I113 オートマトンと形式言語
(Automata and Formal Languages)
基本情報と
講義(1): 導入と数学的準備

平成18年度

担当: 上原 隆平(Ryuhei UEHARA)
uehara@jaist.ac.jp
http://www.jaist.ac.jp/~uehara/

I113 オートマトン
(Automata and Form
基本情報

年度

uhei UEHARA)

ist.ac.jp

c.jp/~uehara/

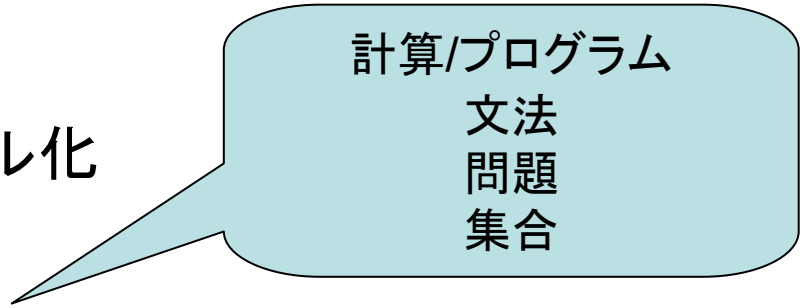
オートマトンと言語理論

- 講義: 14回
- 成績:
  - プレースメントテスト(4月12日(水)13:30～15:00)
  - レポート(50%)＋期末試験(50%)
- サポートページも忘れずにチェック
  - http://www.jaist.ac.jp/~uehara/course/2006/i113/
  - プレースメントテストの問題例
  - 講義で使ったPowerPointのファイル
- テキスト

  「オートマトン 言語理論 計算論 I」[第2版]
  J. ホップクロフト/R. モトワニ/J. ウルマン 著
  野崎昭弘/高橋正子/町田元/山崎秀記 訳

0. はじめに

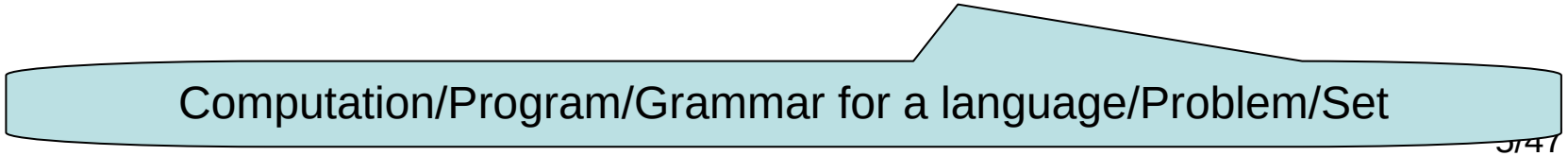
- オートマトンと形式言語
  - 1940～1950年代に独立に考案、研究された
  - オートマトンとチューリングマシン: 機械をモデル化
    - due to A. Turing
  - 形式言語: 言語の文法をモデル化
    - due to N. Chomsky



- オートマトンとチューリングマシン = 形式言語
  - 同じ概念の違った形

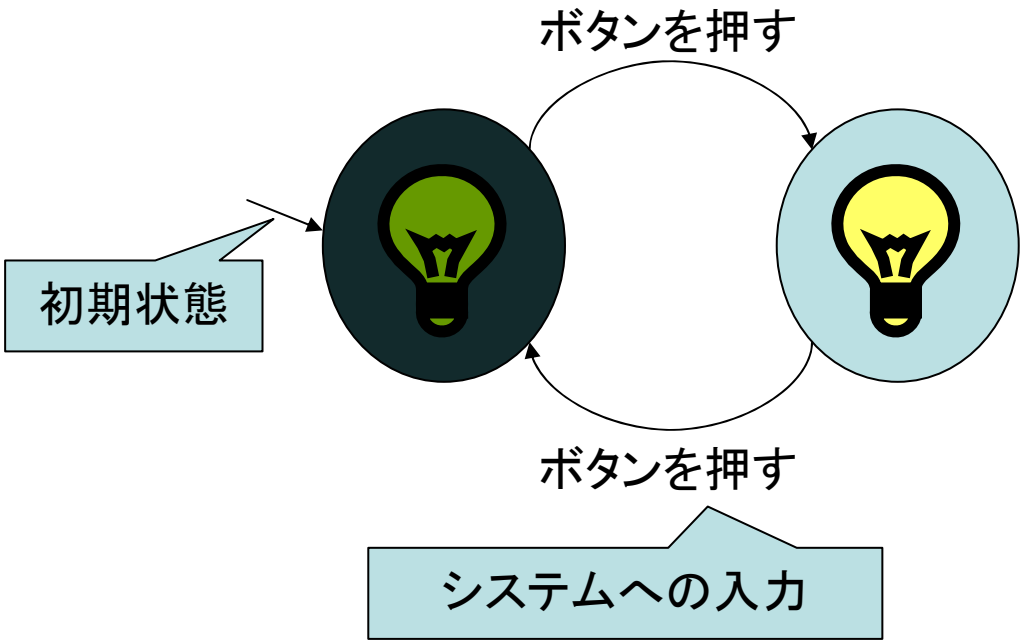
0. Introduction

- Automata and Formal Languages
  - were independently stated and investigated in 1940～1950
  - Automata and Turing machines model machines
    - due to A. Turing
  - Formal Languages models grammars of languages
    - due to N. Chomsky
- Automata and Turing machines = Formal Languages
  - Different forms for the same concept



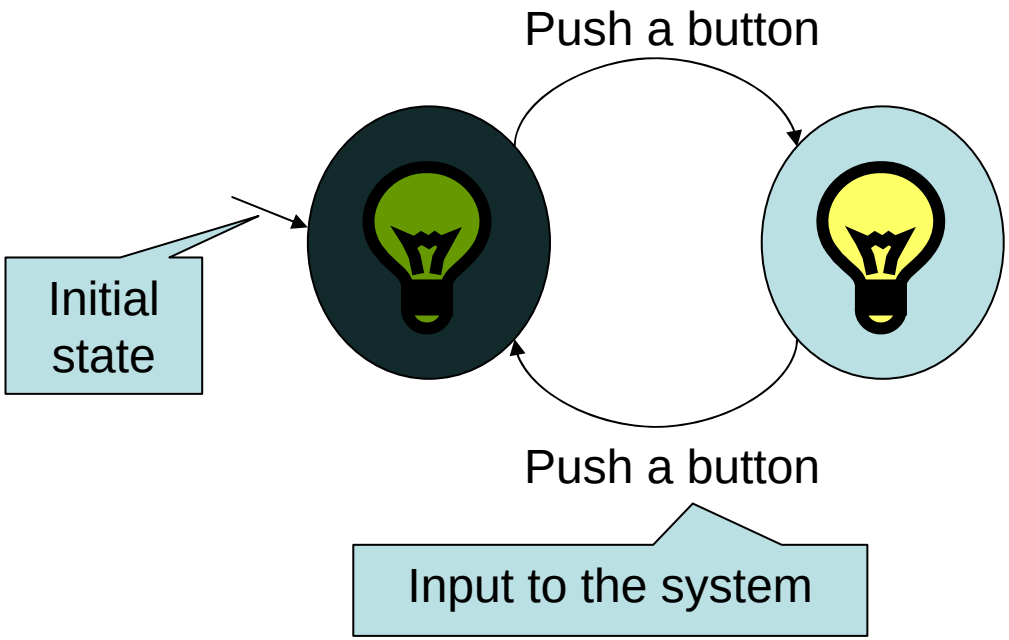
0. はじめに

- 機械をモデル化したものとしてのオートマトン
  - 「状態」と「入(出)力」をもつ機械モデル



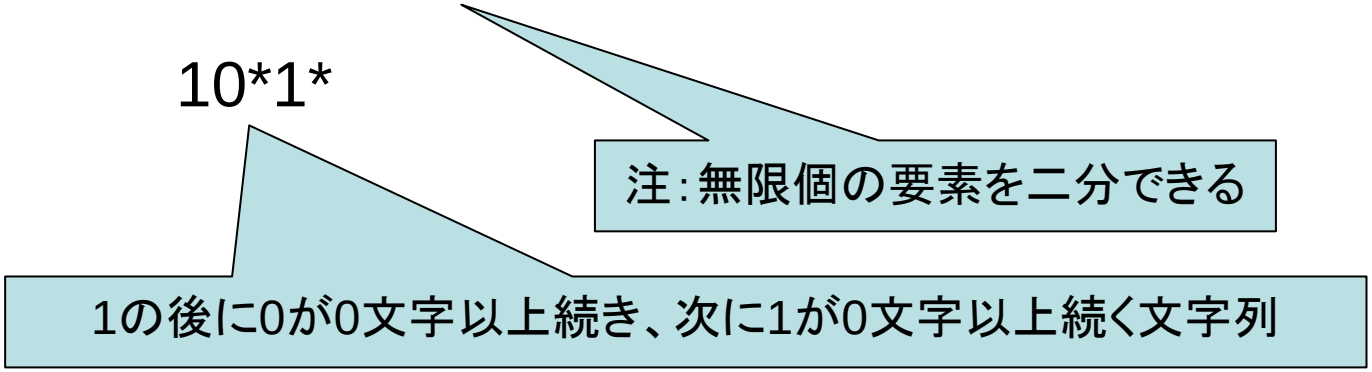
0. Introduction

- Automaton as a model of a machine
  - machine model with ‘states’ and ‘in(out)put’



0. はじめに

- 言語の文法をモデル化したものとしてのオートマトン
  - 「文字列の集合」を記述するための規則

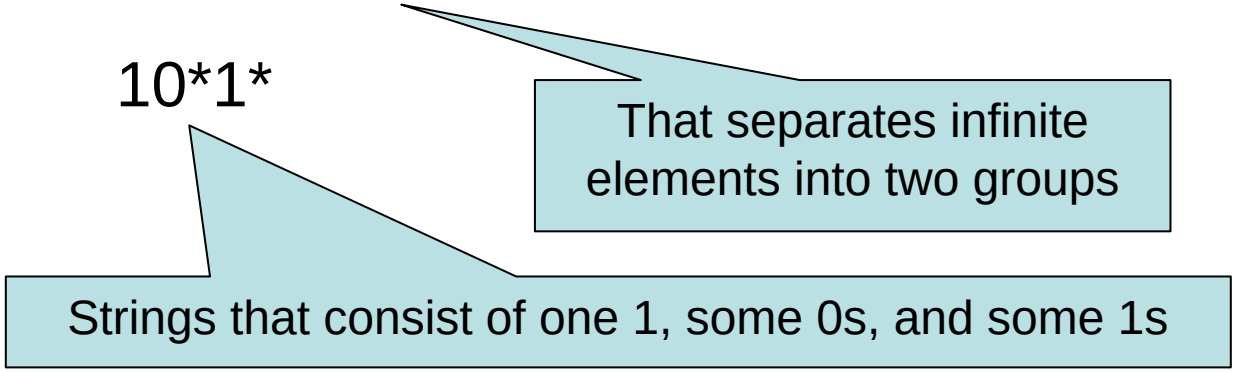


○ 101, 100111, 1, 10, 11, 100001111111, …

× 00, 1010, 1110, 0101, 10001110, …

0. Introduction

- Automaton as a model of a grammar of a language
  - Rule for description of a set of strings



- ○ 101, 100111, 1, 10, 11, 100001111111, …
- × 00, 1010, 1111, 0101, 10001110, …

0. はじめに

- オートマトン = 形式言語
  - コンピュータサイエンスの基礎理論
    - 言語処理
      - 自然言語処理, プログラミング言語, コンパイラ, …
    - ハードウェア
      - 機械, ロボット, …
    - その他
      - ネットワークプロトコル, …

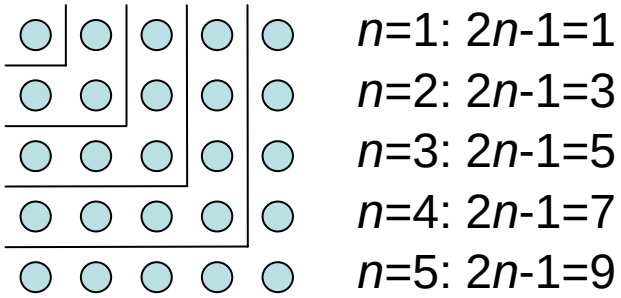
0. Introduction

- Automaton = Formal language
  - which is a basic theory of computer science
    - Language processing
      - Natural languages, Programming languages, Compiler, …
    - Hardware
      - Machine, Robots, …
    - and any systems
      - Network protocol, …

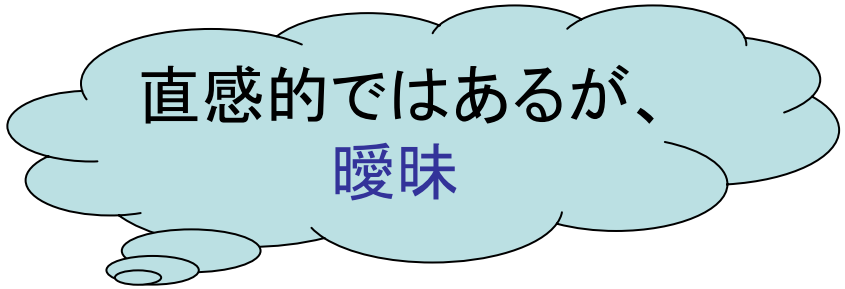
1. 数学的準備: 形式的証明

- 「図示」と「形式的な記述」
  - 図示…直感的だが、不正確・曖昧性
  - 形式的記述…正確であり、曖昧性を排除できる

例: $1+3+5+7+\ldots+(2n-1)=n^2$ を証明せよ



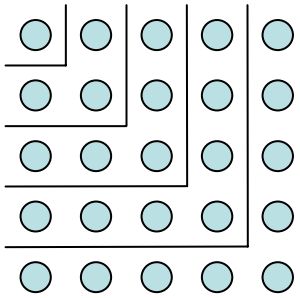
n=1: $2n$-1=1
n=2: $2n$-1=3
n=3: $2n$-1=5
n=4: $2n$-1=7
n=5: $2n$-1=9



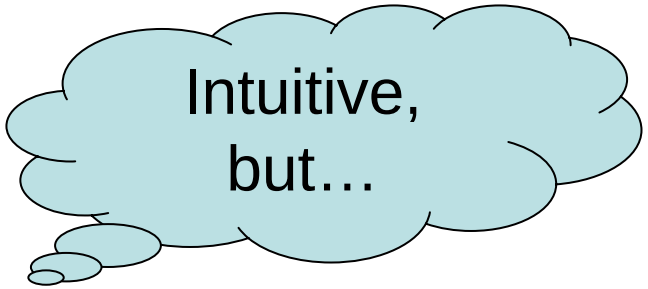
1. Mathematical preliminaries: Formal proof

- ‘Figure’ and ‘formal description’
  - Figure…Intuitive, but ambiguity
  - Formal description…Correctness, and not ambiguity

Example: Prove $1+3+5+7+\ldots+(2n-1)=n^2$



n=1: $2n$-1=1
n=2: $2n$-1=3
n=3: $2n$-1=5
n=4: $2n$-1=7
n=5: $2n$-1=9



1. 数学的準備: 証明技法

- この授業で使う証明技法
  - 演繹的証明
  - 背理法
  - 数学的帰納法
  - 対角線論法は範囲外

有限個の文字で無限個の場合を扱う方法はそれほど多くない

1. Mathematical preliminaries: Proof techniques

- Proof technique in this class
  - Deductive proof
  - Proof by contradiction
  - Induction
  - (Diagonal method is out of this class)

There are a few techniques to deal with infinitely many cases with finite words.

1. 数学的準備: オートマトン理論の基礎概念

- アルファベット(alphabet): 記号の空でない有限集合
  - Σ ＝{0,1}
  - Σ ＝{a,b,c,d,…,x,y,z}
  - Σ ＝{あ, い, …, を, ん} など
- 文字列(string)(または語(word)): アルファベットから選んだ記号の有限個の列
  - 01011, 000, alphabet, うえはら など
  - 0個の記号の列を空列といい、ε であらわす。
  - 文字列に含まれる記号の個数を、長さという。

    |01011|=5, |000|=3, |alphabet|=8, | ε |=0

1. Mathematical preliminaries: Basic notions for automaton

- Alphabet: Nonempty finite set of symbols
  - $\Sigma = \{0,1\}$
  - $\Sigma = \{a,b,c,d,\ldots,x,y,z\}$
  - $\Sigma =$ {あ, い, …, を, ん} etc.
- String (or Word): Finite sequence of symbols in the alphabet
  - 01011, 000, alphabet, うえはら etc.
  - The string consists of 0 symbol is said 'empty string,' which is denoted by ε
  - The length of a string is defined by the number of symbols

    $|01011|=5$, $|000|=3$, $|\text{alphabet}|=8$, $|\varepsilon|=0$

1. 数学的準備: オートマトン理論の基礎概念

- Σ^k: アルファベット Σ の長さ k の語の集合

  例; $\{0,1\}^3=\{000,001,010,011,100,101,110,111\}$

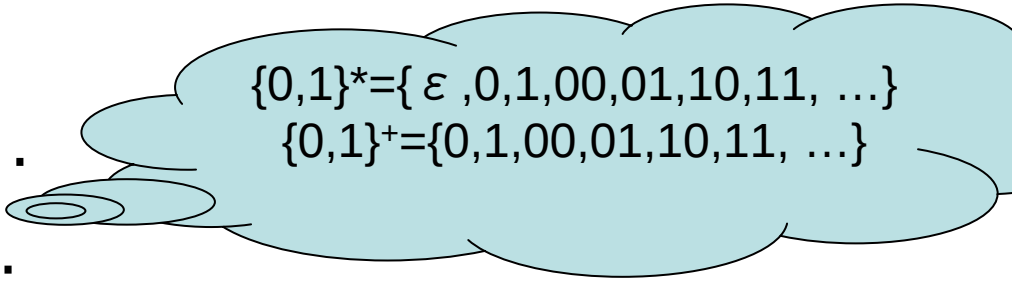
  $\Sigma^0=\{\varepsilon\}$

  注; $\Sigma=\{0,1\}$と$\Sigma^1=\{0,1\}$は見た目は同じだが意味が違う。

  $\Sigma=\{0,1\}$ はアルファベット、あるいは文字の集合

  $\Sigma^1=\{0,1\}$は長さ1の文字列の集合

- $\Sigma^*=\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- $\Sigma^+=\Sigma^1 \cup \Sigma^2 \cup \ldots$



1. Mathematical preliminaries: Basic notions for automaton

- Σ^k: The set of words of length k over the alphabet Σ

  Example; $\{0,1\}^3=\{000,001,010,011,100,101,110,111\}$

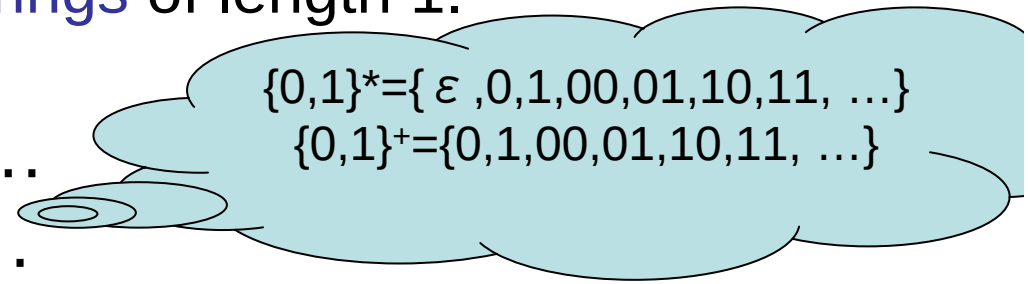
  $\Sigma^0=\{\varepsilon\}$

  C.f.; $\Sigma=\{0,1\}$ and $\Sigma^1=\{0,1\}$ look the same, but have different meanings

  $\Sigma=\{0,1\}$ is alphabet, or a set of letters (characters).

  $\Sigma^1=\{0,1\}$ is a set of strings of length 1.

- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$



1. 数学的準備: オートマトン理論の基礎概念

- 文字列の連接(concatenation):

  二つの文字列 $x=a_1a_2a_3a_4...a_i$ と $y=b_1b_2b_3...b_j$ に対し、$a_1a_2a_3a_4...a_ib_1b_2b_3...b_j$ を文字列 x と y の連接といい、xy と書く。

1. Mathematical preliminaries: Basic notions for automaton

- Concatenation of two (or more) strings:

  For any two strings $x=a_1a_2a_3a_4...a_i$ and $y=b_1b_2b_3\ldots b_j$, the string $a_1a_2a_3a_4...a_ib_1b_2b_3\ldots b_j$ is 'concatenation' of x and y, and denoted by xy.

1.　数学的準備:
オートマトン理論の基礎概念

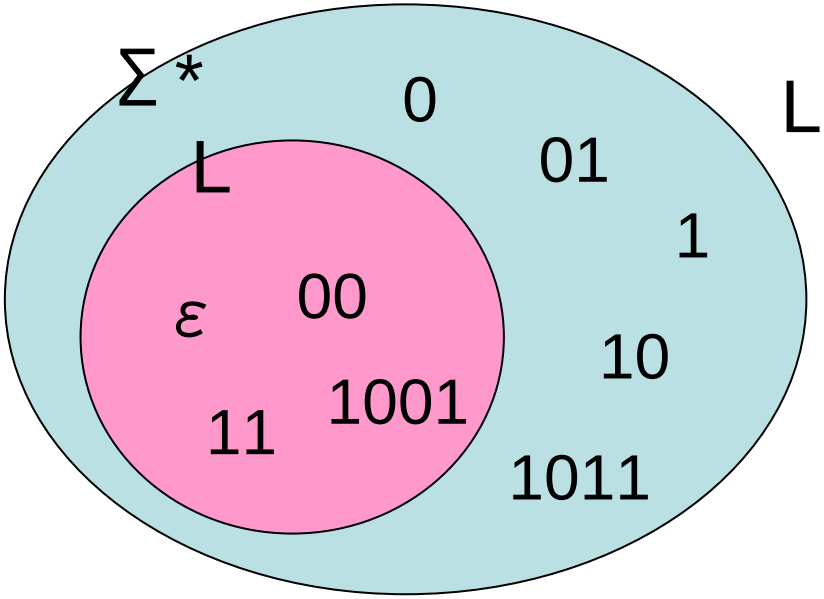
- 言語(Language):

  アルファベットΣに対し、

  $L \subseteq \Sigma^*$

  を満たす集合LをΣ上の言語という。

言語とは、文法的に正しい文字列の集合



L = { x | x に含まれる
0と1 の個数は等しい}

Lに含まれる文字列も
含まれない文字列も
無限にある。

1. Mathematical preliminaries: Basic notions for automaton

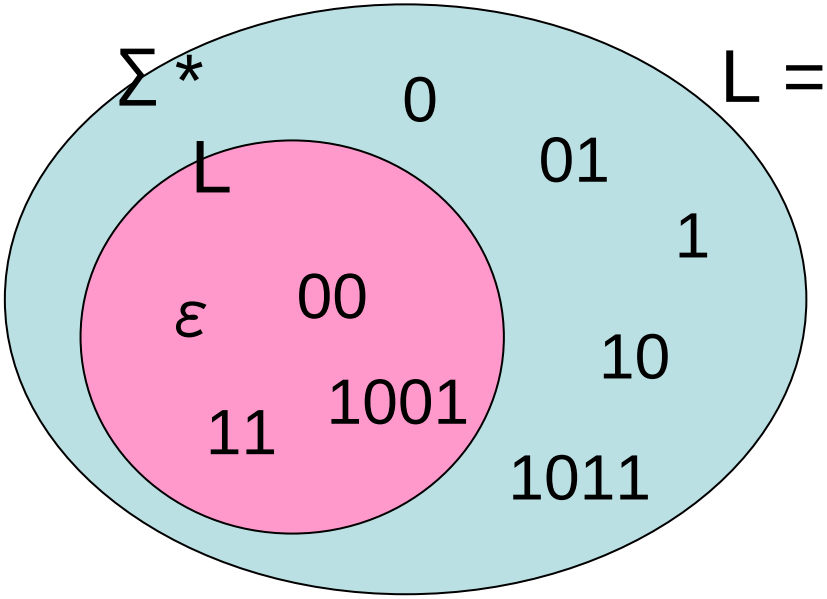
- Language:

  For any alphabet Σ, a

  $L \subseteq \Sigma^*$

  is a 'language' over Σ.

Language is a set of strings which are grammatically correct



L = { x | x contains the same number of 0s and 1s.}

There are infinitely many strings in L and not in L

1. 数学的準備: オートマトン理論の基礎概念

- 言語(Language):

  アルファベット Σ に対し、

  $L \subseteq \Sigma^*$

  を満たす集合 L を Σ 上の言語という。

$L = \{ x \mid x$ に含まれる
0と1 の個数は等しい$\}$

- オートマトン理論＝言語理論
  - 同じ「概念」に対する違ったかたち

機械による計算(プログラム)
言語の文法
問題
集合

1. Mathematical preliminaries: Basic notions for automaton

- Language:

  For any alphabet Σ, a set L with

  $L \subseteq \Sigma^*$

  is a 'language' over Σ.

  L = { x | x contains the same number of 0s and 1s.}

- Automaton＝Formal language
  - The different forms to the same concepts

Computation/Program
Grammar for a language
Problem
Set

1. 数学的準備: 証明技法

- この授業であつかう証明技法
  - 演繹的証明
  - 背理法
  - 数学的帰納法
  - (対角線論法は範囲外)

有限個の言葉で無限の場合を扱う手法はそれほど多くはない

1. Mathematical preliminaries: Proof techniques

- Proof technique in this class
  - Deductive proof
  - Proof by contradiction
  - Induction
  - (Diagonal method is out of this class)

There are a few techniques to deal with infinitely many cases with finite words.

1. 数学的準備: 証明技法

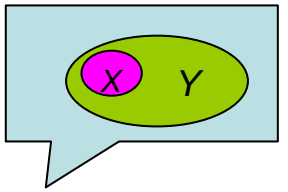
- 演繹的証明

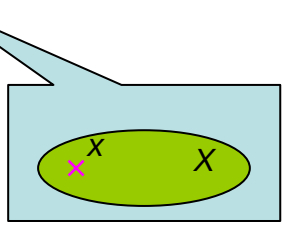
演繹:

一般的な命題から特殊な命題を
抽象的な命題から具体的な命題を
論理的に導き出すこと。例: 3段論法、同値性の証明

「XとYが同値」であることを示すために、
「XならばY」と「YならばX」を示す方法

1. 定義に立ちもどった証明
2. 集合の同一性の証明
   - 集合X,Yに対して、X=Yを証明するのに「X⊆YかつY⊆X」を示す。
   - さらにいえば、例えば前者は「どんな $x \in X$ に対しても $x \in Y$」を示す。
3. 対偶を使用した証明





1. Mathematical preliminaries: Proof techniques

- Deductive proof

Showing X→Y and Y→X to prove X=Y

Deduction: we have

special statement from general statement

concrete statement from abstract statement

by some accepted logical principle.

Example: Syllogism (A→B and B→C imply A→C), Proof of equivalences

1. Reduction to Definitions
2. Proving equivalences about sets
   - For two sets X and Y, showing $X \subseteq Y$ and $Y \subseteq X$ to prove X=Y.
   - (To show $X \subseteq Y$, we prove that $x \in Y$ for any $x \in X$.)
3. Contrapositive

1. 数学的準備: 証明技法

- 背理法
  - AならばBを証明するのに、
    『「AなのにBでない」と仮定すると矛盾が生じる』
    ことを示す手法。

素数: 1と自分自身でしか割り切れない

「素数は無限にある」ことの証明:

- 『素数が n 個しかない』と仮定する。すると、$p_1, p_2, \ldots, p_n$ と小さいほうから番号をつけることができる。
- 数 $p = p_1 \times p_2 \times \ldots \times p_n + 1$ を考える。
  - p が素数であると仮定すると、$p > p_n$ なので、p_nの最大性に矛盾する。
  - p が素数でないと仮定すると、$p_1, p_2, \ldots, p_n$ のどれかで割り切れるはず。しかしpはどの素数で割っても1あまり、矛盾する。
- したがって『素数が n 個しかない』という仮定は誤り。

1. Mathematical preliminaries: Proof techniques

- Proof by contradiction
  - To prove ‘if A then B’, showing
    - “ ‘A and not B’ implies falsehood”

Prime: Only 1 and itself divide it

Proof of ‘primes are infinitely many’:

- We assume that ‘primes are finitely’. Then we can order them $p_1<p_2<\ldots<p_n$ for some finite n.
- Define $p = p_1 \times p_2 \times \ldots \times p_n+1$.
  - If p is prime, $p>p_n$ contradicts the maximality of p_n.
  - If p is not prime, at least one of $p_1,p_2,\ldots,p_n$ divides p. However, by definition of p, we have the surplus 1 for any prime p_i, which is a contradiction.
- Hence the assumption ‘primes are finitely’ is not correct.

1. 数学的準備: 証明技法

- 数学的帰納法
  - オートマトンなど無限の場合を扱うには必須
  - 再帰呼び出しを含むプログラムの正当性の証明などにも深い関連がある
- 基本形(『命題S(*i*)の正しさ』を示すとき)
  - 基礎ステップ: 有限個の例(S(0),S(1)など)に対して正当性を示す
  - 帰納的ステップ: 『S(*i*)が正しければS(*i*+1)も正しい』ことを示す

1. Mathematical preliminaries: Proof techniques

- Induction
  - It is crucial to deal with infinitely many cases like automaton
  - It deeply relates to the proof of the correctness of a program that contains recursive call
- Typical case (prove ‘correctness of a proposition S(*i*)’)
  - Base step: show the correctness for small cases (e.g., S(0), S(1))
  - Inductive step: show ‘S(*i*+1) holds if S(*i*) holds’

1. 数学的準備: 証明技法

- 数学的帰納法(例)
  - S(n): 『どんなnに対しても$1+3+\ldots+(2n-1)=n^2$』

可能なすべての n についてS(n)をチェックすることはできない

1. 基礎ステップ: S(1) の正しさを示す。
2. 帰納的ステップ: 『S(i)が正しければS(i+1)も正しい』ことを示す

1. Mathematical preliminaries: Proof techniques

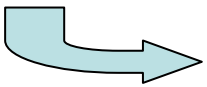
- Induction (example)
  - S(n): 〚$1+3+\ldots+(2n-1)=n^2$ for any n〛

It is impossible to check S(n) for all possible 'n's

1. Base step: Show the correctness of S(1)
2. Inductive step: Show that 'S(i+1) holds if S(i) holds'.

1. 数学的準備: 証明技法

- 数学的帰納法(例)
  - S(n): 『どんなnに対しても$1+3+\ldots+(2n-1)=n^2$』
    1. 基礎ステップ: S(1) の正しさを示す。
    2. 帰納的ステップ: 『S(i)が正しければS(i+1)も正しい』ことを示す
  - もし1,2が正しければ、、、
    - ① 1.より、S(1)は正しい
    - ② ①+2. より、S(2)は正しい
    - ③ ②+2. より、S(3)は正しい
    - ④ …



どんな自然数 n に対しても S(n) は正しい

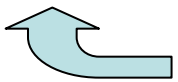
1. Mathematical preliminaries: Proof techniques

- Induction (example)
  - S(n): 〚$1+3+\ldots+(2n-1)=n^2$ for any n〛
    1. Base step: Show the correctness of S(1)
    2. Inductive step: Show that 'S(i+1) holds if S(i) holds'.
  - If 1 and 2 are correct, …
    ① Since 1, S(1) holds,
    ② Since ①+2, S(2) holds,
    ③ Since ②+2, S(3) holds,
    ④ …

    ⇒ S(n) holds for every n

1. 数学的準備: 証明技法

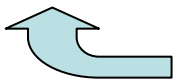
- 数学的帰納法(例)
  - S(n): 『どんなnに対しても$1+3+\ldots+(2n-1)=n^2$』
    1. 基礎ステップ: S(1) の正しさを示す。
    2. 帰納的ステップ: 『S(i)が正しければS(i+1)も正しい』ことを示す
  - ある自然数 n に注目したとき、
    - ① 2. より S(n-1) が正しければ S(n) は正しい
    - ② 2. より S(n-2) が正しければ S(n-1) は正しい
    - ③ …
    - ④ 2. より S(1) が正しければ S(2) は正しい



1.より、S(1) は正しい

1. Mathematical preliminaries: Proof techniques

- Induction (example)
  - S(n): 〚$1+3+\ldots+(2n-1)=n^2$ for any n〛
    1. Base step: Show the correctness of S(1)
    2. Inductive step: Show that 'S(i+1) holds if S(i) holds'.
  - If 1 and 2 are correct, …
    ① Since 2, we have S(n) if S(n-1) is correct,
    ② Since 2, we have S(n-1) if S(n-2) is correct,
    ③ …
    ④ Since 2, we have S(2) if S(1) is correct.



Since 1, we have S(1).

1. 数学的準備: 証明技法

- 数学的帰納法(例)
  - S(n):『どんなnに対しても$1+3+\ldots+(2n-1)=n^2$』
    1. 基礎ステップ: S(1) の正しさを示す。
       $n=1$ のとき、明らかに $1 = 1^2$ なので正しい
    2. 帰納的ステップ:『S(i)が正しければS(i+1)も正しい』ことを示す
       仮定: $n=i$ のときに $1+2+\ldots+(2i-1) = i^2$
       示すこと: $1+2+\ldots+(2i-1)+(2(i+1)-1) = (i+1)^2$
       $1+2+\ldots+(2i-1)+(2(i+1)-1)=1+2+\ldots+(2i-1)+(2i+1)$
       $=i^2 + 2i +1$ (帰納法の仮定より)
       $=(i+1)^2$

「仮定」と「示そうとしていること」を明確にすること!!

1. Mathematical preliminaries: Proof techniques

- Induction (example)
  - S(n): 〚$1+3+\ldots+(2n-1)=n^2$ for any n〛
    1. Base step: Show the correctness of S(1):

       When n=1, clearly, we have $1 = 1^2$.
    2. Inductive step: Show that 'S(i+1) holds if S(i) holds'.

Hypothesis: When n=I, $1+2+\ldots+(2i-1) = i^2$

Goal: $1+2+\ldots+(2i-1)+(2(i+1)-1) = (i+1)^2$

Make 'Hypothesis' and 'Goal' clear!!

$1+2+\ldots+(2i-1)+(2(i+1)-1)=1+2+\ldots+(2i-1)+(2i+1)$

$=i^2 + 2i +1$ (By inductive hypothesis)

$=(i+1)^2$

1. 数学的準備: 証明技法

- 『数学的帰納法』と『再帰的定義』と『再帰呼出』との関連
  - 再帰的定義の基本形(関数f(*i*)を定義するとき)
    - 基礎ステップ: 有限個の例(f(0),f(1)など)に対して値を決めておく
    - 帰納的ステップ: f(*i*+1) の値を f(*i*) の値で定義する

再帰的定義
- f(1)=1
- f(*n*)=f(*n*-1)+(2*n*-1) (ただし*n*>1)

演繹的定義
f(*n*)=1+3+…+(2*n*-1)

定理: $n \geqq 1$ のとき、$f(n)= n^2$ が成立する。

1. Mathematical preliminaries: Proof techniques

- Relationship among ‘Induction’, ‘Recursive definition’ and ‘Recursive call’
  - Typical recursive definition (of a function f(*i*))
    - Base step: definition(s) for some finite instance(s) (typically f(0) or/and f(1))
    - Recursive step: define f(*i*+1) by using f(*i*)

Recursive definition
- f(1)=1
- f(*n*)=f(*n*-1)+(2*n*-1) (for *n*>1)

Deductive definition
f(*n*)=1+3+…+(2*n*-1)

Theorem: we have $f(n)=n^2$ for $n \geqq 1$

1. 数学的準備: 証明技法

- 『数学的帰納法』と『再帰的定義』と『再帰呼出』との関連
  - 再帰的定義の特徴
    - 定義の中に[…]がない

    ⇒計算機で扱える
    （再帰呼び出しで計算できる)

演繹的定義: f(n)=1+3+…+(2n-1)

再帰的定義
- f(1)=1
- f(n)=f(n-1)+(2n-1) (ただしn>1)

  - 数学的帰納法で証明できる関数は再帰的定義ができて、再帰呼び出しで計算できる

★再帰呼び出しで計算できる関数は数学的帰納法で正当性を証明できる

1. Mathematical preliminaries: Proof techniques

- Relationship among ‘Induction’, ‘Recursive definition’ and ‘Recursive call’
  - Property of recursive definition
    - We have no […] in definition

    ⇒Easy to deal with by computers
    （easy to compute by recursive call)
  - The function which can be proved by induction
    - can be defined by recursive definition
    - can be computed by recursive call

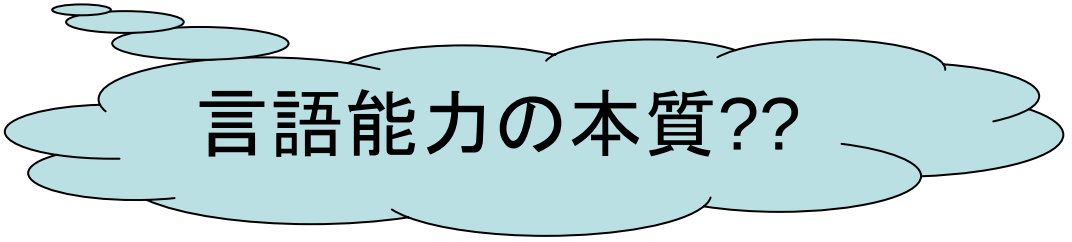
★For the function which can be computed by recursive calls, we can show the correctness of the computation by induction.

Deductive definition
f(n)=1+3+…+(2n-1)

Recursive definition
- **f**(1)=1
- f(n)=f(n-1)+(2n-1) (for n>1)

1. 数学的準備: 余談

- 『数学的帰納法』と『再帰的定義』と『再帰呼出』の正しさの基礎
  - 「数理哲学序説」ラッセル、平野智治訳、岩波文庫、1954.
  - 「自然数」とは何か？
    - 最初の自然数がある
    - どんな自然数にも次の自然数がある
    - 2つ以上の自然数の「次の自然数」になる自然数はない



1. Mathematical preliminaries: Addendum

- The base of the correctness of ‘Induction’, ‘Recursive definition’ and ‘Recursive call’
  - Introduction to Mathematical Philosophy, Bertland Russel, 1920.
  - How can we define ‘Natural Numbers’??
    - There is the first natural number.
    - There is the next natural number for each natural number.
    - There is no natural number that is the next natural number of two or more different natural numbers.