計算モデル特論

非情報系学生さん向けの基礎

国立情報学研究所

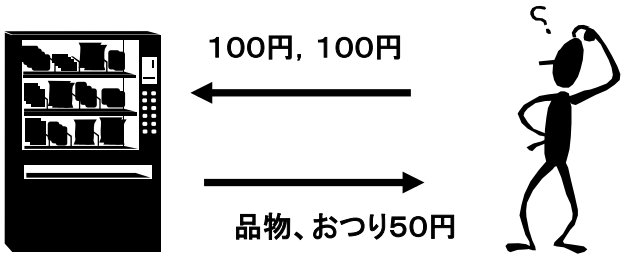
佐藤 一郎

資料ダウンロード

- http://research.nii.ac.jp/~ichiro/lecture/model2012/
- 先週の資料をアップロードできていない
  - サーバ更新と重なりました

オートマトン

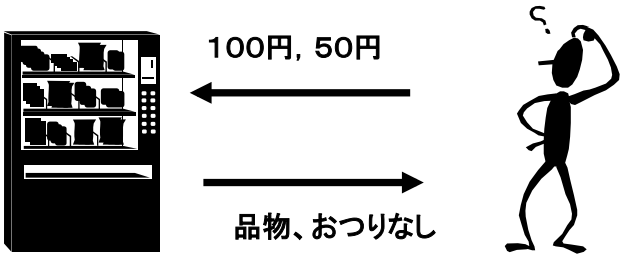
- Automaton: 自動機械



自動販売機
150円の品物を販売

オートマトン

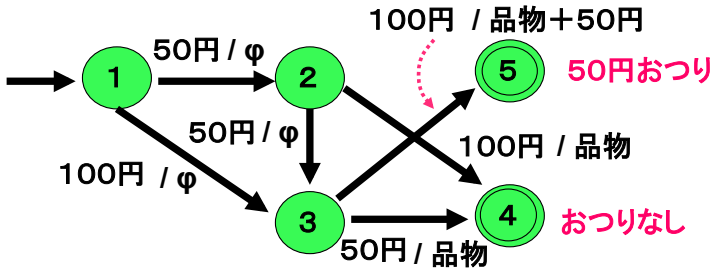
- Automaton: 自動機械



状態機械

- 状態を持つ逐次処理機械

  自動販売機の（例1）は順序機械



準備

Σ ： 有限アルファベット（記号の有限集合）

Σ 上の語 ： Σ の要素の有限列 $w=a_1a_2\ldots a_n\ (a_i \in \Sigma)$

w の長さ $lg(w) = n$

空語 ： 長さ0の語 → εであらわす

$\Sigma^* = \{ w \mid w \text{ は} \Sigma \text{上の語} \}$ （Σのスター閉包）

（例3）

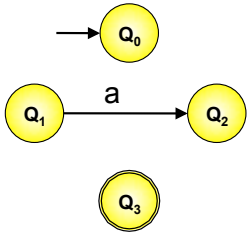
$$\Sigma = \{0, 1\}\ \Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$$

注意

オートマトンの基本

入力に応じて状態遷移する機械

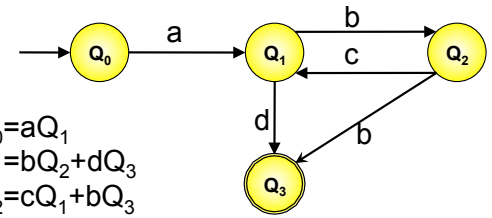
- 初期状態（高々一つ）
- 状態遷移
- 最終状態（0個以上）



オートマトン

手続き型プログラムは状態遷移図によるモデル化が容易

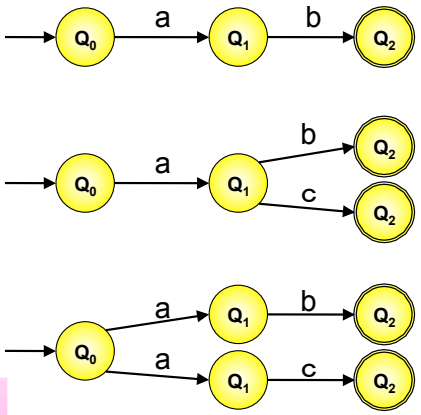
状態機械（オートマトン）は逐次動作



$Q_0 = aQ_1$
$Q_1 = bQ_2 + dQ_3$
$Q_2 = cQ_1 + bQ_3$

$Q_0 = aQ_1$
$Q_1 = b(cQ_1 + dQ_3)$

オートマトンの例



$Q_0=abQ_2$
ab

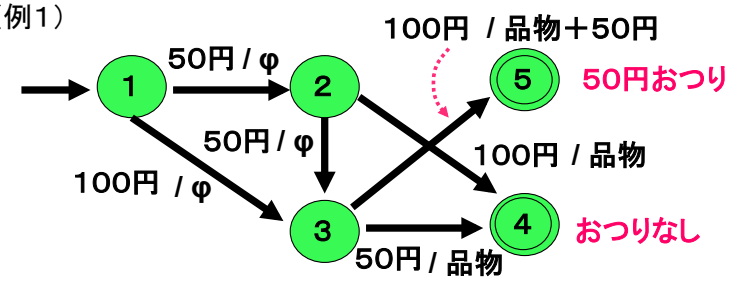
$Q_0=abQ_2+acQ_2$
abまたはac

$Q_0=abQ_2+acQ_2$
abまたはac

Ichiro Satoh

自動販売機のオートマトン

（例1）



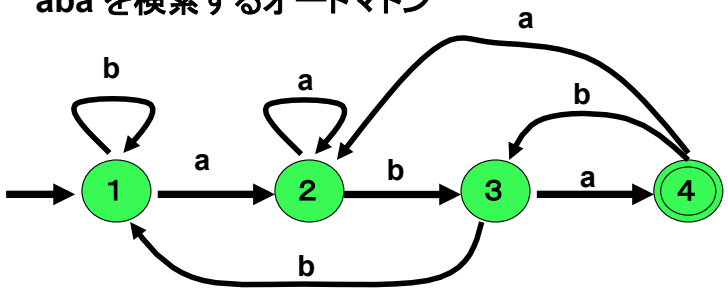
1： 合計0円が入力された状況
2： 合計50円が入力された状況
3： 合計100円が入力された状況
4： 合計150円が入力された状況
5： 合計200円が入力された状況

Ichiro Satoh

文字列検索への応用

（例2）

aba を検索するオートマトン



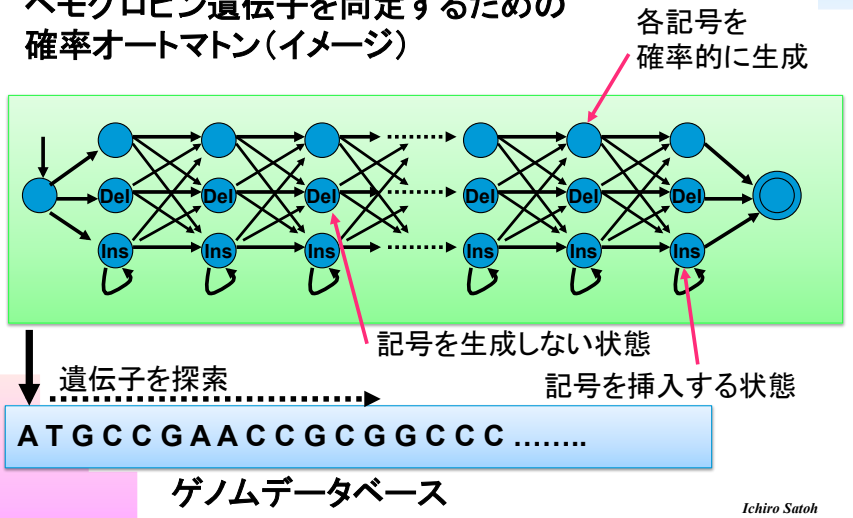
状態4に到達

a a b b a b a b a a b a

Ichiro Satoh

遺伝子配列解析への応用

ヘモグロビン遺伝子を同定するための
確率オートマトン（イメージ）



Ichiro Satoh

形式文法とは

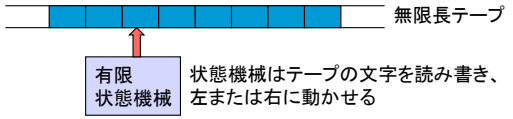
- 言語の規則性を表現するための道具
- 言語として正しい文と間違っている文を判別するための道具

応用範囲

自然言語解析，プログラミング言語，
遺伝子情報解析，など

チューリングマシンと仮想化

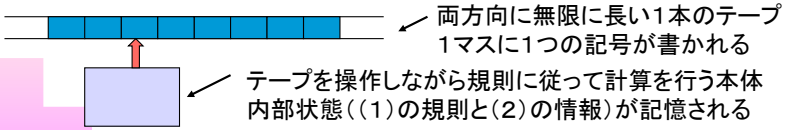
- チューリングマシン(Turing Machine)
  - 計算モデルのひとつで計算機を数学的に議論するための、単純化・理想化された仮想機械である (from Wikipedia)



- 仮想化(Virtualization)
  - コンピュータを構成するプロセッサ、メモリ、入出力などを仮想的に実現する仮想機械
- 抽象機械
  - 実際のハードウェアとは独立した命令をもつ仮想機械(e.g., Java VM)

チューリング機械モデル

- 計算する機械の本質を定義した
  - (1)はじめに計算の手順を一種の規則として記憶する
  - (2)与えられたデータに上の規則を適用し、1ステップづつ計算を行う
    - 1ステップの動作は、データを読み込んで簡単な計算を行い、必要に応じて結果を計算用紙に書きとめたあと、いくらかの必要な情報を記憶して次のステップへ進む。
    - ここで記憶する情報は、前のステップで得られた結果のほか、今どの段階の計算をしているかの情報なども含む。
- 具体的な機械のイメージで実現

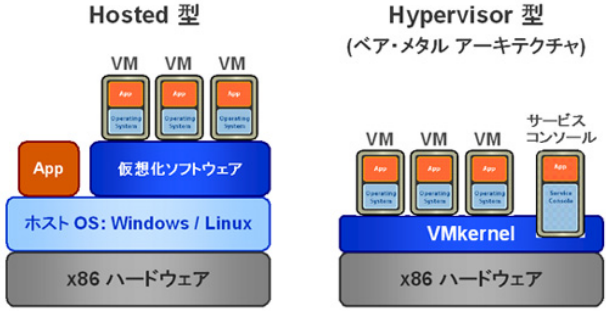


仮想化

- 仮想マシン
  - 物理コンピュータ上で、コンピュータを再現するソフトウェア

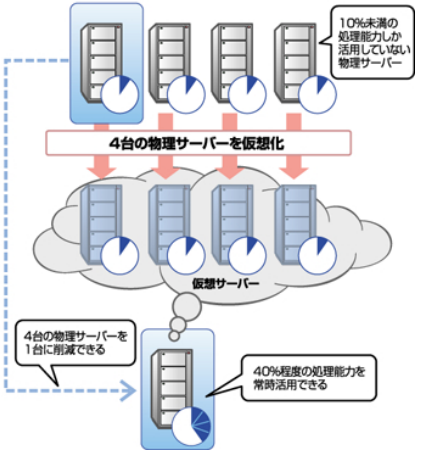
仮想化手法

- ホスト型
  - 既存OS上で仮想マシンを動作
- ハイパーバイザ型
  - 専用モニタ(仮想化のための簡易OS)上で仮想マシンを動作



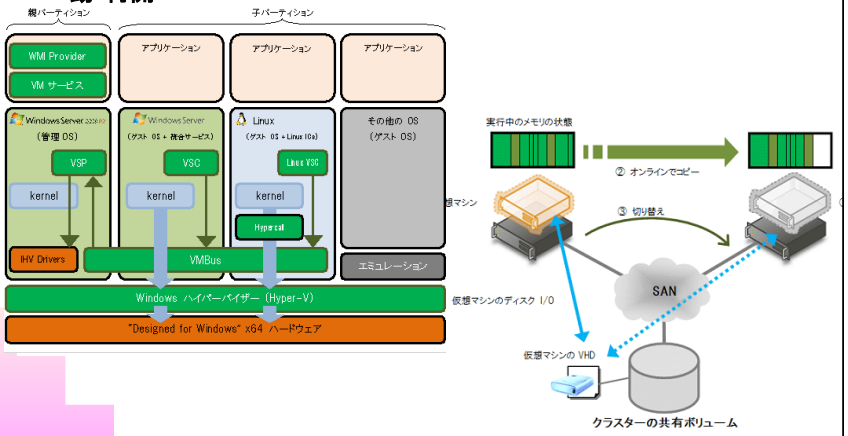
仮想化が必要な理由

- 企業におけるサーバの稼働率は15%
  - 85%は遊んでいる
- コンピュータの台数を減らせる(コストが下がる)
  - 仮想化により一台の物理サーバで複数のサーバを動かす
- 会計制度の変更(国際会計基準(IFRS))
  - 企業資産におけるIT資産の比率の向上



仮想化システム

- 仮想マシンを事実上ダウンタイムゼロで他の物理コンピュータに移動・再開

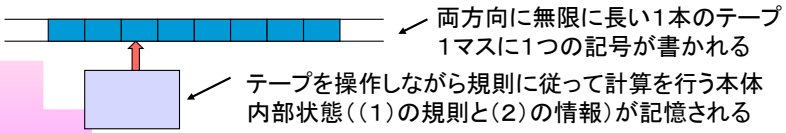


チューリングマシン

- A. M. チューリングが提案した仮想機械
  - すべてのプログラム内蔵計算機をシミュレーションできる機械
  - 基本的な動作(特定の作業)
    - テープの読み込み
    - テープの書き込み
    - テープの送り、戻し
    - 内部状態の変更
  - 記憶領域
    - 内部記憶
    - テープ(外部記憶)

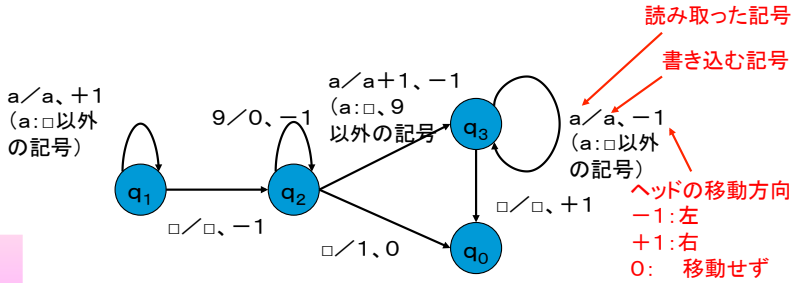
チューリング機械モデル

- 計算する機械の本質を定義した
  - (1)はじめに計算の手順を一種の規則として記憶する
  - (2)与えられたデータに上の規則を適用し、1ステップづつ計算を行う
    - 1ステップの動作は、データを読み込んで簡単な計算を行い、必要に応じて結果を計算用紙に書きとめたあと、いくらかの必要な情報を記憶して次のステップへ進む。
    - ここで記憶する情報は、前のステップで得られた結果のほか、今どの段階の計算をしているかの情報なども含む。
- 具体的な機械のイメージで実現



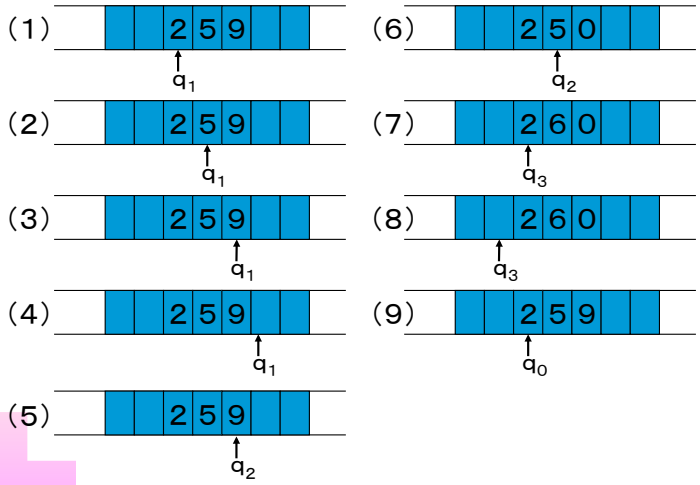
チューリング機械の例

- 具体的なチューリング機械
  - 例:自然数kの10進表現を与えられたとき、k+1の10進表現を得る機械M
  - Mのテープ記号の集合A={□、0、1、. . 9}
  - Mの内部状態の集合Q={q_0、q_1、q_2、q_3}
  - Mの規則を表す関数T(以下の遷移図)



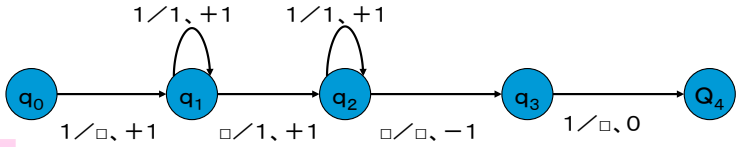
チューリング機械モデルの計算

- 具体的なチューリング機械での計算ステップ



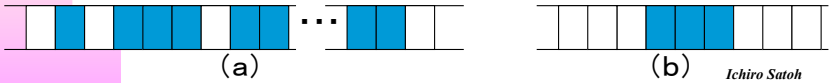
チューリング機械の例(2)

- 足し算をするチューリング機械M
  - 自然数kをk+1個の「1」の並びで表現する。0は1個の「1」で表現する。入力の2つの数値はブランク(□)で区切られて与えられる。結果はテープ上の1の数。
  - Mのテープ記号の集合A={□、1}
  - Mの内部状態の集合Q={q_0、q_1、q_2、q_3、q_4}
  - Mの規則を表す関数T(以下の遷移図)
- 引き算をする機械はどうやって定義するか？



チューリング機械による自然数関数の計算

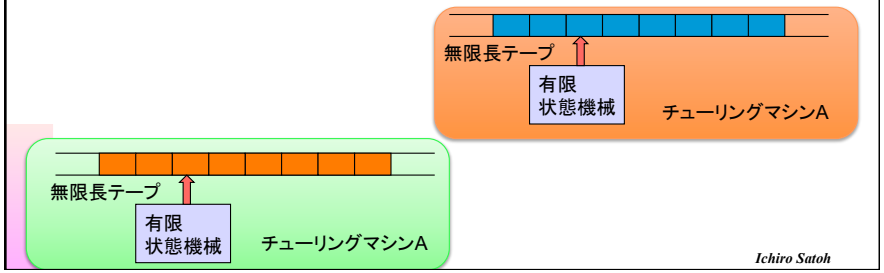
- テープ記号に□、0、1、…、9を含むチューリング機械Mと、関数f:Nn→Nについて考える。
- 自然数m1、m2、…、mnに対し、それらの10進表現を下図(a)のように□をはさんで並べ、□でない最も左のマスにMのヘッドをセットする。
- 初期状態q1から規則に従ってMが計算を行った結果、下図(b)のようにf(m1, m2, …, mn)＝mの10進表現をテープに記録してその左端にヘッドを置いて停止するとする。
- このことがすべての自然数について成り立つとき、関数fはチューリング機械Mで計算されるという。
- NnからNへの部分関数fがチューリング機械で計算されるならば、fは帰納的関数である。また、その逆も成り立つ。



Ichiro Satoh

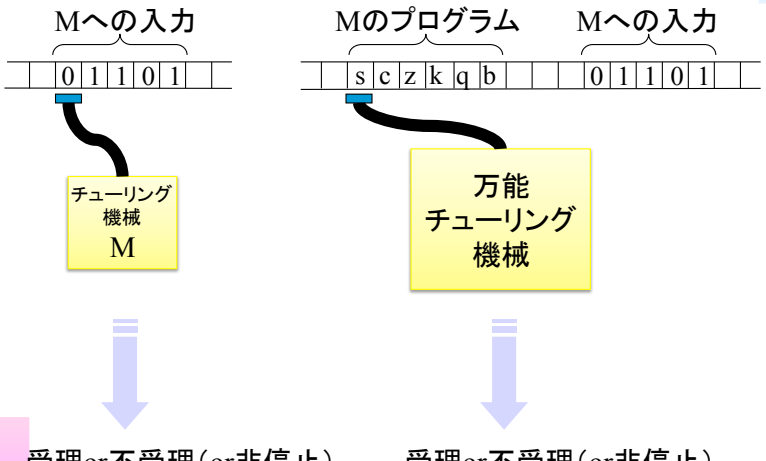
万能チューリングマシン

- 万能チューリングマシンとはチューリングマシンを模倣するチューリングマシン
  - チューリングマシンAの動作を再現するテープ(エンコーディング)
  - チューリングマシンBでそのテープを実行するとチューリングマシンAとして振る舞う



Ichiro Satoh

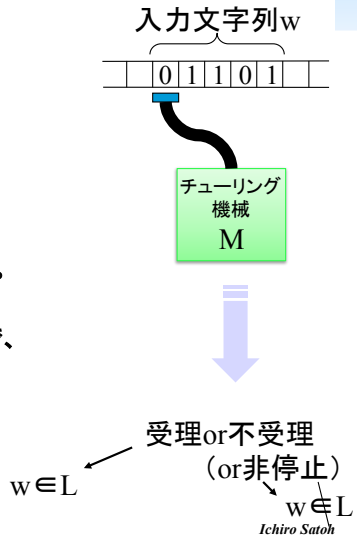
万能チューリング機械



Ichiro Satoh

言語の認識装置として

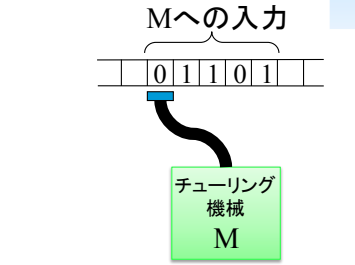
- 入力アルファベット上の言語Lが、
  - チューリング機械Mによって
  - 認識できるとき、その言語を
  - 帰納的に可算という。
- Mを用いれば、与えられた文字列が
  - Lに属しているとき、
  - 有限時間内にそのことがわかる。
  - Lに属していないときは、
  - Mは止まらないかもしれないので、
  - そのことはいつまでたっても
  - わからないかもしれない。
- 以上のような認識が、
- 計算の限界である。
  - チャーチのテーゼ

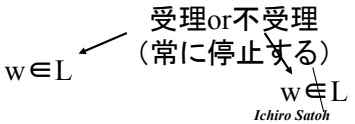


Ichiro Satoh

帰納的に可算 vs. 帰納的

- 入力アルファベット上の言語Lが、
  - 常に停止するチューリング機械
  - Mによって認識できるとき、
  - その言語を帰納的いう。
- Mを用いれば、与えられた文字列が
  - Lに属しているとき、
  - 有限時間内にそのことがわかる。
  - Lに属していないときも、
  - 有限時間内にそのことがわかる。
- 言語Lとその補集合がどちらも
- 帰納的に可算ならば、
- L(およびその補集合)は
- 帰納的になる。



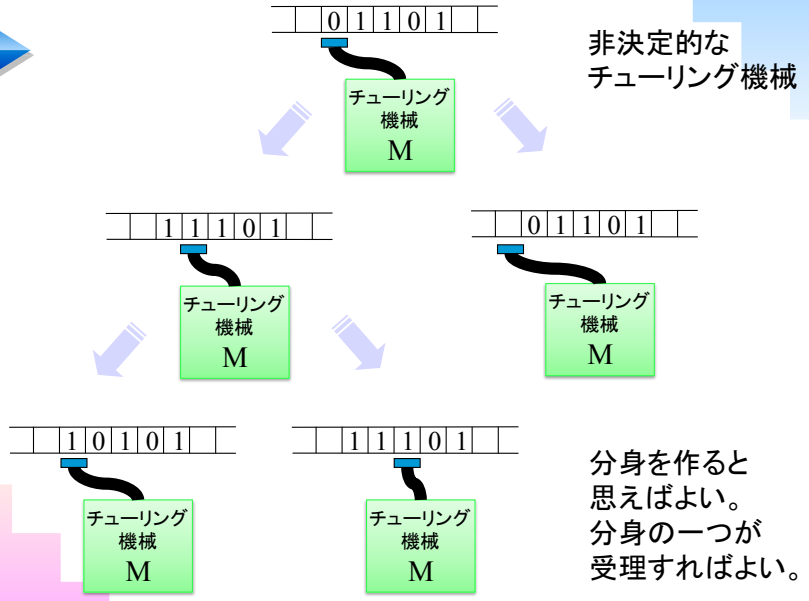


関数の実現機構として

- チューリング機械を用いて、
  - 文字列をもらって文字列を返す関数を実現できる。
  - 入力文字列をテープ上に書き込み、
  - チューリング機械を走らせる。
  - 停止したら、テープ上の文字列を返す。
    - 空白は無視する。
- チューリング機械は止まらないかもしれないので、
  - いつも出力が得られるとは限らない。
  - 部分関数
- このような関数を部分帰納的関数という。
- 常に出力が得られるときは、帰納的関数という。
  - チューリング機械が常に止まる場合。

決定性と非決定性

- 計算可能性に関して(言語の認識装置として)、
  - 決定的なチューリング機械と
  - 非決定的なチューリング機械に
  - 能力の差はない。
    - 非決定的なチューリング機械が入力文字列を
      受理するとは、入力文字列をテープ上に書き込み、
      非決定的な動作を適切に選べば、
      (受理状態で)停止すること。
    - 任意の非決定的なチューリング機械に対して、
      同じ言語を受理する決定的なチューリング機械が
      存在する。



万能チューリング機械と停止問題

- 万能チューリング機械は、
  - 任意のチューリング機械Mのプログラムと
  - Mへの入力に対して、Mをシミュレートできる。
    - Mが停止しないときは、
    - 万能チューリング機械も停止しない。
- Mが停止しないときも停止して、
  - Mの非停止を判定できるような
  - 万能チューリング機械は存在するか。
    - プログラムが停止しないことを、
    - プログラムを実行せずに判定できるか？
      - 答えは否。

チューリング機械の停止

- 停止状態を定めることもある。
- 次の動作が定義されなかったら停止。
- 場合によっては、停止しないこともある。