots, x_n, y)$
- $\mathrm{comp}_n\left(f\left(S_{1,n}(a, b)\right), x_1, \ldots, x_n\right) = \mathrm{comp}_{n+1}(b, x_1, \ldots, x_n, b) = \mathrm{comp}_n\left(S_{1,n}(a, b), x_1, \ldots, x_n\right)$
- $c = S_{1,n}(a, b)$

(QED)

Riceの定理

定理: n を自然数とする. 述語 $P(z)$ が次の2つの条件を満たすとき, $P(z)$ は帰納的述語ではない.

(1) $\forall c \forall c' \left(\forall x_1 \forall x_2 \cdots \forall x_n \left(\mathrm{comp}_n(c, x_1, \ldots, x_n) = \mathrm{comp}_n(c', x_1, \ldots, x_n) \right) \Longrightarrow P(c) \equiv P(c') \right)$

(2) $\exists c \exists c' \left(P(c) \wedge \neg P(c') \right)$

証明: P が帰納的述語であるとして, その特性関数を C_P とする. (2)を満たす c, c' に対して, $f: N \to N$ を次のように定義する.

$$f(z) = C_{P(z)} \times c' + \left(1 - C_{P(z)}\right) \times c$$

- どのような z に対しても $P\left(f(z)\right) \not\equiv P(z)$
  - $P(z)$ が真ならば $f(z) = c'$ なので $P(f(z))$ は偽
  - $P(z)$ が偽ならば $f(z) = c$ なので $P(f(z))$ は真
- f は帰納的な全域的関数なので再帰定理より

  $\mathrm{comp}_n(f(c''), x_1, \cdots x_n) = \mathrm{comp}_n(c'', x_1, \cdots x_n)$

  となる c'' が存在する.
- (1)より $P\left(f(c'')\right) \equiv P(c'')$ であるが, これは矛盾する. (QED)

- この定理を使って多数の決定不能な述語を証明することができる.

Post対応問題

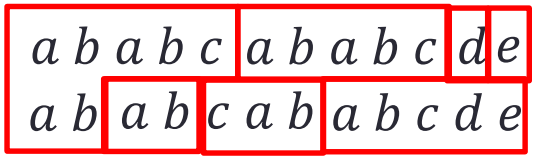
問題:有限個の文字列の対の集合が与えられた時,

$$\{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$$

文字列の連結を使って,次のようになる数列 $i_1, \ldots, i_m$ が存在するか?

$$s_{i_1} s_{i_2} \ldots s_{i_m} = t_{i_1} t_{i_2} \ldots t_{i_m}$$

例: $\{(e, abcde), (ababc, ab), (d, cab)\}$



- この問題は決定不能(計算機で解く一般的な方法はない.プログラムは止まらない.)

まとめ

- 決定可能な問題
  - プログラムはyesあるいはnoとして答えることができる問題.
- 決定不能な問題
  - 決定可能ではない問題
- 決定不能な述語
  - 停止問題
  - 全域性問題
  - Post対応問題