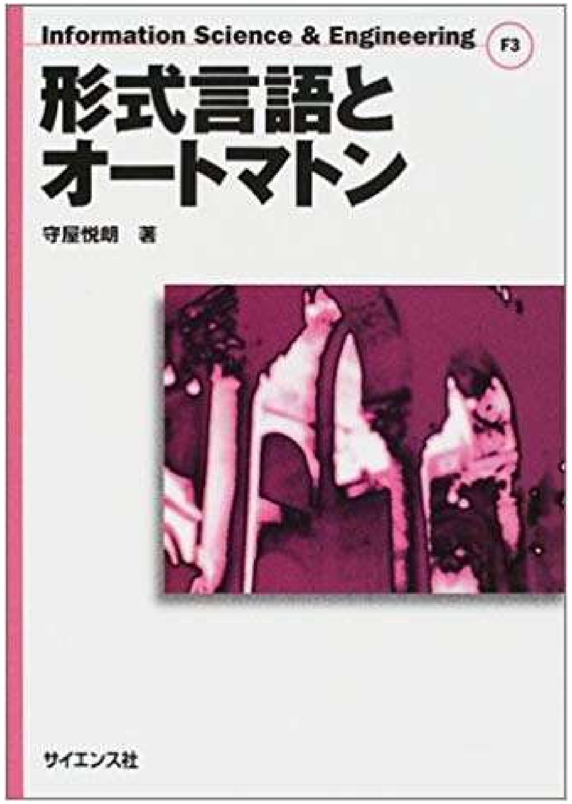
計算理論　第15回
文脈依存言語・
チューリングマシン

基礎工学部情報科学科
中川　博之

参考文献



- 形式言語とオートマトン
  - 守屋 悦朗
  - サイエンス社
- オートマトン言語理論 計算論II
  - J.ホップクロフト, R.モトワニ, J.ウルマン
  - サイエンス社

文脈依存文法と文脈依存言語

文脈依存文法 (CSG)

- 文脈依存文法 (Context-Sensitive Grammar: CSG)
- 4つ組 G = (V, T, P, S)で表現
  - V, T, S : CFGと同じ
  - P : 生成規則の有限集合
    - CFGより規則が緩い

CSGの生成規則

- 形式 : $\alpha X\beta \to \alpha\gamma\beta$
  - Xをγで置き換える
  - Xがαとβに挟まれていることが条件
  - 例 : aXB → aX0B, X → XY1
- 頭部 αXβ : 置き換え対象 X
  - $\alpha, \beta \in (V \cup T)^*$, $X \in V$
- 本体 αγβ : 置き換える列γ
  - $\gamma \in (V \cup T)^+$　※長さは1以上
  - 生成規則適用により<u>列が短くなることはない</u>
- 最終的に終端記号だけの列に

文脈の意味

- 生成規則：$\alpha X \beta \rightarrow \alpha \gamma \beta$
  - 置き換わるのはXのみ(γに変換)
  - αとβはそのまま
- Xの前後のαとβが文脈
  - 置き換えはXの出現する文脈に応じて（依存して）実施

文脈依存言語

- 文脈依存言語（Context-Sensitive Language：CSL）
  - 文脈依存文法で生成される言語
  - LがCSLならば，L∪{ε}もCSLと定める

- 文脈依存言語のクラス
  = 文脈依存文法で生成される言語のクラス

単調文法

- すべての生成規則 $\alpha \to \beta$ が $|\alpha| \leq |\beta|$ である文法
- L(単調文法)=L(CSG)
  - 単調文法で生成される言語のクラス
    = 文脈依存言語のクラス

生成規則AB→BAの導入

- 生成規則AB→BAを利用しても良い
  - 2変数の前後を交代させる式
  - 本来のCSGでは許されていない
- 利用しても良い理由
  - 新たに変数X, Yを導入し,
  - 生成規則1 : AB→XB
  - 生成規則2 : XB→XY
  - 生成規則3 : XY→BY
  - 生成規則4 : BY→BA
  - これらはいずれもCSGで許されている生成規則
  - よって，$AB \overset{*}{\Rightarrow} BA$

文法例

- 言語$L=\{a^n b^n c^n \mid n \geq 1\}$
  - CFLではない
- 生成規則
  - $S \rightarrow aSBC$
  - $S \rightarrow aBC$
  - $aB \rightarrow ab$
  - $bC \rightarrow bc$
  - $CB \rightarrow BC$
  - $bB \rightarrow bb$
  - $cC \rightarrow cc$
- 導出例

  S

  $\Rightarrow aSBC$

  $\Rightarrow aaBCBC$

  $\Rightarrow aabCBC$

  $\Rightarrow aabBCC$

  $\Rightarrow aabbCC$

  $\Rightarrow aabbcC$

  $\Rightarrow aabbcc$

CSGの標準形：黒田標準形

- 生成規則を以下の形に限定した文法
  - X→UV
  - XY→UV
  - X→a
  - ただし，X, Y, U, V∈V, a∈T
- 任意のCSGに対して，それと等価な黒田標準形が存在

線形有界オートマトン

線形有界オートマトン

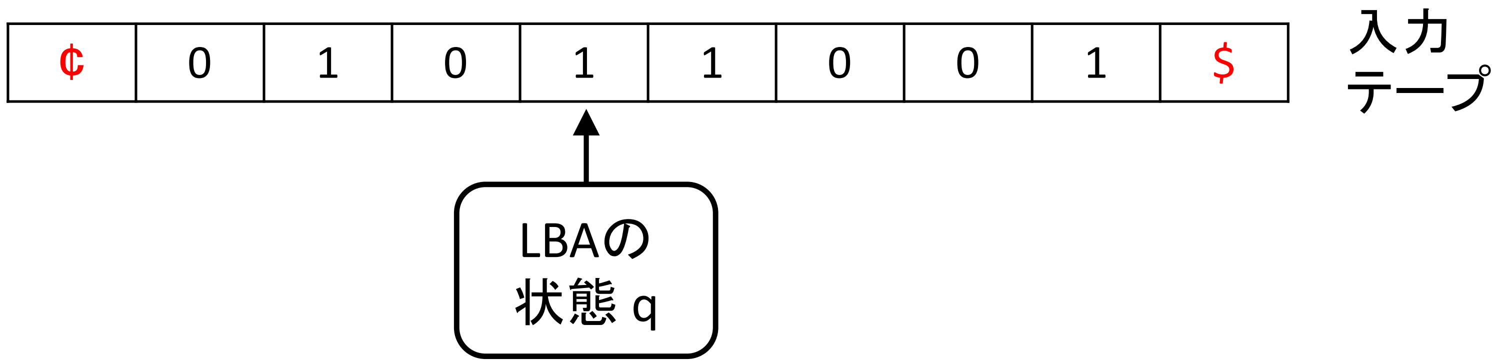
- 線形有界オートマトン
(Linear-Bounded Automaton: LBA)

- CSLを認識するオートマトン

- LBAの受理する言語のクラスはCSLのクラスに一致

線形有界オートマトン(LBA)の概要

- 有限制御部
  - 状態遷移関数に従って動作（決定性/非決定性）
- テープ
  - 入力記号列が書き込まれて与えられる
  - 左端記号 ¢，右端記号$ が置かれる
  - ヘッドを左右に動かし，テープ上の入力記号列を書き換えることができる



LBAの定義

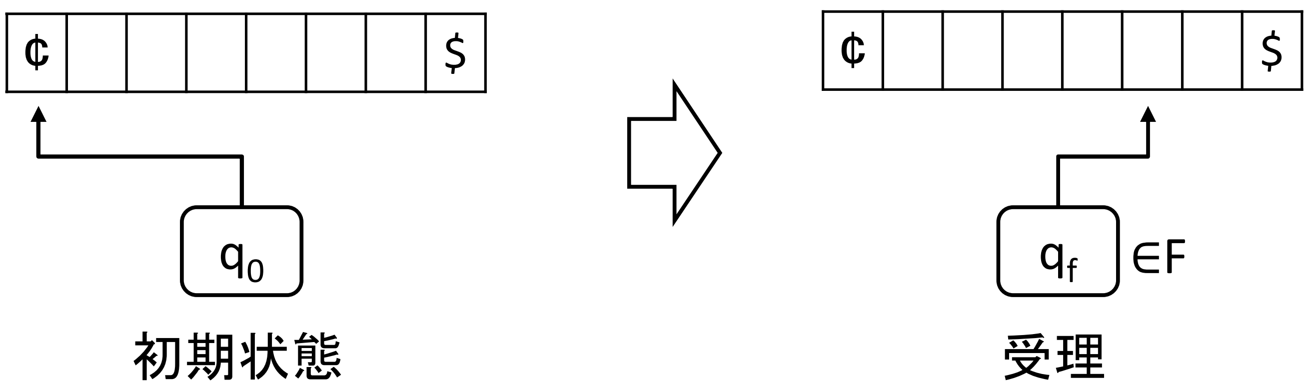
- 8つ組 $M=(Q, \Sigma, \Gamma, \delta, q_0, ¢, \$, F)$
  - Q: 状態集合
  - Σ: 入力記号集合($\Sigma \subseteq \Gamma$)
  - Γ: テープ記号の集合 (有限集合)
  - δ: 遷移関数
  - q_0: 初期状態 ($\in Q$)
  - ¢, $: 入力の左端右端を表す終止符 (endmarker)
    - Γの元ではない特殊記号
  - F: 受理状態の集合 ($\subseteq Q$)

LBAの遷移関数δ

- 遷移関数δ:
  - $Q \times \Gamma_{¢\$}$から$Q \times \Gamma_{¢\$} \times \{-1, 0, 1\}$の部分集合への関数
- 入力:
  - 現在の状態: $q \in Q$
  - ヘッドが読む記号: $a \in \Gamma \cup \{¢, \$\}$
- 出力:
  - 次の状態: $p \in Q$
  - ヘッド位置の記号の書き換え: $b \in \Gamma \cup \{¢, \$\}$
  - ヘッドの移動量: -1 (左), 0 (動かさない), 1 (右)

LBAの動作

- 動作開始時
  - 状態：初期状態q_0
  - ヘッドは左端：¢ 上
- 受理状態 (∈F)に到達すれば受理
  - ヘッドは右端 ($) を指していなくても良い



LBAの例1: $L=\{a^n b^n c^n \mid n \geq 1\}$

- LBAの動作概要
  - a, b, cを一つずつ消し，最後に全てが同時に消えれば良い

1) 初期状態におけるテープ

¢	a	a	a	b	b	b	c	c	c	$

2) 左にあるa, b, cをそれぞれA, B, Cに書き換える

¢	A	a	a	B	b	b	C	c	c	$

3) これを繰り返し，a, b, cが同時に無くなると受理

LBAの例2:
$L=\{w_1cw_2 \mid w_1,w_2 \in \{a, b\}^*, w_1=w_2\}$

- 言語LはCFLではない (反復補題で証明可能)
- LBAの動作概要
  - w1, w2を左から一文字ずつ照合
  - 照合が済んだ文字はAまたはBに書き換え

1) 初期状態におけるテープ

¢	b	a	b	b	c	b	a	b	b	$

2) w1の左端とw2の左端の未照合文字を照合
- 一致しない時点で棄却

¢	B	a	b	b	c	B	a	b	b	$

3) これを繰り返し，全ての文字が合致すれば受理

非決定性LBAと決定性LBA

- 非決定性LBA (Non-deterministic LBA: NLBA)
  - 複数の遷移が存在
  - いずれかのパスで受理に至れば入力を受理

- 決定性LBA (Deterministic LBA: DLBA)
  - 各状態において遷移は高々1つ

NLBAの受理能力

- 定理：NLBAが受理する言語クラス L(NLBA) = 文脈依存言語のクラス L(CSG)
- 証明概要
  - ⇐の証明
    - 任意のCSGと等価な黒田標準形文法G’が存在
    - G’の導出を模倣するNLBAが構成可能
  - ⇒の証明
    - 任意のNLBAの遷移関数をCSGの生成規則で表現可能
    - NLBAが入力を受理するとき，その受理計算に対応する導出が存在

文脈依存言語 (CSL) の性質

反復補題

- CSLに対する反復補題
  → まだ知られていない

- CSLでない具体的な言語例を示すのは難しい

閉包性

- CSLは以下の演算のもとで閉じている
  - 和集合 (∪)
  - 共通部分 (∩)
  - 差集合 (\)
  - 補集合 (¯)
  - 連接 (･)
  - Kleene閉包 (*)
  - 反転(鏡像) (^{R})
- CSLは以下の演算のもとで閉じていない
  - 準同型写像
  - 代入

証明例：共通部分

- L_1, L_2 : CSL

- M_1, M_2 : それぞれL_1, L_2を受理するNLBAとするとき，M_1, M_2を使って$L_1 \cap L_2$を受理するNLBA Mが構成できることを示せばよい

- Mの概要
  1. L1の語か否かを検査
  2. L2の語か否かを検査
  3. 両者の語であれば受理

NLBA Mの構成法 (1/3)

- Step 0: 準備

¢	0	1	1	0	$

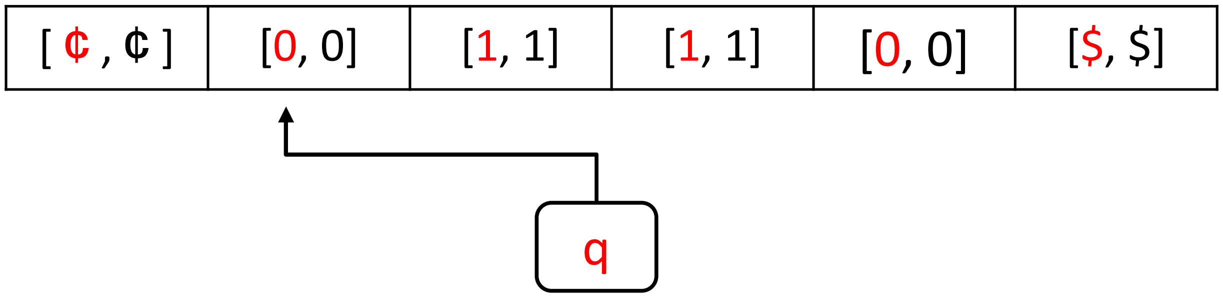
  - 入力テープの各コマの内容aを[a,a]と書き換える

[¢ , ¢]	[0, 0]	[1, 1]	[1, 1]	[0, 0]	[$, $]

  - 左成分をM_1が処理，右成分をM_2が処理

NLBA Mの構成法 (2/3)

- Step 1: L_1の認識
  - テープコマの左成分のみアクセス
  - M_1の動作を模倣してL_1を認識
  - M_1が入力を棄却すれば，Mは棄却
  - M_1が入力を受理すれば次のステップへ



NLBA Mの構成法 (3/3)

- Step 2: L_2の認識
  - テープコマの右成分のみアクセス
  - M_2の動作を模倣してL_2を認識
  - M_2が入力を棄却すれば，Mは棄却
  - M_2が入力を受理すれば，Mは受理

[¢ , ¢]	[0, 0]	[1, 1]	[1, 1]	[0, 0]	[$, $]

p

Mが受理 ⇔ 入力後は$L_1 \cap L_2$の語

Chomsky階層

Chomsky階層

- ４つの文法クラス
  - 0型文法：句構造文法
    (Phrase Structure Grammar)
  - 1型文法：文脈依存文法
    (Context-Sensitive Grammar)
  - 2型文法：文脈自由文法
    (Context-Free Grammar)
  - 3型文法：正則文法（正規文法） (Regular Grammar)

これらの文法クラスは階層をなす

3型文法:正則文法

- 生成規則の形:X→γ
  - 左辺は1つの変数
  - γ∈TVまたは γ∈T
    - 導出により文形式が短くなることはない
  - ただし, S→ε を許す (以降の文法も同様)
- 例
  - A→c
  - A→aB
  - A→aBc　×

2型文法：文脈自由文法

- 生成規則の形：X→γ
  - 左辺は1つの変数
  - $\gamma \in (V \cup T)^{+}$
    - 導出により文形式が短くなることはない
- 例
  - A→c
  - A→aB
  - A→aBc
  - bA→aBB　×

1型文法:文脈依存文法

- 生成規則の形:$\alpha X\beta \rightarrow \alpha\gamma\beta$
  - 左辺には少なくとも一つの変数
  - $\alpha, \beta \in (V \cup T)^*$
  - $\gamma \in (V \cup T)^+$
    - 導出により文形式が短くなることはない
- 例
  - A→c
  - A→aB
  - A→aBc
  - bA→aBB
  - aAaA→aBa　×

0型文法:句構造文法

- 生成規則の形:$\alpha X\beta \rightarrow \alpha\gamma\beta$
  - 左辺には少なくとも一つの変数
  - $\alpha, \beta \in (V \cup T)^*$
  - $\gamma \in (V \cup T)^*$
    - 導出により文形式が短くなっても良い
- 例
  - A→c
  - A→aB
  - A→aBc
  - bA→aBB
  - aAaA→aBa

4つの言語クラス

- 句構造言語 (Phrase Structure Language)
  - 句構造文法で生成される言語
- 文脈依存言語 (Context-Sensitive Language)
  - 文脈依存文法で生成される言語
- 文脈自由言語 (Context-Free Language)
  - 文脈自由文法で生成される言語
- 正則言語（正規言語） (Regular Language)

言語と認識機械の関係

- 句構造言語：チューリングマシン (TM)
- 文脈依存言語：線形有界オートマトン (LBA)
- 文脈自由言語：プッシュダウンオートマトン (PDA)
- 正則言語：有限オートマトン (FA)

言語クラスの階層性

- 言語クラス
  - L_{ps} : 句構造言語のクラス
  - L_{cs} : 文脈依存言語のクラス
  - L_{cf} : 文脈自由言語のクラス
  - L_{re} : 正則言語のクラス

- Chomsky階層

$$L_{ps} \supset L_{cs} \supset L_{cf} \supset L_{re}$$

  - 4つの言語クラス間には真の包含関係がある

$L_{cf} \supset L_{re}$

- 包含関係 $L_{cf} \supseteq L_{re}$ は自明
  - 生成規則の関係より
- 真の包含関係 $L_{cf} \supset L_{re}$
  - $L \in L_{cf}$ かつ $L \notin L_{re}$ である言語Lが存在
    - LはFAでは受理できないがPDAで受理できる言語

  - 例1: $L = \{a^n b^n \mid n \geq 0\}$
  - 例2: $L = \{wcw^R \mid w \in \{a, b\}^*\}$

$L_{cs} \supset L_{cf}$

- 包含関係 $L_{cs} \supseteq L_{cf}$ は自明
  - 生成規則の関係より
- 真の包含関係 $L_{cs} \supset L_{cf}$
  - $L \in L_{cs}$ かつ $L \notin L_{cf}$ である言語Lが存在
    - LはPDAでは受理できないがLBAで受理できる言語

  - 例1: $L = \{a^n b^n c^n \mid n \geq 0\}$
  - 例2: $L = \{wcw \mid w \in \{a, b\}^*\}$

$L_{ps} \supset L_{cs}$

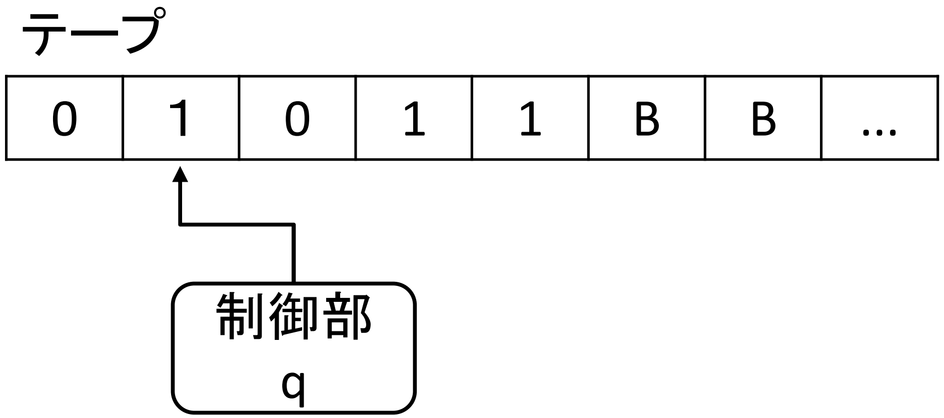
- 包含関係 $L_{ps} \supseteq L_{cs}$ は自明
  - 生成規則の関係より
- 真の包含関係 $L_{ps} \supset L_{cs}$
  - L∈ L_{ps}かつL∉ L_{cs}である言語Lが存在
    - LはLBAでは受理できないがTMで受理できる言語

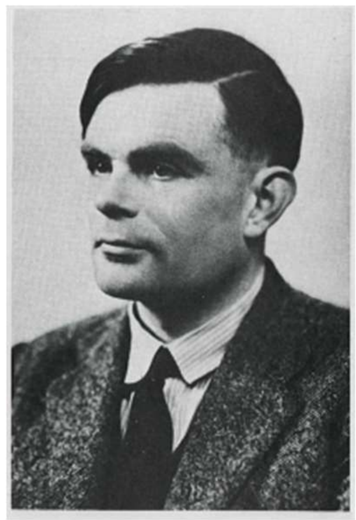
チューリングマシン

チューリングマシン

- チューリングマシン (Turing Machine: TM)
  - Alan Turingが提案した論理的機械モデル

- 有限状態数の制御部と1本のテープで構成
  - テープは右方向に無限の長さ
  - ヘッドは左右に動ける
  - テープのヘッド位置の記号を書き換え可能





Alan Mathison Turing
(1912-1954)

TMの動作

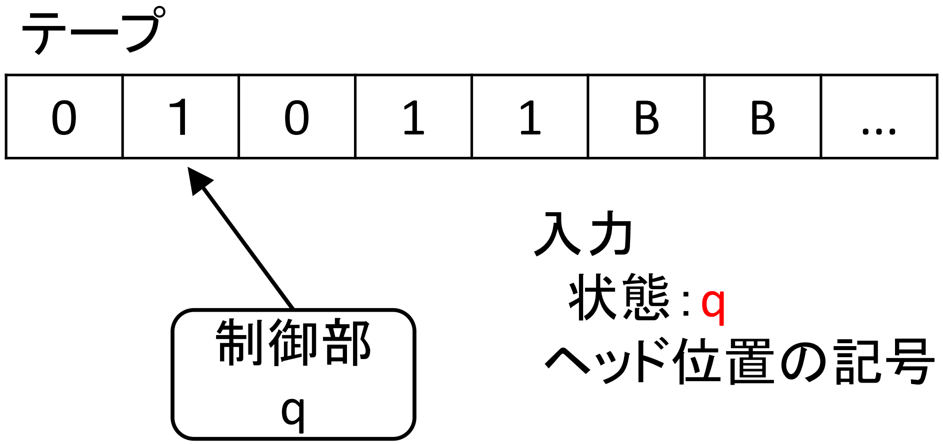
- 動作開始時
  - テープに入力記号列が書き込まれている
  - 入力の右端より先はすべて空白文字B
  - ヘッドは左端に位置
- 動作終了
  - 受理状態に遷移したとき
  - 遷移が定義されていない
- 永久に停止しない場合もあり
  - 無限ループに陥ったとき

TMの定義

- 6つ組 $M=(Q, \Sigma, \Gamma, \delta, q_0, F)$
  - Q: 有限制御部の状態集合
  - Σ: 入力記号集合 (空白文字Bは含まない)
  - Γ: テープ記号の集合 ($\Sigma \subseteq \Gamma$)
  - δ: 遷移関数
  - q_0: 初期状態 ($\in Q$)
  - F: 受理状態の集合 ($\subseteq Q$)

遷移関数の入出力

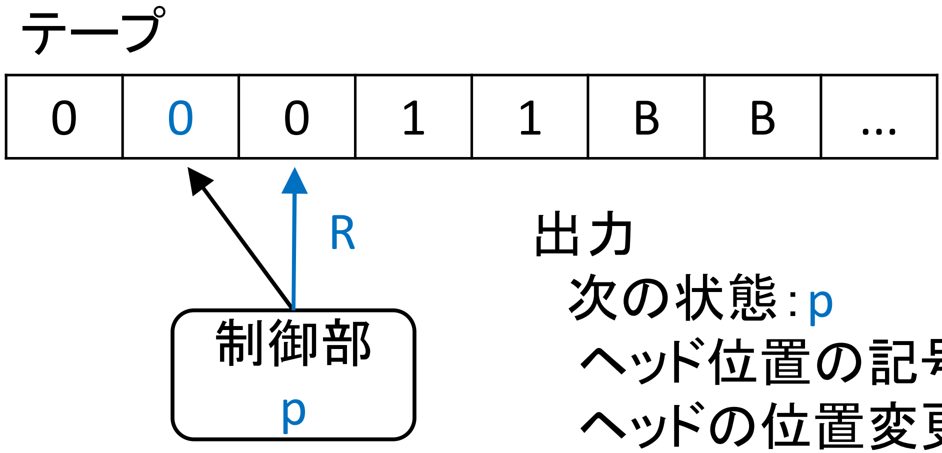
- 入力
  - 状態 (∈ Q)
  - ヘッド位置の記号 (∈Γ)



入力
状態 : q
ヘッド位置の記号 : 1

遷移関数の入出力

- 出力
  - 次の状態 (∈ Q)
  - ヘッド位置の記号の書き換え (∈Γ)
    - 空白Bを他の記号に書き換えても良い
  - ヘッドの位置変更: L (左) or R (右)



出力
次の状態: p
ヘッド位置の記号: 0
ヘッドの位置変更: R

例:L1={0^n1^n | n≥1}の認識:概要

- TMの構成方針:0と1をペアで消していく
  - 0はX, 1はYで上書き

0	0	0	0	1	1	1	1	B	B	...
X	0	0	0	**Y**	1	1	1	B	B	...
X	**X**	0	0	**Y**	**Y**	1	1	B	B	...
X	**X**	**X**	0	**Y**	**Y**	**Y**	1	B	B	...
X	**X**	**X**	**X**	**Y**	**Y**	**Y**	**Y**	B	B	...

  - 繰り返した後, 0も1も残っていない → 受理
  - いずれかが残る → 棄却

例:L1={0^n1^n | n≥1}の認識：動作解説

- テープの左端より動作を開始し，以下を繰り返す
  - 0を見つけたらXに置き換える
  - 0, Yを読み飛ばして右に移動

X	0	0	0	1	1	1	1	B	B	...

  - 1を見つけたらYに置き換える
  - Xを見つけるまで左に移動

X	0	0	0	Y	1	1	1	B	B	...

  - Xを見つけたら右隣へ移動

X	0	0	0	Y	1	1	1	B	B	...

例:L1={0^n1^n | n≥1}の認識:動作終了の判別

- 入力が0^n1^nでないとき
  - 0が多い場合:1が見つからずBを発見する
  - 1が多い場合:右端のXを見つけ，一歩右に動いたときに0ではなくYを発見．その後右に動いて1を見つける．

X	**X**	**X**	**0**	**Y**	**Y**	**Y**	B	B	...

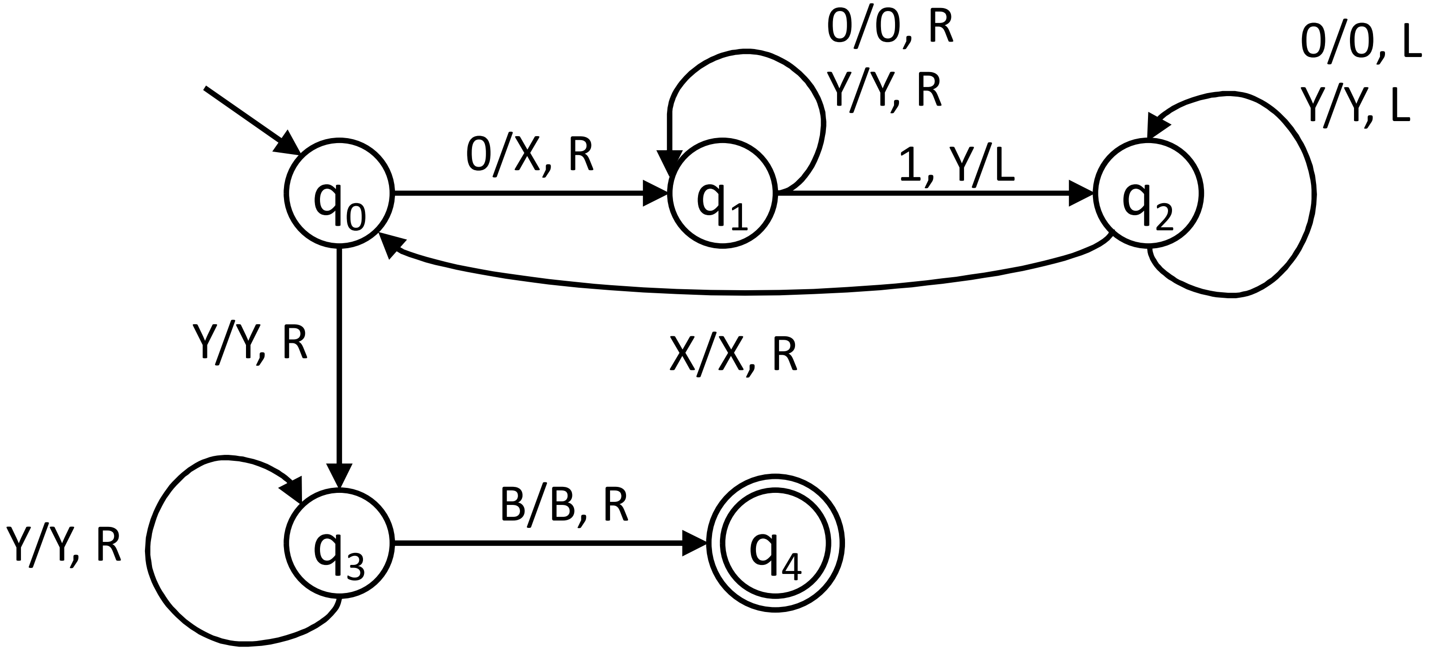
X	**X**	**X**	**Y**	**Y**	**Y**	**1**	B	B	...

- 入力が0^n1^nのとき
  - 右端のXを見つけ，一歩右に動いたときに0ではなくYを発見．その後右に動いてBを見つける．

X	**X**	**X**	**X**	**Y**	**Y**	**Y**	**Y**	B	B	...

例:L1={0^n1^n | n≥1}の認識:遷移図

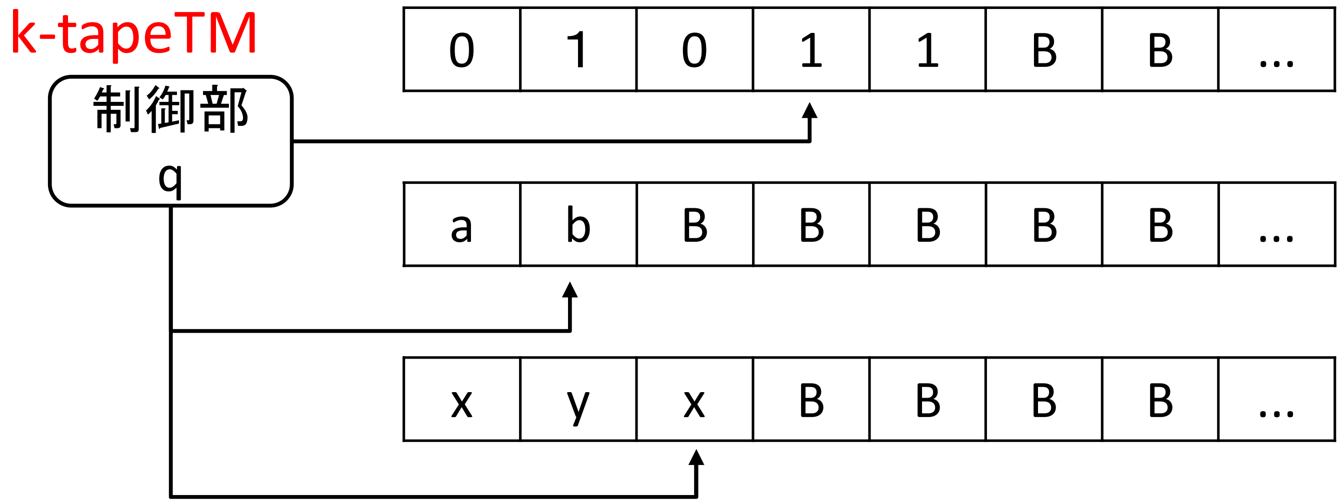
– [凡例] 0/X, R : 0を読んだらそれをXに書き換えて右(R)に移動
– 図中に遷移がない場合，棄却状態(q_5)に遷移し停止



多テープTM

多テープTM

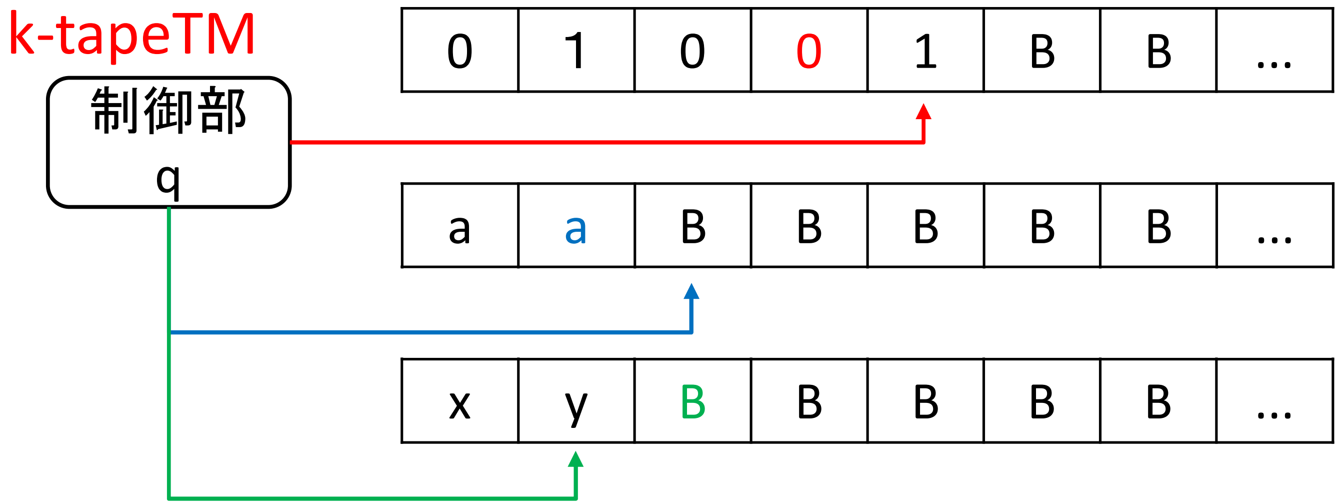
- 多テープTM：複数のテープを持つTM
  - 1本は入力テープ
  - 残りは作業用テープ（初期状態では内容は全てB）



- 利点
  - 動作記述が簡潔，処理時間を短縮

k-テープTMの遷移関数δ

- $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$
  - k個の各ヘッドが記号を読み，書き換え，移動する（L：左，R：右，S：移動せず）



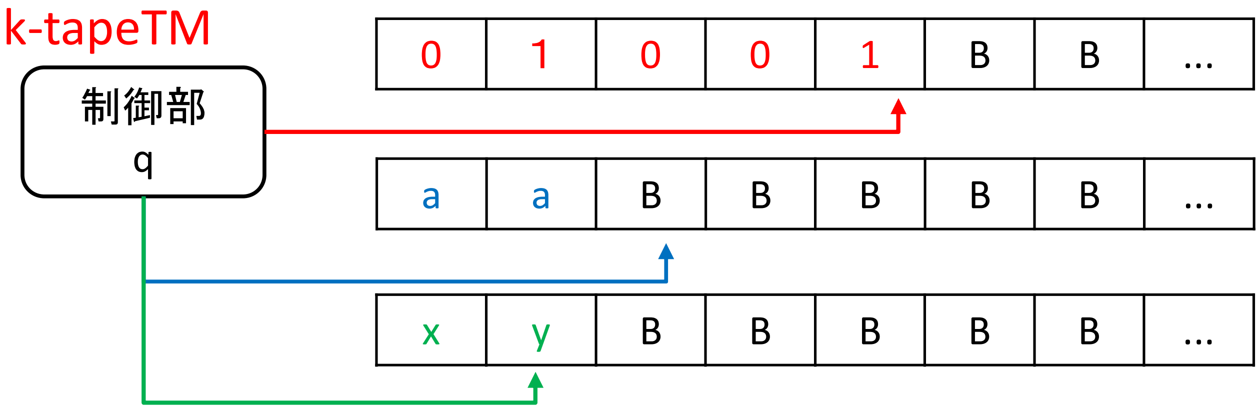
1-テープTMと多テープTMの能力

※1テープTM = 今までのTM (テープ1つ)

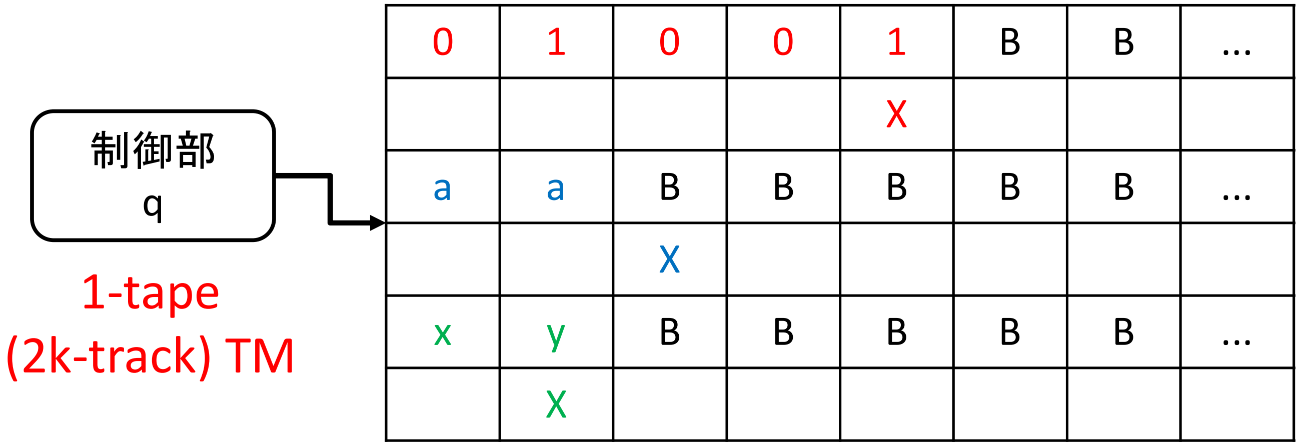
- 多テープTMの能力 ≥ 1-テープTMの能力
  - 自明
- 多テープTMの能力 ≤ 1-テープTMの能力
  - 1-テープTMで多テープTMの動作を模倣できる

- 1-テープTMと多テープTMは受理する言語のクラスが同じ
  - 受理に要する時間は同じとは限らない

1-テープTMによるk-テープTMの模倣

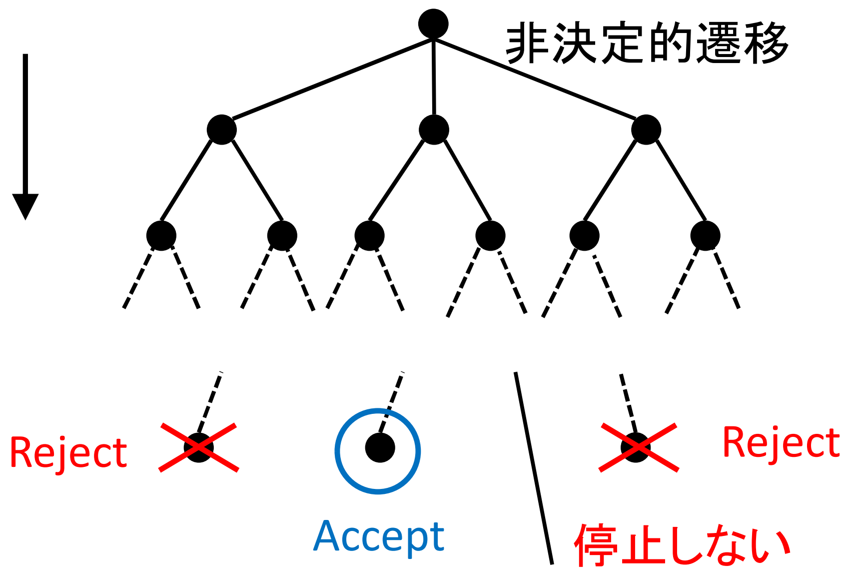


- 1テープを2kトラックに分割した1-テープTMで模倣
  - 元テープ1本の内容の記憶に1トラックを使用
  - 各テープのヘッド位置の記憶に1トラックを使用



非決定性TM

- 非決定的な遷移関数を持つTM
- $\delta: Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$
  - $P(S)$は集合Sのべき集合
- 非決定性TMが受理
⇔受理状態までの選択列が少なくとも一つ存在



非決定性TMと決定性TMの能力

- 非決定性TMの能力 ≥ 決定性TMの能力
  - 自明
- 非決定性TMの能力 ≤ 決定性TMの能力
  - 非決定性TMの動作を決定性TMで模倣できる
    - 幅優先探索で計算木をたどる
- [結論]非決定性TMと決定性TMにおいて，認識できる言語クラスは等価
  - 計算時間は等しいとは限らない

決定問題

決定問題(decision problem)とは?

- Aに対するPの決定問題 (P, A):
  - 各a∈Aに対して，P(a)が真か偽かを決定する問題
  - P : 性質
  - A : あるクラス（要素数が無限の集合）

- 例 :
  - A:正整数，P : 正整数は素数
  - A : グラフ，P:グラフは連結

決定可能性と決定不能性

- 問題(P, A)は決定可能
  ⇔ 決定アルゴリズムが存在する
  - 具体的に決定アルゴリズムを示せばよい

- 問題(P, A)は決定不能
  ⇔ 決定アルゴリズムが存在しない
  - 決定アルゴリズムが存在しないことを証明する

- 決定可能か決定不能かが分かっていない問題も存在する

Church-Turingの提唱

- fが計算できる（決定可能である）
  ⇔ fを計算するTMが存在する

- 右辺によって左辺を定義しようという提唱

決定問題の言語表現

- 決定問題 (P, A) を言語定義を用いて表現

$$L_{(P, A)} = \{a \in \Sigma^* \mid a \in A \land P(a)\}$$

- 問題(P, A)が決定可能
  ⇔ あるTMにより言語$L_{(P, A)}$が判定可能
- ここでの「判定」とは？
  - 任意の入力a∈Aに対して，受理or棄却で停止
    - 停止することが重要

言語の判定可能性と認識可能性

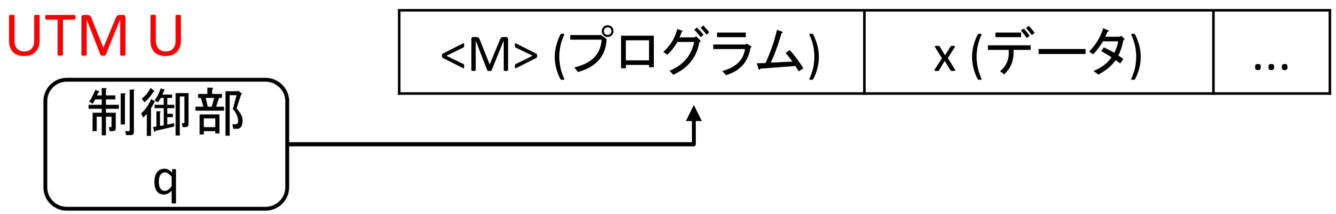
- TMによる言語$L_{(P, A)}$の判定
  - 任意の入力a∈Aに対して，必ず受理または棄却で停止
  - 無限ループには入らない
  - <u>停止の保証がある</u>

- TMによる言語$L_{(P, A)}$の認識
  - 入力$a \in L_{(P, A)}$に対して，受理して停止
  - 入力$a \notin L_{(P, A)}$に対して，棄却または無限ループ
  - <u>停止の保証が無い</u>

決定不能問題

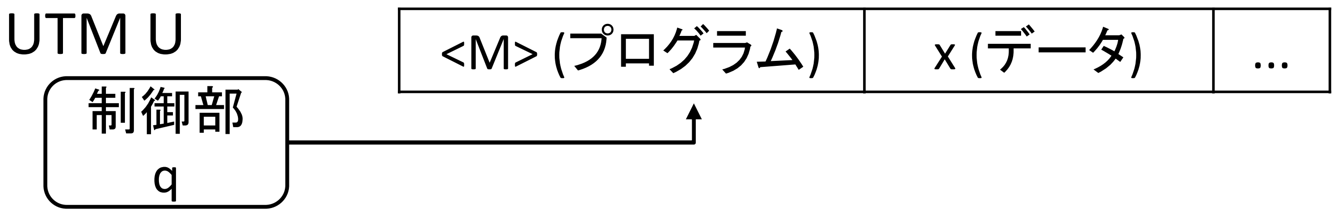
万能TM (Universal TM : UTM)

- 入力
  - <M> : TM Mの動作記述（記号列で表現したもの）
  - x : Mへの入力



- Uの動作 : TM Mにxを与えたときの動作を模倣
  - 万能TM Uは一般的なTMの振る舞いの範囲内

万能TMが対応するもの



- 制御部　→　CPU
- <M>　→　プログラム (バイトコード)
- テープ　→　(作業用) メモリ

- 今までのTMは特定動作専用のもの
  - そのために作られた（他の用途で利用できない）機械
- 万能TMはプログラム格納式計算機（現在のPC）に相当

決定不能な問題

- TMの停止問題 (halting problem)

L_{TM} = {<M, w>| M∈TM∧Mはwを受理}

  - <M, w> : Mとwの組を文字列で表したもの
  - Mはwを受理するか？

- 言語L_{TM}を認識するTM Tは存在
  - 単にMの動作を模倣すればよい
  - Tへの入力は<M,w>
  - 模倣したMがwを受理/棄却すれば，Tは受理/棄却して停止
  - Mが無限ループのときはTも無限ループ

決定不能問題であることの証明 (1/3)

- TMの停止問題 (halting problem)

  L_{TM} = {<M, w>| M∈TM∧Mはwを受理}

  – <M, w> : Mとwの組を文字列で表したもの
  – Mはwを受理するか？

- 背理法で証明
  – L_{TM}を判定するTM Hの存在を仮定して矛盾を示す
  – H(<M, w>) = $\begin{cases} \text{受理}: \text{Mがwを受理するとき} \\ \text{棄却}: \text{Mがwを受理しないとき} \\ \quad (\text{棄却 } or \text{ 無限ループ}) \end{cases}$
  – 「判定」できるということは，必ず停止できるということ

決定不能問題であることの証明 (2/3)

- このとき，次のようなTM Dを構成
  - 入力：<M> (TM Mの文字列表現)
  - 動作：内部でTM Hを動かす
    - Hへの入力： <M, <M>>
    - その後Hの操作を模倣して，受理か棄却が判明
  - 出力: 以下の通りとする

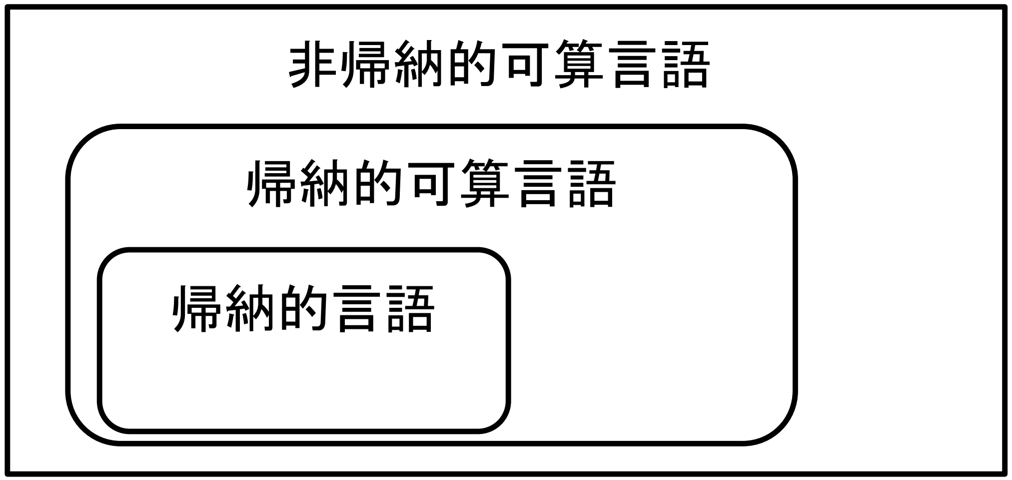
$$
\mathrm{D}(<\mathrm{M}>)=\begin{cases}\text{受理}:\ \mathrm{M}\text{が} < \mathrm{M} > \text{を受理しないとき} \\ \qquad (\text{棄却}\ or\ \text{無限ループ}) \\ \text{棄却}:\ \mathrm{M}\text{が} < \mathrm{M} > \text{を受理するとき}\end{cases}
$$

決定不能問題であることの証明 (3/3)

- ここで，Dに自身の文字列表現<D>を与えると
  - D(<D>) = $\begin{cases} \text{受理} & \text{D}\text{が} < \text{D} > \text{を受理しないとき} \\ & (\text{棄却 } or \text{ 無限ループ}) \\ \text{棄却} & \text{D}\text{が} < \text{D} > \text{を受理するとき} \end{cases}$
  - Dが<D>を受理するとき，内部のHはDが<D>を受理しないと判定している　→ 矛盾
- 結論：L_{TM}を判定するTM Hは存在しない

認識・判定の観点からの言語分類

- 帰納的言語
  - 判定するTMが存在（停止保証あり：決定可能）
- 帰納的可算言語
  - RE (Recursively enumerable language)と呼ばれる
  - 認識するTMが存在（停止保証なし：決定不能）
- 非帰納的可算言語
  - 認識するTMが存在しない（決定不能）



以上です！

計算理論の講義は
これでおわり