授業概要

　計算機科学の基礎である言語理論・オートマトン理論について学びます。
　この授業でいう言語とは、文字列の集りです。この授業では、言語を生成する仕組みとしての文法と、言語を認識する仕組みとしてのオートマトン（自動機械）との対応関係について学びます。
　言語理論・オートマトン理論と呼ばれるこの分野は、コンピュータによるテキスト処理や構文解析など幅広い応用を持ち、さらには計算機の能力の限界を明らかにする、計算機科学の重要な基礎分野です。この授業では基本的な諸概念を分かりやすく解説します。また、インターネットにおける暗号システムの安全性とも密接に関係することを学びます。

受講者へのメッセージ

この授業では数学的な内容を取り扱いますが、文系高卒程度の知識を前提に、関連するトピックを幅広く紹介します。
学習を表面的な理解に留めずより深く自分のものとするために、授業中に提示する課題、パズル等（必ずしも一つの正解があるとは限りません）に積極的に取り組み、コンピュータの理論的な可能性について考えてみて下さい。

レポート課題

興味を持ったパズルの答、この授業を通して学んだこと・考えたこと、授業への意見・感想等をレポートとして提出してください。
注意：レポートが提出されない場合、不合格（評価不能）となります。

期日：授業終了後１週間以内
提出場所：多摩学習センター事務室
　　窓口提出・郵送・メール添付（tama-sc@ouj.ac.jp）
　　山崎　メール添付（yamasaki.hideki@ouj.ac.jp）
形式：A4、用紙は何でもかまいません。
　　メール添付の場合、WORD又はPDFファイルで。
表紙：以下の事項を記載した表紙を付けて下さい。
　タイトル：言語理論とオートマトンレポート
　学籍番号：
　氏名：
　提出日：

参考書

著者名　ホップクロフト、モトワニ、ウルマン
書名　オートマトン 言語理論 計算論 I［第2版］
出版社名　サイエンス社
定価　3024円【税込】
ＩＳＢＮ　978-4-7819-1026-0

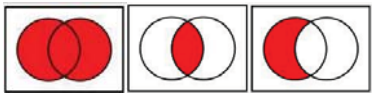
著者名　ホップクロフト、モトワニ、ウルマン
書名　オートマトン 言語理論 計算論 II［第2版］
出版社名　サイエンス社
定価　2808円【税込】
ＩＳＢＮ　978-4-7819-1027-7

原著（https://mcdtu.files.wordpress.com/2017/03/introduction-to-automata-theory.pdf）

集合の記法

集合は要素(もの)の集まり
　外延的記法：集合の要素を列挙 $\{2,3,5,7\}$
　内包的記法：要素が集合に属す条件を記述 $\{x|x$は10以下の素数$\}$

属す：$a \in A$
含まれる：$A \subseteq B$　真に含まれる：$A \subset B$　等しい：$A = B$
和集合：$A \cup B := \{x|x \in A$または$x \in B\}$
積集合：$A \cap B := \{x|x \in A$かつ$x \in B\}$
差集合：$A - B := \{x|x \in A$かつ$x \notin B\}$
補集合：$A^C := U - A$　Uは全体集合
（$:=$ は左辺を右辺で定義することを表す）


ウィキペディア

$\mathbb{N} := \{0,1,2,\cdots\}$：自然数(0以上の整数)の集合
$\mathbb{N}_+ := \{1,2,\cdots\}$：正整数の集合

パズル．上にならって補集合を図示せよ

文字列とその演算

アルファベット：文字（記号）の有限集合、通常Σ（やΓ）で表す

Σ上の長さnの文字列：Σの文字を(重複を許して)n個ならべたもの
　特に長さ0の文字列を空列といい通常εで表す
$|w|$：文字列wの長さ
例．$\Sigma = \{a,b\}$上の長さ3の文字列：$aaa,\ aab,\ aba,\ abb,\ baa,\ \cdots$
　　$w = abba \Rightarrow |w| = 4$

Σ上の文字列の全体をΣ^*と表す
例．$\{a,b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$

注．計算機科学で*は0回以上の繰り返しを意味することが多い
Σ^*はΣの文字を0回以上繰り返したものの全体

文字列の演算（以下v, wは任意の文字列）
　連接 ・（つなげる）　$a \cdot ba = aba,\ ba \cdot a = baa,\ \varepsilon \cdot w = w \cdot \varepsilon = w$
　ベキ（n回つなげる）　$a^4 = aaaa,\ (ab)^3 = ababab,\ w^0 = \varepsilon$
　反転（逆順にする）　$(abb)^R = bba$、一般に$(a_1 a_2 \cdots a_n)^R = a_n \cdots a_2 a_1$
　準同型写像　$h : \Sigma^* \to \Gamma^*$　で　$h(vw) = h(v)h(w)$　を満たすもの。
　　$h : \Sigma \to \Gamma^*$　から一意に定まる(拡張できる)
　　例えば　$h(a) = 00,\ h(b) = 101$　なら　$h(aba) = 00101100$。
　　特に　$v \neq w$　なら　$h(v) \neq h(w)$　となるとき、
　　　h　を符号化と呼び、$h(\Sigma) := \{h(a)|a \in \Sigma\}$　を符号と呼ぶ。

パズル．$h : \Sigma^* \to \Gamma^*$　が準同型写像ならば　$h(\varepsilon) = \varepsilon$　であることを示せ

文字列のパズルとコラム

パズル
1. $vw = wv$　となる必要十分条件は文字列　v　と　w　がともにある文字列　x　のベキ（$v = x^n,\ w = x^m$）であることを示せ
2. $h(a) = 1,\ h(b) = 01,\ h(c) = 00$　のとき　$h(w) = 011001$　となる　w　を求めよ。また　h　が符号化であることを示せ。

コラム
$h(0) = 1,\ h(1) = 0$　とし、
$w_1 = 01,\ w_2 = w_1 h(w_1) = 0110,\ w_3 = w_2 h(w_2) = 01101001, \cdots$　と定めると、
$w_1, w_2, w_3, \cdots$　は同一文字列の3回の繰り返し　x^3　を部分文字列として含まない（証明は少し難しい）。このことから将棋の千日手の規定が、「同一手順を3回繰り返した時」から、1983年に「同一局面が4回出現した時」に変わった。

言語とその演算

Σ^n ：Σ上の長さnの文字列全体
$\Sigma^* := \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cdots$ ：(長さ0以上の)文字列全体
例. $\{a,b\}^0 = \{\varepsilon\},\ \{a,b\}^1 = \{a,b\},\ \{a,b\}^2 = \{aa, ab, ba, bb\}$
$\{a,b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \cdots\}$

Σ上の言語：Σ^*の部分集合(つまり文字列の集合)
集合演算：和：$\cup$, 積：$\cap$, 補集合：^{C}, 差：$-$
連接 $A \cdot B := \{vw | v \in A,\ w \in B\}$：$A$の文字列に$B$の文字列を繋げた文字列全体
ベキ $A^0 := \{\varepsilon\},\ A^1 := A,\ A^2 := A \cdot A,\ A^3 := A \cdot A \cdot A, \ldots$
A^n ：Aの文字列をn個繋げた文字列全体
クリーネ閉包 $A^* := A^0 \cup A^1 \cup A^2 \cup \cdots$：$A$の文字列を0個以上繋げた文字列全体
($A^? := A^0 \cup A = \{\varepsilon\} \cup A,\quad A^+ := A^1 \cup A^2 \cup \cdots$)
反転 $A^R := \{w^R | w \in A\}$
準同型写像 $h(A) := \{h(w) | w \in A\}$。$h$が符号化のとき、集合$h(\Sigma)$を符号という。

パズル. $A = \{a, ab\}$とする。$A^? A,\ AA^R,\ A^+$を求めよ

言語の記述

言語は、一般に無限集合で、外延的に(並べあげて)は記述できない
内包的に記述する仕組み
①言葉あるいは演算による表現
②文法：言語を生成する仕組み
③オートマトン：言語を認識する仕組み

例. 日本語とは
②日本語の文法にかなった文字列
③日本人が理解（認識）する文字列

この授業のテーマ：文法とオートマトン（機械）の対応関係とそれらが生成・認識する言語の特徴づけ

パズル.
1. automaton（オートマトン）の本来の意味は何か
2. $\{x | x$は20以下の素数$\} = \{2,3,5,7,11,13,17,19\} \subseteq \{1,2,3,5,7,9\}^*$ は正しいか。

文法とは

チョムスキー：人が幼児期に母語を驚くほど短期間に獲得するのは、言語の初期状態である普遍文法を生得的に備えているから

普遍文法：句構造文法
言語を生成する仕組み。(書換え)規則の集まりと開始記号から成る
/文/→/主語//述部/
/述部/→/述語/|/目的語//述部/
(/述部/→/述語/、/述部/→/目的語//述部/の略記)
/主語/→ 私は|君は|・・・
/目的語/→車を|彼に|・・・
/述語/→買う|売る|・・・

・規則は 左辺→右辺の形
・規則中の記号：終端記号(文字)か非終端記号（文法要素）
・開始記号は非終端記号（/文/）
・規則の左辺はすくなとも1つ非終端記号を含む

導出と生成

導出：規則の左辺を右辺で書き換え /文/ ⇒/主語//述部/ ⇒/主語//目的語//述部/ ⇒私は/目的語//述部/ ⇒私は彼に/述部/ ⇒私は彼に/目的語//述部/ ⇒私は彼に/目的語//述語/ ⇒私は彼に車を/述語/ ⇒私は彼に車を売る	文法が生成する文字列： 開始記号から導出される終端文字列 文法が生成する言語： 文法が生成する文字列の集合 二つの文法が等しい： 生成する言語が等しい。

パズル. 上に倣って、前頁の文法が生成する文を例示せよ。

文法の階層

Γ:非終端アルファベット（文法要素）、Σ:終端アルファベット（文字）

文法クラス	規則の形 $A \in \Gamma,\ w \in \Sigma^*,\ B \in \Gamma^?$ $\alpha \in (\Sigma \cup \Gamma)^* - \Sigma^*,\ \beta \in (\Sigma \cup \Gamma)^*$	文法の例 開始記号S	生成する言語
正則(正規)文法	$A \to wB$	$S \to 0S \mid 0B$ $B \to 1B \mid 1$	$\{0^n1^m \mid n, m \in \mathbb{N}_+\}$
文脈自由文法	$A \to \beta$	$S \to 0S1 \mid 01$	$\{0^n1^n \mid n \in \mathbb{N}_+\}$
単調文法	$\alpha \to \beta$ $(\lvert\alpha\rvert \le \lvert\beta\rvert)$ 注	$S \to 012 \mid 0SB2$ $2B \to B2$ $1B \to 11$	$\{0^n1^n2^n \mid n \in \mathbb{N}_+\}$
句構造文法 (制限なし)	$\alpha \to \beta$		

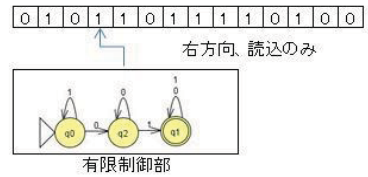
注．$S \to 0S \mid 0B$は左辺が同じ規則$S \to 0S,\ S \to 0B$をまとめて表したもの。

パズル．単調文法の上の定義では空列を生成できないことを示せ。
注．このため単調文法は規則の右辺に現れない開始記号Sに限り$S \to \varepsilon$を許す

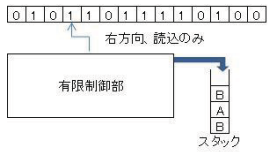
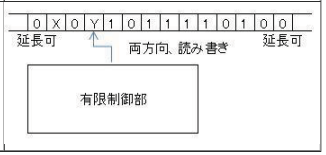
オートマトンとは

言語を認識(受理)する仕組み（機械）
　有限制御部と（入力が記された）テープを持ち
　テープセルを読み（書きし）ながら、有限制御部の状態が遷移してゆく。

オートマトンが受理する文字列
　開始状態から出発し受理状態に到達して停止する動作列を持つテープ内容
オートマトンが認識(受理)する言語：オートマトンが受理する文字列の集合
２つのオートマトンが等しい：認識する言語が等しい。

有限オートマトン	テープヘッドは読込のみで、右方向にしか動かない	

オートマトンの階層

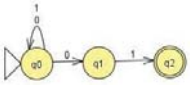
プッシュダウンオートマトン	有限オートマトン＋スタック 【一方向から記号の追加・読出し(取出し)ができる】	
線形有界オートマトン	入力テープが同時に作業テープの役割を果たし、テープ上を行きつ戻りつしながら読み書きできる	
チューリングマシーン	さらにテープを必要なら両方向にいくらでも延長できる	

パズル．スタックがLIFO（Last In First Out）であるとはどういう意味か？

非決定性オートマトン

決定性オートマトンの動作
　有限制御部の状態とテープ(やスタック)セルの内容から１つに定まる
　受理状態に到達して停止すれば、入力は受理される
非決定性オートマトンの動作
　１つに定まらず、複数の動作が定義されていてもよい
　可能な動作列中に受理状態に到達して停止するものがあれば、入力は受理
「運よく受理できればよい」：機械としては妙な非常に都合のよい仮定
　設計はしやすいが、実装には決定性（への変換）が必要

例．01で終わる$0,1$の列を認識する非決定性有限オートマトン
状態q_0で0を読んだ時、状態$q_0,\ q_1$のどちらにも遷移できる。



JFLAPで作成

入力(ラベル)列	受理・非受理状態に至る動作列	受理
1001	ある・ある	受理
1010	ない・ある	非受理

文法とオートマトンの対応関係

文法の生成能力とオートマトンの認識能力の階層関係

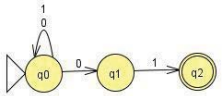
決定性有限オートマトン＝非決定性有限オートマトン＝正則文法：正則言語
⊂決定性プッシュダウンオートマトン：決定性文脈自由言語
⊂非決定性プッシュダウンオートマトン＝文脈自由文法：文脈自由言語
⊂決定性線形有界オートマトン
⊆非決定性線形有界オートマトン＝単調文法：文脈依存言語
⊂決定性チューリング機械＝非決定性チューリング機械＝句構造(無制限)文法：帰納的可算言語

注．文脈依存言語という術語は、このクラスが最初、文脈依存文法（規則が$\alpha A\beta \to \alpha\gamma\beta, |\gamma| \geq 1$の形）で生成される言語として定義されたことに由来する。帰納的可算言語という術語は、このクラスの言語では、属する文字列の列挙（のみ）が可能であるという事実に由来する。

言語の階層

言語クラス	下の階層に属さない例
任意の言語	$\{(P,w)\|$プログラムPは入力wで停止しない$\}$
帰納的可算言語	$\{(P,w)\|$プログラムPは入力wで停止する$\}$
文脈依存言語	未解決
決定性文脈依存言語	$\{0^n1^n2^n \| n \in \mathbb{N}_+\}$, $\{ww \| w \in \{0,1\}^+\}$
文脈自由言語	$\{ww^R \| w \in \{0,1\}^+\}$
決定性文脈自由言語	$\{0^n1^n \| n \in \mathbb{N}_+\}$, $\{wcw^R \| w \in \{0,1\}^+\}$
正則言語	$\{0^m1^n \| m,n \in \mathbb{N}_+\}$

有限オートマトン（FA）

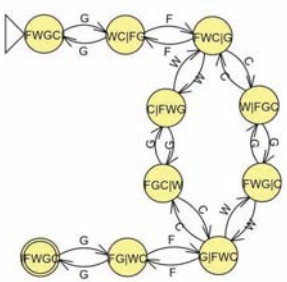


Σ上の有限オートマトン（FA）M の構成要素
　状態（〇）と遷移（ラベル付き矢印$\xrightarrow{a}$, $a \in \Sigma^?$）
　　入力a（εの場合は無入力）による状態遷移
　開始状態(▷〇、1つ)と、受理状態(◎、複数可)
開始状態から受理状態への遷移列（路）の入力（ラベル）列を受理
受理入力列の集合を認識(受理)

時点表示（状態,未読文字列）と動作列
$(q_0, 1001) \vdash (q_0, 001) \vdash (q_0, 01) \vdash (q_1, 1) \vdash (q_2, \varepsilon) \Rightarrow (q_0, 1001) \overset{*}{\vdash} (q_2, \varepsilon)$：受理
$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2$を$q_0 \xrightarrow{1001} q_2$と書けば $(q, w) \overset{*}{\vdash} (p, \varepsilon) \Leftrightarrow q \xrightarrow{w} p$
FAMが認識する言語：$L(M) = \{w \mid (\triangleright\bigcirc, w) \overset{*}{\vdash} (\textcircled{\scriptsize◎}, \varepsilon)\} = \{w \mid \triangleright\bigcirc \xrightarrow{w} \textcircled{\scriptsize◎}\}$

パズル．右上図のFAについて
①00101に対する可能な動作をすべて調べよ　答
②どのような文字列を受理するか

農夫パズル



オオカミ(W)とヤギ(G)を連れキャベツの籠(C)を持った農夫(F)が小舟で川を渡りたい。
1. 小舟には農夫の他１頭(１籠)しか乗らない
2. 農夫がいないと、オオカミはヤギを食べ
　ヤギはキャベツを食べる
無事川を渡るにはどうすればよいか。

状態：両岸にいる(ある)ものの配置
遷移：農夫が小舟で運ぶもの W, G, C。なければ F
開始状態：▷FWGC　　受理状態：◎FWGC

パズル．農法パズルの解を２つ示せ。

3で割り切れる2進数

3で割り切れる2進数を受理する有限オートマトンを考えてみよう。
基本的なアイディアは
　開始状態から$0,1$の列wで状態q_kに到達⇔2進数wの値を3で割った余りがk
となるようにすることである。
2進数wの値を$[w]_2$で表すと、$[w0]_2 = 2[w]_2$, $[w1]_2 = 2[w]_2 + 1$である。

パズル．上のアイディアに沿って、有限オートマトンを描け。
　答　JFLAPファイル

決定性有限オートマトン（DFA）

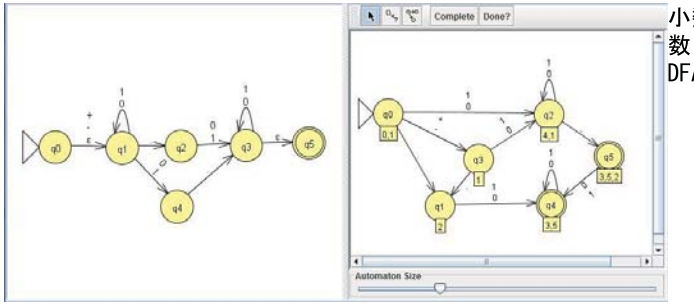
辺のラベル（文字）の集合をΣとする。
有限オートマトンは次の条件を満たすとき決定性であるという。
　ε遷移を持たず、すべての状態とラベル($\in \Sigma$)の組に対して、遷移先の状態が1つ

DFA(Deterministic Finite Automaton)では、
　開始状態から与えられた入力(ラベル列)を持つ路は唯一つ(ある①)
　入力は、到達先の状態が受理状態なら受理、受理状態でなければ非受理
FA(Finite Automaton)では、
　開始状態から与えられた入力(ラベル列)を持つ路は複数(または0個ある②)
　入力は、到達先の状態の中に受理状態があれば受理、なければ非受理

注．
① DFAにおいて、遷移先の状態がないことを許す場合もある。その場合は死状態（そこでループする非受理状態）への遷移を省略していると考えればよい
② 状態とラベル($\in \Sigma$)の組に対し複数の遷移を許すものをNFA(非決定性有限オートマトン, Nondeterministic Finite Automaton)、さらにεラベルの遷移を許すものをε-NFAと区別することもある。この授業でのFAはε-NFA

FA→DFAの変換

定理．FAMで認識可能⇒DFAM_Dで認識可能
略証．M_Dを次のように作る。
- M_Dの入力wで到達する状態:=Mのwで到達可能な状態の集合
- M_Dの開始状態:=Mの開始状態からεで到達可能な状態の集合
- M_Dの受理状態:=Mの受理状態を含む（M_Dの）状態

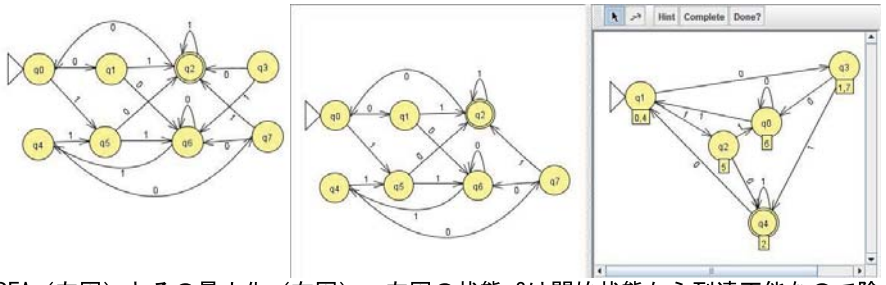


小数点（符号）を含む2進数を認識するFA（左図）とDFA（右図）

DFAの最小化

DFAの状態p, q が同値：$p \xrightarrow{w} ◎ \Leftrightarrow q \xrightarrow{w} ◎$
　即ちそれぞれを開始状態とみて同じ言語を認識するとき

定理．DFAの状態を、開始状態から到達不能な状態を除いて同値類に分割して得られるDFAは、同じ言語を受理する状態数最小のDFAである。



DFA（左図）とその最小化（右図）：左図の状態q3は開始状態から到達不能なので除かれる(中央図)。

正則文法（RG）

規則の形：$A \to wB$（$A \in \Gamma,\ w \in \Sigma^*,\ B \in \Gamma^?$。右線形文法ともいう）
　（Σ：終端アルファベット、Γ：非終端アルファベット）
$A \to wB$による導出：$vA \Rightarrow vwB$（正則文法の導出では非終端記号は右端だけ）
導出列：$v_0A_0 \Rightarrow v_0v_1A_1 \Rightarrow \cdots \Rightarrow v_0v_1\cdots v_nA_n$のとき$v_0A_0 \overset{*}{\Rightarrow} v_0v_1\cdots v_nA_n$と書く
文法Gが生成する言語：$L(G) = \{w \in \Sigma^* | S \overset{*}{\Rightarrow} w\}$

例．RG S→1S|0S|0A、A→1 の導出例
$S \Rightarrow 1S \Rightarrow 10A \Rightarrow 101$
パズル．①上の文法で文字列1001が生成される(Sから導出される)ことを示せ
②上の文法で生成されるのはどのような文字列か。

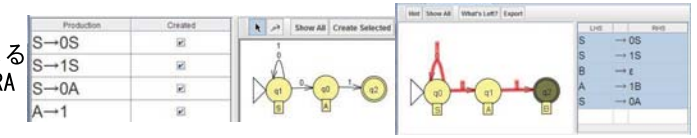
標準形：RGの規則は$A \to aB$（$A \in \Gamma,\ a \in \Sigma^?,\ B \in \Gamma^?$）の形にできる。
∵）$A \to a_1a_2\cdots a_kB$をk個の規則$A \to a_1B_1,\ B_1 \to a_2B_2, \cdots,\ B_{k-1} \to a_kB$に置き換えればよい。（$B_1,\ B_2 \cdots,\ B_{k-1}$は規則毎に追加する非終端記号）

RG⇔FA

標準形RGは、次の対応によってFAの表現と見なせる。
開始記号S ⇔ 開始状態 $\triangleright Ⓢ$
$A \to aB$ ⇔ $Ⓐ \xrightarrow{a} Ⓑ$　（$A \in \Gamma,\ a \in \Sigma^?,\ B \in \Gamma$）
$A \to a$ ⇔ $Ⓐ \xrightarrow{a} ◎$（$A \in \Gamma,\ a \in \Sigma^?$）
このとき、 $S \overset{*}{\Rightarrow} wA \Leftrightarrow \triangleright Ⓢ \xrightarrow{w} Ⓐ$、$S \overset{*}{\Rightarrow} w \Leftrightarrow \triangleright Ⓢ \xrightarrow{w} ◎$ が成り立つ。

定理．次の３条件は同値。これらを満たす言語を正則言語という
　①DFAで認識される⇔②FAで認識される⇔③RGで生成される

例．JFLAPによるRG⇒FAとFA⇒RA



正則言語の性質

定理．正則言語は以下の演算で閉じている。
①和集合 ②積集合 ③補集合 ④反転 ⑤準同型写像 ⑥連接 ⑦クリーネ閉包
∵）L_1, L_2を生成するRGG_1, G_2の開始記号をそれぞれS_1, S_2とする。
①規則$S \to S_1 \mid S_2$ を追加すれば、Sは$L_1 \cup L_2$ を生成する。
③Lを認識するDFAで、受理状態と非受理状態を入れ替えればL^Cを認識する。
②$L_1 \cap L_2 = (L_1^C \cup L_2^C)^C$より明らか。
⑤G_1の規則$A \to wB$を$A \to h(w)B$に置き換えればS_1は$h(L_1)$を生成する。
⑥G_1の規則$A \to w$を$A \to wS_2$に置きかえればS_1はL_1L_2を生成する。
⑦G_1の規則$A \to w$を$A \to wS_1$に置き換え$S_1 \to \varepsilon$を追加すればS_1はL_1^*を生成する。

パズル．Lが正則集合ならその反転L^Rも正則集合であることを示せ。

正則言語の特徴づけ

$w\backslash L := \{v \mid wv \in L\}$（左微分）と定義する。
定理．$L\ (\subseteq \Sigma^*)$ が正則言語 $\Leftrightarrow$ $\{w\backslash L \mid w \in \Sigma^*\}$が有限集合
∵）Lを認識するDFAMで、▷○$\xrightarrow{w}$ⓠならば$w\backslash L = \{v \mid$ⓠ$\xrightarrow{v}$◎$\}$なので、$\{w\backslash L \mid w \in \Sigma^*\}$は$L$を認識する最小DFAの状態集合とみなせる。

反復補題．$L\ (\subseteq \Sigma^*)$ が正則言語ならば、Lの十分長い文字列wは$w = xyz,\ y \neq \varepsilon$と書けて、任意の自然数$k$に対し $xy^kz \in L$。
∵）Lを認識するDFAMにおける遷移列 ▷○$\xrightarrow{w}$◎ は、wが十分長ければ途中で同じ状態qが2回現れる（鳩ノ巣原理）。よって、$w = xyz$と書けて ▷○$\xrightarrow{x}$ⓠ$\xrightarrow{y^k}$ⓠ$\xrightarrow{z}$◎。

応用．$L = \{0^n1^n \mid n \in \mathbb{N}\}$は正則言語ではない。
【証明1】$0^k\backslash L = \{0^n1^{n+k} \mid n \in \mathbb{N}\}$は$k$毎に異なり、$\{w\backslash L \mid w \in \{0,1\}^*\}$は非有限集合
【証明2】Lの十分長い文字列xyzに対し、$y = 0^j,\ 0^j1^k,\ 1^k$のどの場合も $xy^2z \notin L$

正則（正規）表現

定理．Lが正則言語$\Leftrightarrow$ Lは文字列（の単元集合）から$\cup$, $\cdot$, *を繰り返して得られる
$\Leftarrow$）明らか
$\Rightarrow$を証明するために言語の方程式を考える。
補題．言語の方程式$X = AX \cup B$の（最小）解はA^*Bである。
∵）$A(A^*B) \cup B = A^+B \cup A^0B = A^*B$より$X = A^*B$は解。$Y = AY \cup B$（解）ならば $Y \supseteq B,\ Y \supseteq AY \supseteq AB,\ Y \supseteq AY \supseteq A^2B,\ \cdots$より$Y \supseteq A^*B$。
$\Rightarrow$）Lを生成する正則文法の規則$A \to w_1B_1 \mid w_2B_2 \mid \cdots \mid w_kB_k$に対し式$A = \{w_1\}B_1 \cup \{w_2\}B_2 \cup \cdots \cup \{w_k\}B_k$を作成して得られる連立方程式の（最少）解は、各非終端記号から生成される言語であり、その構成には$\cup$, $\cdot$, *しか現れない。

例．文法$\begin{cases} S \to 0S \mid 0B \\ B \to 1B \mid 1 \end{cases}$から連立方程式$\begin{cases} S = \{0\}S \cup \{0\}B & ① \\ B = \{1\}B \cup \{1\} & ② \end{cases}$を得る。②より、
$B = \{1\}^*\{1\}$。これを①に代入して$S = \{0\}S \cup \{0\}\{1\}^*\{1\} = \{0\}^*\{0\}\{1\}^*\{1\}$

正則言語の、文字列の単元集合と$\cup$, $\cdot$, *を使う表現を正則（正規）表現という。

正則言語のパズル

パズル．3で割り切れる2進数の集合について
１．認識する有限オートマトンを求めよ　答
２．生成する正則文法を求めよ
３．正則表現を求めよ（文字列に対する{, }は省略してよい）

正規（正則）表現とは

正規(正則)表現は応用範囲が広く、コンピュータで入力可能なように、

- 単元集合の{,}を省略
- εは文字通り空列で
- 和集合∪を|で
- 連接は単に表現をつなげて
- 閉包 * を*で
- *、|、(、)などの文字は直前に"¥"をつけ
  例えば、*は¥*で、¥は¥¥で

例. 正則集合
$(\{0\} \cup \{1\})^*\{01\}$
を
(0|1)*01
と表現

表す。さらに、多くの便宜的記法があり、それらを用いた表現を以後慣習に基づき正規表現と呼ぶ

また正規表現は、文字列パターンの記述と捉えることができ、文字列wが正規表現αが表す正則言語に属すとき、wはαに合致するという。

正規表現の記法

基本

記法	説明
連接	正規表現を続けて書けば、それらを続けた文字列と合致
|	和(または)。どちらかのパターンと合致すれば合致。
*	直前の文字(文字列)の0回以上の繰り返しと合致

繰返し

記法	説明	使用例
+	1回以上の繰り返し	12+ → 12、122、・・・ と合致 12+=122*
?	0,1回の繰り返し	12? → 1、12 と合致 12?=1|12
{n}	n回繰り返し	(ab){3} → abababに合致
{n, }	n回以上繰り返し	a{3,} →aaa、aaaa、…に合致
{m,n}	m～n回繰り返し	a{3,5} →aaa、aaaa、aaaaaに合致

正規表現の記法

文字の集合：1文字指定する

記法	説明	使用例
.	改行文字を除く任意の1文字	windows.. windowsの後に2文字続く文字列と合致
[ ][ ]	内に並べた文字のどれか	[ab]＝a|b
		[a-z]＝a|b|・・・|z 小文字アルファベット全て
-	文字の区間。[ ] 内で用いる	[０-９]＝０|１|・・・|９ 全角の数字
		[0-9A-Z] 大文字のアルファベットか数字
^	否定。[ ] 内で用いる	[^a-z] 小文字のアルファベット以外全て

合致する文字列は、通常はパターンに適合する最長の文字列(最長合致)であるが、"?"を付加すれば最短合致となる。

12+ →1222222222 に対し10桁目の1222222222までと合致
12+? →1222222222 に対し2桁目の12までと合致

正規表現の記法

特殊な意味を持つ表現

記法	説明	使用例
^	行頭	^abc →行頭にある abc と合致
$	行末	abc$ →行末にある abc と合致

記法	説明	記法	説明
¥w	アルファベット、数字又は下線	¥W	アルファベット、数字、下線以外
¥d	数字	¥D	数字以外。[^0-9] と同じ
¥s	空白文字(スペース、タブ、改行)	¥S	空白文字以外
¥b	単語の区切り	¥B	単語の区切り以外
¥n	改行		
¥r	リターン(復帰)		
¥t	タブ		

正規表現による英文分析

正規表現による検索では grep が有名だが、ここでは英文の分析に利用する例を紹介しよう。これによって、文書・文献の特徴を数量的に抽出し、著者の異同や時代の推定、思想と文体との関係等を分析することも可能になる。その意味で、正規表現検索は文書の処理・解析のための重要なツールになっている。

実験. 秋田大学のSCORP検索(http://www.mis.med.akita-u.ac.jp/%7Etaka/FreeSCORP/)

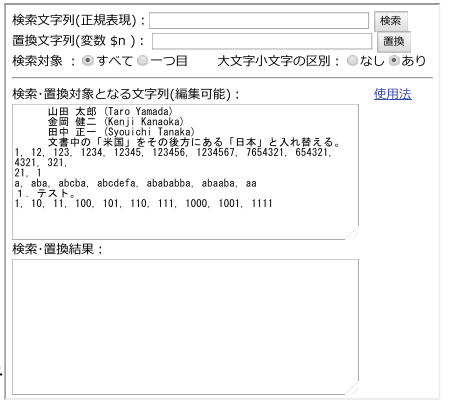
1. Search Word欄に色々な正規表現を入れて試してみよう。
2. 適当な動詞(過去、過去分詞、三単元を含む)の出現数を調べてみよう。

パズル. 正規表現 ¥s(for|in|on|to|with)¥s[a-z]+ing¥s(for|in|of|on|to|with)¥s は何を意味するか考えよ。

正規表現による検索と置換

n：正規表現のn番目の括弧対で括られた部分に合致した文字列

右のフレームで検索文字列や置換文字列を指定して[検索]、[置換]を押すと結果が下の欄に表示される（Java Scriptで実装）



パズル. 1～5の正規表現を求め検索・置換せよ。

1. 3桁の数字
2. 3000未満の非負整数
3. aで始まりaで終わる6文字以下の英小文字列
4. 全角数字０～９で始まり句点。で終わる行
5. 英字の姓と名を入れ替えよ
6. 行頭の空白を削除せよ

Lex

字句解析プログラム生成ツール。Lexでは正規表現で指定された文字列に対する処理が記述でき、その応用領域は、コンパイラの(字句解析部)の作成、自然言語処理や簡単な整形まで幅広い。

規則の形は以下

正規表現 アクション(合致文字列に対する処理);

合致する正規表現がない文字列はそのまま出力

用途

（１）テキスト処理（単独で）

例. 行の末尾の空白 を消去

[ ]+$;

（２）字句解析プログラム生成系（yaccと組み合わせて）

字句解析：文の中から字句（単語）を切り分け、分類する

Lex（字句解析）

例. プログラム中の予約語、名前、数、演算記号等を切り分け分類するlex入力

正規表現	アクション
else	return(ELSE);
[A-Za-z][A-Za-z0-9]*	{見つけた名前を記号表に登録;return(ID);}
[+-]?(0|[1-9][0-9]*)	{表す整数値を定数表に登録;return(NUM);}
>=	return(GE);
=	return(EQ);
・・・	・・・

パズル. 小数点数を表す正規表現を求めよ

文脈自由文法（CFG）

規則が$A \to \alpha$の形の文法（$A \in \Gamma,\ \alpha \in (\Sigma \cup \Gamma)^*$）
（Σ：終端アルファベット、Γ：非終端アルファベット）

$A \to \alpha$による導出：$\beta A \gamma \Rightarrow \beta \alpha \gamma$
導出列：$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$のとき$\alpha_1 \overset{*}{\Rightarrow} \alpha_n$と書く
$A \in \Gamma$が生成する言語：$\{w \in \Sigma^* | A \overset{*}{\Rightarrow} w\}$

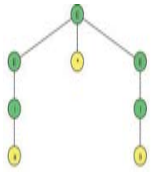
文法Gが生成する言語：Gの開始記号Sが生成する言語$L(G) := \{w \in \Sigma^* | S \overset{*}{\Rightarrow} w\}$

例．CFG $S \to 0S1|\varepsilon$ は$\{0^n1^n | n \in \mathbb{N}\}$を生成する
∵) $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow \cdots \Rightarrow 0^nS1^n \Rightarrow 0^n1^n$

パズル．$\{ww^R | w \in \{0,1\}^*\}$を生成するCFGを示せ

導出木

導出の過程を表現した木
例．単純な式を生成する文法 $\begin{cases} E \to I|E+E|E*E|(E) \\ I \to a|b|Ia|Ib|I0|I1 \end{cases}$ において

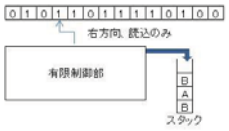


$a*b$は本質的に同じ導出を複数持つが導出木は１つ
$\begin{cases} E \Rightarrow E*E \Rightarrow I*E \Rightarrow a*E \Rightarrow a*I \Rightarrow a*b & \text{(最左導出)} \\ E \Rightarrow E*E \Rightarrow E*I \Rightarrow E*b \Rightarrow I*b \Rightarrow a*b & \text{(最右導出)} \end{cases}$

パズル．例の文法で$a+b*a$は複数の導出木を持つ（あいまいである）ことを示せ
ヒント

プッシュダウンオートマトン（PDA）

構成要素：状態$\in Q$、スタック$\in \Gamma^*$、入力テープ$\in \Sigma^*$
動作規則（ラベル付き辺）と時点表示（状態, 未読部, スタック）
$q \xrightarrow{a, X/\alpha} p \Rightarrow (q, aw, X\beta) \vdash (p, w, \alpha\beta)$
（スタックは左が上）　$(a \in \Sigma^?,\ X \in \Gamma,\ \alpha, \beta \in \Gamma^*)$



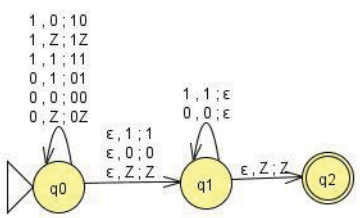
動作の意味
状態がq、スタックトップ記号がXで入力記号aを読んだとき（εなら読まずに）
状態をpに変え、スタックからXを取り除き α（の記号）を（右から）順に積む。
動作列：$I_1 \vdash I_2 \vdash \cdots I_n$のとき$I_1 \overset{*}{\vdash} I_n$と書く。

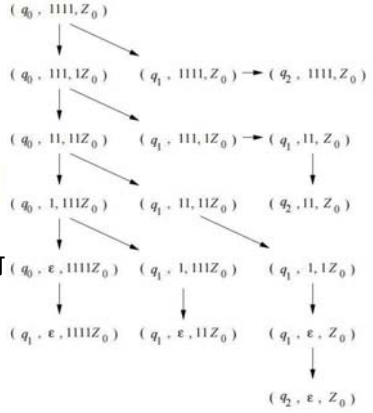
PDAMに対し、q_0を開始状態、Z_0をスタックの開始記号とする。
$(q_0, w, Z_0) \overset{*}{\vdash} (◎, \varepsilon, \alpha)$の（動作列がある）とき、$M$は最終（受理）状態で$w$を受理する。
$(q_0, w, Z_0) \overset{*}{\vdash} (○, \varepsilon, \varepsilon)$の（動作列がある）とき、$M$は空スタックで$w$を受理する。

注．スタックに積む動作をプッシュダウン、取り除く動作ポップアップという。

ww^Rを受理するPDA

右図のPDAは語ww^Rを受理する。
直感的には状態q_0で入力をスタックに積んでいき、中間地点で状態q_1に（ε）遷移し、以降スタックと入力を照合していき、成功すれば受理する。このPDAは非決定性なので、入力1111に対する可能な動作は複数あるが、右下に至る動作列で受理される。



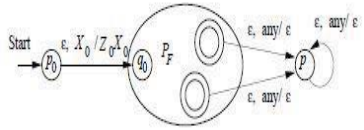


パズル．0^n1^nを受理するPDAを示せ
答え

最終状態受理⇔空スタック受理

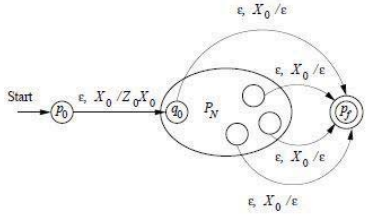
最終状態受理PDAM_f⇒空スタック受理PDA

M_fの最終状態に右図の遷移を付け加えてスタックを空にさせる



空スタック受理PDAM_ε⇒最終状態受理PDA

M_εのスタックの底に新たな記号X_0を置き、記号X_0が現れた(M_εでスタックが空になった)ら、最終状態に遷移させる



定理．言語Lが最終状態受理PDAで認識されることと空スタック受理PDAで認識されることは同値である。

PDA⇒CFG

空スタック受理PDAM⇒CFGG

Mの状態集合をQ、スタック記号をΓとする。Gの非終端アルファベットは$\{S\} \cup Q\Gamma Q$で

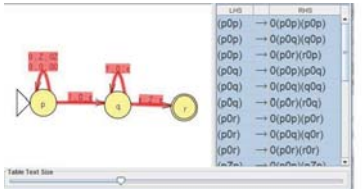
①$(q,w,X) \overset{*}{\vdash} (p,\varepsilon,\varepsilon) \Leftrightarrow [qXp] \overset{*}{\Rightarrow} w$となるように、$M$の遷移ⓠ$\xrightarrow{a,X/Y_1Y_2\cdots Y_k}$ⓡに対し

Gに規則$[qXr_k] \to a[rY_1r_1][r_1Y_2r_2]\cdots[r_{k-1}Y_kr_k]$（$r_1r_2\cdots r_k \in Q^k$）を追加し

②規則$S \to [q_0Z_0q]$（q_0はMの開始状態でZ_0はスタックの開始記号、$q \in Q$）を追加すれば

SはMが空スタックで受理する入力列を生成する。

例．JFLAPによるPDA⇒CFG



PDAとCFGの等価性

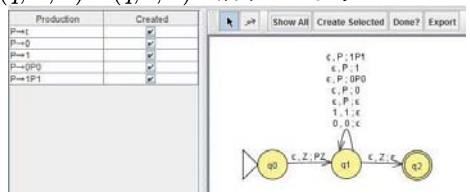
CFGG⇒空スタック受理PDAM

Mは唯1つの(開始)状態qを持ち、$S \overset{*}{\Rightarrow} w\alpha \Leftrightarrow (q,w,S) \overset{*}{\vdash} (q,\varepsilon,\alpha)$が成り立つように

①各入力記号aに対する遷移ⓠ$\xrightarrow{a,a/\varepsilon}$ⓠと

②Gの各規則$A \to \alpha$に対する遷移ⓠ$\xrightarrow{\varepsilon,A/\alpha}$ⓠを持つ。

例．JFLAPによるCFG⇒PDA



定理．同値な次の３条件をみたす言語Lを文脈自由言語（CFL）という。

①LはCFGで生成される

②Lは空スタック受理PDAで認識される

③Lは最終状態受理PDAで認識される

CFLの性質

和、連接、閉包、反転、準同型写像で閉じている　∵）文法を使って証明

正則言語との共通部分で閉じている　∵）PDAを使って証明

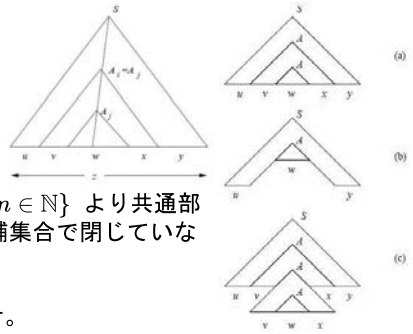
反復補題　$L\ (\subseteq \Sigma^*)$が文脈自由言語ならば、Lの十分長い文字列zは

$z = uvwxy,\ vx \neq \varepsilon$と書けて、任意の自然数$k$に対し、$uv^kwx^ky \in L$。

∵）Lを生成するCFGにおける$z = uvwxy$の構文木は、zが十分長ければ途中で同じ非終端記号が現れる路を持ち(鳩ノ巣原理、右図)、その部分の導出を繰り返すことができる(最右図)。

CFLでない言語

$\{0^n1^n2^n|n\in\mathbb{N}\}\notin$CFL　∵）反復補題よりCFLは3か所同期しては増えない



共通部分、補集合で閉じていない
　∵）
$\{0^n1^n2^n|n\in\mathbb{N}\}=\{0^n1^n|n\in\mathbb{N}\}\{2\}^*\cap\{0\}^*\{1^n2^n|n\in\mathbb{N}\}$ より共通部分で閉じていない。和集合では閉じているので、補集合で閉じていない。

パズル．$\{ww|w\in\{0,1\}^*\}$ はCFLでないことを示せ。

CFLの特徴づけ

Dyck言語　D_k：k対の括弧 $\{(_1,)_1,(_2,)_2,\cdots,(_k,)_k,\}$ のバランス（対応）のとれた列の集合
$S\to\varepsilon|(_1S)_1S|(_2S)_2S|\cdots|(_kS)_kS$　で生成される言語

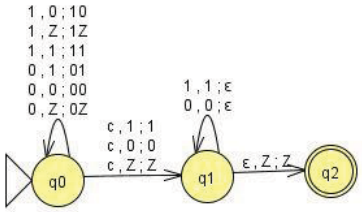
定理．任意のCFLはDyck言語と正則言語の共通部分の準同型写像の像
　∵）省略。

定理より、文脈自由言語は本質的に正則言語に括弧構造を付加したものと見なせる。

決定性PDA（DPDA）

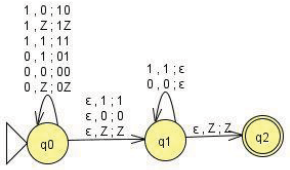
次の条件を満たすPDA
すべての状態q、$a\in\Sigma$、$X\in\Gamma$に対し
①動作$\textcircled{q}\xrightarrow{a,X/\cdot}$は高々1つ
②ε−動作$\textcircled{q}\xrightarrow{\varepsilon,X/\cdot}$があれば$\textcircled{q}\xrightarrow{a,X/\cdot}$を持たない

この条件を満たすと、与えれた入力に対する動作列は高々１つ（決定性）



例．右図は$\{wcw^R|w\in\{0,1\}^*\}$を認識するDPDA

文脈自由言語$\{ww^R|w\in\{0,1\}^*\}$を認識するDPDAは存在しない
　∵）wとw^Rの切れ目をDPDAでは判定できない。（PDAは右図）



プログラミング言語の記述

プログラミング言語は多くの括弧構造を持つ
C, Java : ( ), [ ], { }, /* */, if else
Pascal : ( ), [ ], begin end, { }
C, Java におけるif elseは緩やかな括弧構造で
elseを伴わないifも許される
$S \to$ if (C) S else S | if (C) S

パズル. 上記の文法はあいまいで
if (C) if (C) S else S
は複数の構文木（解釈）を持つ。それを示せ。

プログラミング言語（の骨格）はあいまいでないCFGで記述できる
elseが直前のifと対をなすことを反映した文法
$S \to$ if (C) S_{ie} else S | if (C) S | …(その他の規則)
$S_{ie} \to$ if (C) S_{ie} else S | …(その他の規則)

コンパイラコンパイラとは

コンパイラ
高級プログラミング言語で書かれたプログラムを
機械語に翻訳する

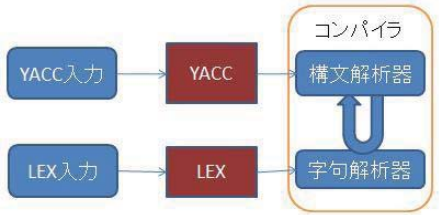
コンパイラコンパイラ
プログラミング言語の文法(CFG)と意味(動作)定義を
その言語のコンパイラに翻訳（自動生成）する

YACC

YACC (Yet Another Compiler Compiler) は
①言語の文法を与えるとその構文解析器（シフト還元機械：DPDAの一種）を出力
②言語の文法＋アクションを与えると
文法規則による還元を行うたびに対応するアクションを実行する
コンパイラーを出力

例. 簡単な電卓：Nは数値
$$は左辺の記号の値
$$n$$は右辺のn番目の記号の値

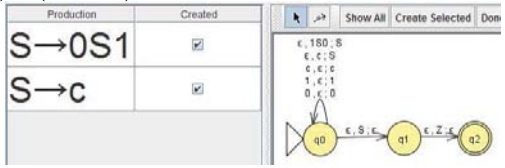
```
s : e '='      {$1を表示;}
e : '(' e ')'  {$$=$2;}
  | e '+' e    {$$=$1+$3;}
  | e '*' e    {$$=$1*$3;}
  | N          {$$=$1;}
  ;
```



簡易動作例：（残り入力, スタック）（スタックは便宜上右が上で表示）
$(2+3=,) \Rightarrow (+3=,2) \Rightarrow (+3=,e[2]) \Rightarrow (3=,e[2]'+') \Rightarrow (=,e[2]'+'3)$
$\Rightarrow (=,e[2]'+'e[3]) \Rightarrow (=,e[5]) \Rightarrow (,e[5]'=') \Rightarrow (,S[5\text{を表示}])$

YACCが生成するプッシュダウンオートマトン（構文解析器）

シフト還元機械：(D)PDAの(等価な)機能拡張。
スタックで記号列αを読み、まとめて取り除ける。
$S \overset{*}{\Rightarrow} \beta w \overset{*}{\Rightarrow} vw \iff (q,vw,Z) \overset{*}{\vdash} (q,w,\beta^R Z) \overset{*}{\vdash} (q,\varepsilon,SZ)$が成り立つように
①各入力記号aに対する
シフト動作ⓠ$\xrightarrow{a,\varepsilon/a}$ⓠと
②Gの各規則$A \to \alpha$に対する
還元動作ⓠ$\xrightarrow{\varepsilon,\alpha^R/A}$ⓠを持つ。
（同時に対応するアクションを実行）



例. JFLAPによるCFG⇒シフト還元機械：PDA(LR)とその動作例
$(q_0,0c1,Z) \overset{*}{\vdash} (q_0,1,c0Z) \vdash (q_0,1,S0Z) \vdash (q_0,\varepsilon,1S0Z) \vdash (q_0,\varepsilon,SZ) \overset{*}{\vdash} (q_2,\varepsilon,\varepsilon)$

注. YACCには決定性シフト還元機械を生成するための
様々な指定・制限があり、実はこれほど単純ではない。

マークアップ言語

HTML：Webページの記述言語、Hyper Text Markup Language
　タグ：<要素名>要素内容</要素名>の形をした括弧構造
　ブラウザがタグ情報を利用して要素内容を表示

HTML文書が従うCFG（の一部）：斜体は非終端記号(文法要素)
Char → a | A |・・・　（文字）
Text → ε|*Char Text*　（文字列）
Doc → ε|*Element Doc*　（要素列）
Element → *Text* | <EM>*Doc*</EM> | <OL>*List*</OL> | ・・・　（要素）
List → ε| <LI>*Doc*</LI> *List*　（OL要素：リスト）

XMLと文書型の定義

タグは要素の意味内容を示し、利用者が自由に定義可能で
　他のプログラムでタグ情報を利用
　　例．<地名>山崎</地名>、<姓>山崎</姓>
例．JFLAPファイルはXML文書

DTD（標準型定義）：グループ内で標準に使うXML文書を生成する(拡張)CFG

DTD
```
<!DOCTYPE PcSpecs [
 <!ELEMENT PCS    (PC*)>
 <!ELEMENT PC     (MODEL,PRICE,PROCESSOR,
                  RAM,DISK+)>
 <!ELEMENT MODEL (¥#PCDATA)>
 ・・・
]>
```

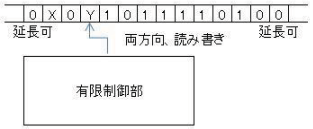
XML文書
```
<PCS> <PC>
 <MODEL>PC2016</MODEL> <PRICE>15,000</PRICE>
 ・・・
</PC> <PC>
 <MODEL>NOTE16</MODEL> <PRICE>5,000</PRICE>
 ・・・
</PC> </PCS>
```

（¥#PCDATA はタグに関係しないテキストデータを表す）

DTDに基づいて
　XML文書の表示指定：CSS、XSL
　XML文書の処理：要素の取り出し(SAX、DOM)→プログラミング言語

チューリング機械（TM）

入力テープが作業テープの役割も持ち
テープ上を行きつ戻りつしながら読み書きでき
テープを必要なら両方向にいくらでも延長できる



動作 $ⓠ \xrightarrow{X/YD} ⓟ$：状態qでテープ記号Xを読むと

1. テープ記号をYに書きかえ、ヘッドをD方向（左L／右R）に動かし
2. 状態pに遷移する

ただし、テープの両端から先は空白記号□で満たされていると仮定し

- テープの両端を必要に応じて左右に
  伸ばし（□を別のテープ記号に置き換え）たり
  縮め（□を端のセルに書い）たりでき、
- 最終状態では停止する（動作を持たない）

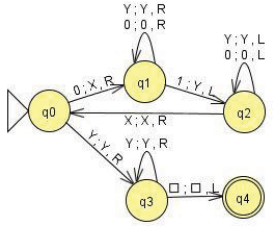
ものとする

動作と時点表示

時点表示 $\alpha q X \beta$：状態qでテープ$\alpha X \beta$のXを見ている状況
$ⓠ \xrightarrow{X/YR} ⓟ$なら$\alpha q X \beta \vdash \alpha Y p \beta$
　状態qでXを読むと、状態をpにXをYに換えて右へ移動
$ⓠ \xrightarrow{X/YL} ⓟ$なら$\alpha Z q X \beta \vdash \alpha p Z Y \beta$
　状態qでXを読むと、状態をpにXをYに換えて左へ移動

時点表示列：右上図のTM（JFLAPファイル）
　0をXに1をYに書き換え、同数なら最終状態q_4に遷移
$q_0 0011 \vdash X q_1 011 \vdash X 0 q_1 11 \vdash X q_2 0 Y 1 \vdash q_2 X 0 Y 1 \vdash X q_0 0 Y 1$
$\vdash X X q_1 Y 1 \vdash X X Y q_1 1 \vdash X X q_2 Y Y \vdash X q_2 X Y Y \vdash X X q_0 Y Y$
$\vdash X X Y q_3 Y \vdash X X Y Y q_3 □ \vdash X X Y q_4 Y$

シミュレータ（作者のページ）のプログラム（右図）



q_0, 0, X, R, q_1
q_0, Y, Y, R, q_3
q_1, 0, 0, R, q_1
q_1, Y, Y, R, q_1
q_1, 1, Y, L, q_2
q_2, 0, 0, L, q_2
q_2, Y, Y, L, q_2
q_2, X, X, R, q_0
q_3, Y, Y, R, q_3
q_3, BLANK, BLANK, L, q_accept

TMにおける認識と計算

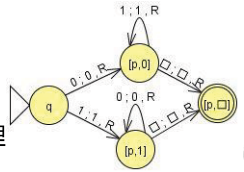
TMMにおいて、$▷○w \overset{*}{\vdash} \alpha◎\beta$とな（って停止する動作列があ）るとき、

1. Mはwを受理するといい、Mはそれが受理する文字列の集合を認識（受理）するという
　前頁のTMは$\{0^n1^n | n \in \mathbb{N}_+\}$を受理する
2. Mは入力wに対し、$\alpha\beta$を出力すると言い、その入出力関係を計算するという
　前ページのTMは入出力関係$\{(0^n1^n, X^nY^n) | n \in \mathbb{N}_+\}$を計算する。

TMMが認識する言語$L(M) = \{w | ▷○w \overset{*}{\vdash} \alpha◎\beta\}$
TMMが計算する入出力関係$R(M) = \{(w, \alpha\beta) | ▷○w \overset{*}{\vdash} \alpha◎\beta\}$

TMの技法

状態への記憶：$01^* \cup 10^*$を認識するTM（JFLAP）
　状態に最初の文字を記憶（[p, 0]、[p, 1]）し、[p, 0]（[p, 1]）で1（0）を連続して読み、右端に至れば受理



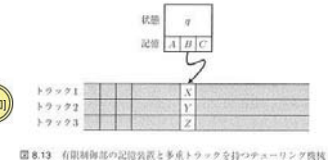
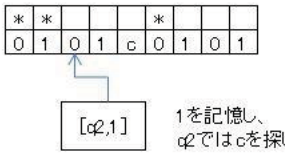


図 8.13 有限制御部の記憶装置と多重トラックを持つチューリング機械

多重トラック：$\{wcw | w \in \{0,1\}^*\}$を認識するTM
　セルの文字を状態に記憶し*印を付け右に移動
　cを超えて最初の未照合（*なしの）セルと照合し
　一致したらそのセルに*を付けて
　最も左の未照合セルに戻る

wcwを受理するTM、*は照合済みの印



TMの技法

サブルーチン：TMの埋め込み

Copy $(wq_1 0^n 10^k \overset{*}{\vdash} wq_5 0^n 10^{k+n})$ を利用して、乗算TMを実現する

$(q_0 0^m 10^n 1 \overset{*}{\vdash} 0^{m-i} 1 q_1 0^n 10^{(i-1)n} \overset{*}{\vdash} 0^{m-i} 1 q_5 0^n 10^{in} \overset{*}{\vdash} q_8 B 10^n 10^{mn} \overset{*}{\vdash} q_{12} 0^{mn})$

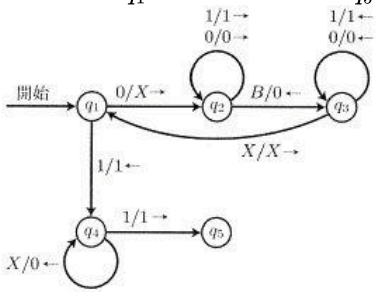


図 8.14 サブルーチン Copy

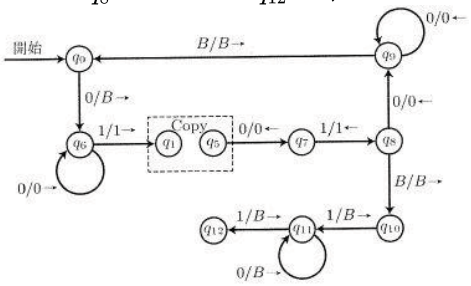


図 8.15 サブルーチン Copy を用いた乗算チューリング機械

TMの機能拡張

TMの能力を高めるわけではない

多テープチューリング機械：右図
1テープTMで、多重トラック技法を用いてシミュレートできる（最右図）

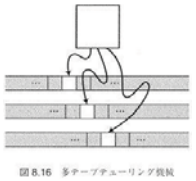


図 8.16 多テープチューリング機械

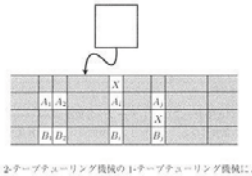


図 8.17 2-テープチューリング機械の 1-テープチューリング機械による模倣

万能チューリング機械 $\mathcal{U}$：右図
$\mathcal{U}$は入力テープ上に書かれたTM Mの動作規則と入力wから、
・Mのテープ内容を保持するテープ
・Mの状態を記録するテープ
・メモ用テープ
を使用してMの動きを模倣する
$\mathcal{U}$が認識する言語$= \{M\ w \mid M$はwを受理する$\}$
$\mathcal{U}$が計算する関係$= \{(M\ w, \alpha) \mid M$は$(w, \alpha)$を計算する$\}$

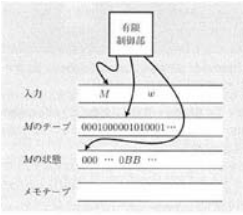


図 9.5 万能チューリング機械の構成

決定性TM（DTM）

状態とテープ記号から、動作が一意に決まるTM

定理．（非決定性）TM M_Nに対しそれを模倣する決定性TM M_Dが構成できる
∵）NTM M_Nの動きは、初期IDを根とする木（動作木）で表される。M_DはM_Nの時点表示(ID)の列を保持するテープ（キュー）を別に持ち、それを使って、M_Nの動作木を幅優先探索(動作ステップの少ない順の探索)する
（特に区別するときは非決定性TMを NTM、決定性TMを DTM という）

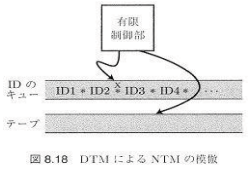


図 8.18 DTM による NTM の模倣

制限されたチューリング機械

定理．TMの機能を以下のように制限しても、能力は不変
1. 半無限テープのTM：テープは右側にしか延長できない
2. 2スタック機械：テープの代わりに２本のスタックを持つ
3. 2計数機械：スタックの記号が(底記号以外に)1種類のみの2スタック機械

パズル．2スタック機械でTMを模倣できることを示せ

句構造文法

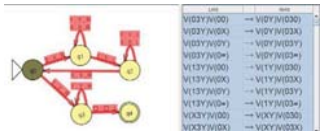
規則が$\alpha \to \beta$の形の文法（$\alpha \in (\Sigma \cup \Gamma)^* - \Sigma^*$, $\beta \in (\Sigma \cup \Gamma)^*$）
（Σ：終端アルファベット、Γ：非終端アルファベット）
即ち、規則の左辺αは非終端記号を1個は含む。

$\alpha \to \beta$による導出：$\gamma\alpha\delta \Rightarrow \gamma\beta\delta$
導出列：$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$のとき$\alpha_1 \overset{*}{\Rightarrow} \alpha_n$と書く。

文法Gが生成する言語：$L(G) = \{w \in \Sigma^* | S \overset{*}{\Rightarrow} w\}$（$S$は開始記号）

句構造文法＝TM

句構造文法G⇒TMMの証明のアイディア
Mは入力wに対し、Gによる導出の逆（還元）を（非決定的に）実行しSがテープ上に得られれば、受理する。すなわち
Gで規則$\alpha \to \beta$を使って$S \overset{*}{\Rightarrow} \gamma\alpha\delta \Rightarrow \gamma\beta\delta \overset{*}{\Rightarrow} w$のとき
Mで$q_0 w \overset{*}{\vdash} \gamma p \beta\delta \overset{*}{\vdash} \gamma\alpha q\delta \overset{*}{\vdash} q_f S$ となるようにMを構成する。
TMM⇒句構造文法Gの証明のアイディア
最初に最終状態を含む時点表示を生成し、動作を逆順に模倣して初期時点表示に戻し
最後に状態記号を消去する。すなわち
Mで$\textcircled{q} \xrightarrow{X/YR} \textcircled{p}$を使って
$q_0 w \overset{*}{\vdash} \alpha q X\beta \vdash \alpha Y p\beta \overset{*}{\vdash} \gamma q_f \delta$のとき
Gでは$S \overset{*}{\Rightarrow} \gamma q_f\delta \overset{*}{\Rightarrow} \alpha Y p\beta \Rightarrow \alpha q X\beta \overset{*}{\Rightarrow} q_0 w \Rightarrow w$
となるようにGを構成する



帰納的可算言語

定理．同値な次の３条件をみたす言語を帰納的可算言語という。
- 句構造文法（制限なし）で生成される
- （非決定性）チューリング機械で認識される
- 決定性チューリング機械で認識される

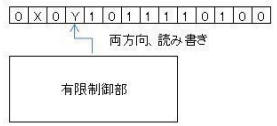
帰納的可算の意味：帰納法により数え（並べ）あげること（だけ）ができる

句構造文法Gが生成する言語は次のように帰納的に定義できる
1. 開始記号はGで導出される
2. Gで導出さた列に規則を適用した結果は、Gで導出される
3. Gで導出される終端記号列のみがGで生成される

パズル．１．帰納的可算言語は和、反転、準同型写像で閉じていることを示せ。
２．帰納的可算言語は決定性TMで認識されるが、補集合演算で閉じていない。何故か。

線形有界オートマトン

入力テープが同時に作業テープの役割を果たし、テープ上を行きつ戻りつしながら読み書きできる（が延長は不可）

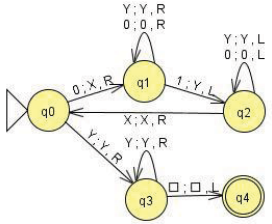


動作$\textcircled{q} \xrightarrow{X/YD} \textcircled{p}$：テープ記号$X$と状態$q$から
1. 状態をpに変え
2. マスの記号をテープ記号Yに書きかえ、ヘッドをD方向（右Rまたは左L）に動かす
ただし、テープの両端（の空白マス□）から飛び出すことはできない

コラム．「線形有界」は、使えるテープの長さが入力長の定数k倍（線形）という意味。使う記号を増やせばkマスを1マスで表現できるので、ここでの定義と同等。

動作と時点表示

時点表示 $\alpha q X \beta$：状態qでテープの$\alpha X \beta$のXを見ている
$q \xrightarrow{X/YR} p$ なら $\alpha q X \beta \vdash \alpha Y p \beta$（$X, Y \neq \square$）
$q \xrightarrow{X/YL} p$ なら $\alpha Z q X \beta \vdash \alpha p Z Y \beta$（
$Z \neq \square$, $X = \square \Rightarrow Y = \square$）
注．空白記号□は別のテープ記号に置き換えられない



時点表示列：$0^n 1^n$を受理する(決定性)LBA
$q_0 0011 \vdash X q_1 011 \vdash X 0 q_1 11 \vdash X q_2 0 Y 1 \vdash q_2 X 0 Y 1 \vdash X q_0 0 Y 1 \vdash X X q_1 Y 1 \vdash X X Y q_1 1$
$\vdash X X q_2 Y Y \vdash X q_2 X Y Y \vdash X X q_0 Y Y \vdash X X Y q_3 Y \vdash X X Y Y q_3 \square \vdash X X Y q_4 Y$

単調文法＝LBA

単調文法：規則が$\alpha \to \beta$の形の文法（$\alpha \in (\Sigma \cup \Gamma)^* - \Sigma^*$, $\beta \in (\Sigma \cup \Gamma)^*$, $|\alpha| \leqq |\beta|$）
ただし規則の右辺に現れない開始記号Sに限り$S \to \varepsilon$を許す
（Σ：終端アルファベット、Γ：非終端アルファベット）

証明は一般のTMMと句構造文法Gの等価性の証明と同様である。ただし、
1. LBAの時点表示のテープ長は入力wの長さであり
2. 単調文法でwの導出途中に現れる記号列はwの長さ以下である
ことに注意。

定理．同値な次の2条件を満たす言語を文脈依存言語という。
1. 単調文法で生成される
2. (非決定性)LBAで認識される

文脈依存言語の性質

定理．文脈依存言語は、集合演算(和、積、差、補集合)、反転、準同型写像のもとで閉じている。

文脈依存言語が補集合演算で閉じていることは、1988年に示された。
(非決定性)LBAが決定性LBAより真に能力が高いか否かは未解決問題である。

チューリング機械とコンピュータ

チューリング機械はコンピュータの数学的モデル

歴史
- 1936年：チューリングがチューリング機械を提唱（万能TMも）
  映画「イミテーションゲーム」
- 1945年：ノイマンがプログラム内蔵方式（ノイマン型コンピュータ）を提唱
- 1946年：最初のコンピュータENIAC
- 1949年：最初のノイマン型コンピュータEDVAC

パズル．EDVACの写真とENIACの写真に見られる大きな違いとその理由は何か

チューリング機械⇒コンピュータ

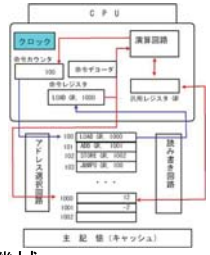
チューリング機械（TM）を模倣するコンピュータプログラムが存在する
有限制御部⇒プログラム
テープ⇒メモリー
TMシミュレータ（作者のページ）

コンピュータは万能TM。個々のTM⇔プログラム

チューリング機械のテープはいくらでも延長できるのに対し
　コンピュータのメモリは（ハードディスクを含めても）有限
　しかし、外部記憶装置は必要に応じていくらでも増やせ
　いくらでも大きなメモリー番地を利用できると仮定して(すれば)よい。

コンピュータ⇒チューリング機械

コンピュータ(CPU)の基本動作サイクル
1. 命令カウンタの値(番地)を読み取る
2. その番地にあるデータ(命令)を命令レジスタに移す
3. プログラムカウンタの値を１(命令サイズ分)増やす
4. 命令デコーダが命令レジスタ中の命令を解釈する
5. 演算装置等が解釈された命令を実行する



コンピュータをシミュレートする(多テープ・万能)チューリング機械
　有限制御部記憶：命令(番地部を除く)
　記憶テープ：コンピュータのメモリー、プログラムとデータ
　命令カウンタテープ：命令カウンタの内容
　記憶番地テープ：命令中の番地とその内容
　入力ファイル用テープ：コンピュータへの入力を保持
　メモテープ：作業用テープ、汎用レジスタ

チャーチ・チューリングのテーゼ

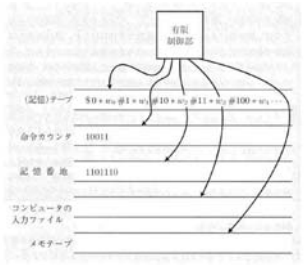


図 8.22 コンピュータを模倣するチューリング機械

１９３０年代、様々な計算モデルが提唱され、それらの計算能力が等しいことが証明された
- チューリング機械
- 帰納的関数
- λ計算
- ・・・
- コンピュータ

チャーチ・チューリングのテーゼ
これらのモデルで計算できる⇔コンピュータで計算できる
⇔一般的な意味で計算できる

以後、コンピュータ(プログラム)に関する直感に基づいて議論を進める。

パズル．
１．プログラムが真の(人工)知能だとみなせる条件は何か(チューリングテスト)。
２．数学的に明確に述べられる問題は、(いずれ数学が進歩すれば)必ず解けるか。

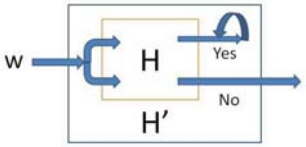
計算不可能な問題

それを解くプログラムが存在しない判定(yes/no)問題

プログラムの停止問題：
入力：プログラムPとその入力w
出力：動作させたとき$P(w)$がいつかは停止(yes)するか否(no)か

定理．停止問題を解くプログラムは存在しない。
略証．停止問題を解くプログラム$H(P,w)$から次の$H'(w)$のプログラムが作れる。
- $H'(w)$は停止しない⇔$H(w,w)$ =yes
- $H'(w)$ =no（停止する）⇔$H(w,w)$ =no

$H'(H')$ =no⇔$H(H',H')$ =no⇔$H'(H')$は非停止
⇔$H(H',H')$ =yes⇔$H'(H')$は停止
となって、矛盾



パズル．「プログラムPを入力wの下で実際に走らせる」こと（万能TM $\mathcal{U}$）では停止問題を解くことはできない。何故か？

ライスの定理

問題：問題例が無数にあり、答えがyes/noであるもの
自明な問題：答えがすべてyes、あるいはすべてnoである問題
プログラムの働きが等しい：
同じ入力に対し(停止しないことも含めて)同じ出力になる
プログラムの働きに関する問題：
働きが等しいプログラムに対しては答(yes/no)が等しくなる問題

定理．プログラムの働きに関する問題は自明なものを除いてすべて計算不能

ライスの定理の証明

定理．プログラムの働きに関する問題は自明なものを除いてすべて計算不能
略証．プログラムの働きに関する自明でない問題を考え、それを解くプログラムMがあるとする。今Mは、決して停止しないプログラムP_Nに対しNoを返し、あるプログラムP_Yに対しYesを返すものとする。

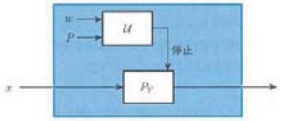


図 6.1 M'の設計図

プログラムMから停止問題を解くプログラム$H(P,w)$が作れることを示す。Hは入力として(P,w)を受け取ったら、図に示したM'のプログラムを作成し、Mに読み込ませる。（図中の$\mathcal{U}$は(P,w)を受け取り$P(w)$の動作を模倣する万能プログラム）
するとMにM' を読み込ませた結果は
$P(w)$が停止する　⇒M'はP_Yに働きが等しい⇒Yes
$P(w)$が停止しない⇒M'はP_Nに働きが等しい⇒No
になる。よって、$P(w)$が停止するか否か判定できることになり、矛盾する。

解けない問題

Postの対応問題
与えれた文字列の対(の集合)を並べて、同じ文字列対が作れるか否か。

	解を持つ		解を持たない		
i	x_i	y_i	i	x_i	y_i
1	1	111	1	10	101
2	10111	10	2	011	11
3	10	0	3	101	011

ヒルベルトの第10問題
整係数の多変数多項式の方程式(ディオファントス方程式)が整数解を持つか否か。
$x^2+y^2=z^2$　$(0,0,0),(3,4,5),(5,12,13),\cdots$
$(x^2+1)^3+(y^2+1)^3=z^3$　整数解は存在しない

パズル．ポストの対応問題について、それぞれ解の有無を示せ。

プログラム（TM）の計算量

サイズ（長さ）nの入力データの計算に必要な領域$S(n)$や時間$T(n)$

線形有界オートマトン：線形$(S(n) \leqq kn)$の計算領域を持つTM

多項式時間（非）決定性TM（コンピュータプログラム）
$T(n) \leqq$ (nの多項式) である（非）決定性TM（コンピュータプログラム）

時間計算量：計算時間の見積もり

べき乗計算x^n：入力サイズはnの表現の長さ（桁数）
素朴な方法：n回の掛け算⇒$n \fallingdotseq 10^{(n\text{の桁数})}$に比例
高速計算法：$2\log n$ 回の掛け算⇒nの桁数に比例

整列：n個のデータを大きさの順に並べ替える
選択法：n^2に比例
クイックソート：平均的には $n\log n$ に比例

表 6.1 関数値とサイズ 10 の計算時間を単位（ミリ秒）とした計算時間

n	1	2	4	8	10	100	10^3	10^4
					1 ミリ秒	10 ミリ秒	0.1 秒	1 秒
n^2	1	4	16	64	100	10^4	10^6	10^8
					1 ミリ秒	0.1 秒	10 秒	16 分 40 秒
$\log_2 n$	0	1	2	3	3.322	6.644	9.966	13.288
					1 ミリ秒	2 ミリ秒	3 ミリ秒	4 ミリ秒
$n\log_2 n$	0	2	8	24	33.22	664.4	9966	132877
					1 ミリ秒	20 ミリ秒	0.3 秒	4 秒
2^n	2	4	16	256	1024	1.268×10^{30}	1.072×10^{301}	1.995×10^{3010}
					1 ミリ秒	3.9×10^{16} 年	—	—

ベキ乗y^xの計算時間

素朴なベキ乗計算法
$y, y^2, y^3, \cdots, y^x$ とyを順に掛けていく
$x-1$回の掛け算⇒ x(の表示サイズの指数乗)に比例する計算時間
⇒xの桁数が増えると実際には計算できない
高速ベキ乗計算　Binary法
　例．$y^{22} = y^{16+4+2} = y^{16}y^4y^2$、$y^2, y^4, y^8, y^{16}$の計算に乗算4回なので、乗算計6回
y^xの高速計算は、xが2進表示でk桁のとき、$y^1, y^2, y^4, \cdots, y^{2^{k-1}}$ の系列を計算するのに乗算$k-1$回、これらを組み合わせてy^xを計算するのに(最大)$k-1$回の乗算、計$2(k-1)$回の乗算で計算できる。2進10桁がほぼ10進3桁（$2^{10} = 1024 \fallingdotseq 10^3$）なので、高速ベキ乗は$x$の10進桁数$\log_{10} x$に比例する(約20/3倍)時間(乗算回数)で計算できる。

実験．$y^x \bmod m$の計算時間をMOD電卓で試してみよう
【y^x】ボタンは1にyをx回掛けてy^xを計算
【速y^x】ボタンは高速ベキ乗計算

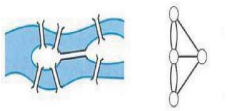
整列(ソーティング)

n個のデータを大きさの順に並べ替える
選択ソート
「未整列部(赤字)の最大値を選択して右端に移動させる(青字)」を繰り返す。
計算時間は$n(n+1)/2$に比例
クイックソート
「未整列部([赤字])を、右端の値を基準に、それ以下([赤字])とそれ以上([赤字])に分割」を繰り返す。
平均計算時間は$n\log n$に比例（計算例で赤字部分の長さの総和）

実験．整列法比較アプリで計算時間を比較してみよう。データ数が、20以下であれば整列法のデモが見られ、20より大なら計算時間だけが確認できる。

グラフのオイラー路

(無向)グラフ：頂点（○）とそれを結ぶ辺（—）からなる
オイラー(閉)路：各辺を1度ずつ通る(閉じた)路



オイラーの定理
オイラー閉路(路)がある⇒奇数本の辺を持つ頂点が0(0または2)個
∵）路は、始点と終点の1本の辺を使い、中間頂点の2本(入りと出)の辺を使う

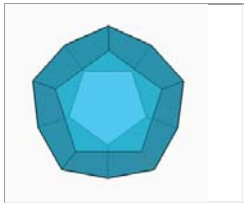
与えられたグラフにオイラー路があるかという問題は、オイラーの定理とその逆も成り立つので、各頂点の辺の数を数えるだけで解ける。

パズル：ケーニヒスベルグの7つの橋（右上図）を一度ずつ通る路はあるか。

ウィキペディア

グラフのハミルトン路

ハミルトンの世界一周パズル
ハミルトンは1857年に正12面体の各頂点を1度ずつ通る周遊路を求めるパズルを作成した。
ハミルトン(閉)路：
　各頂点を1度ずつ通る(閉)路

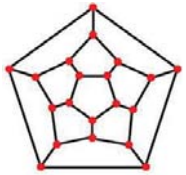


パズル：右のグラフのハミルトン閉路を求めよ

Wikipedia

ハミルトン路の解法には
　オイラー路のような簡明な判定法がなく
　虱潰しの試行錯誤しかないので
頂点数の階乗時間かかると信じられている
どうしてそんなことが言えるのか？？
「信じられている」とはどういう意味か？？

手に負えない問題



プログラムが実行可能：計算時間が問題(入力)のサイズの多項式で抑えられる
　高速べき乗、選択法、クイックソート、オイラー路：実行可能
　素朴べき乗、ハミルトン路(虱潰し法)　　：実行不能

手におえない問題：実行可能なプログラムが存在しない問題

以後、問題をいくらでも大きなサイズの問題例がある判定(yes/no)問題に限定する。

パズル．多項式時間プログラムを実行可能なプログラムとすることの是非を論ぜよ。

非決定性プログラム

非決定性プログラム：非決定性命令を含むプログラム
　非決定性命令：次の動作を有限個の候補の中から選べる
　計算：次の動作をうまく選んだ時に正しく計算できればよい
　　⇔可能な計算の中に正しく計算するものがあればよい
　計算時間：正しく計算するものの中での最短時間としてよい
但し、正答がnoのときにyesと答えるような事はいけない。
　非決定性判定問題プログラム（TM）の計算時間

正答	出力yesの場合	出力noの場合	計算時間
yes	あり	あってもよい	出力yesの最短時間
no	なし	あってもよい	（除外してよい）

定理．非決定性プログラムで$T(n)$時間で計算できる判定問題は、
決定性プログラムで$a^{T(n)}$時間(aは選択肢数の最大値)で計算できる。
略証：NTMのDTMによる模倣同様、高々T(n)時間以内のすべての計算（選択肢）を虱潰しで調べればよい。

$\mathcal{P}$, $\mathcal{NP}$

クラス$\mathcal{P}$: 多項式時間決定性プログラム(DTM)で解ける問題
クラス$\mathcal{NP}$：多項式時間非決定性プログラム(NTM)で解ける問題

$\mathcal{P} \subseteq \mathcal{NP}$。$\mathcal{P} \neq \mathcal{NP}$か否かは有名な未解決問題（ミレニアム問題）

グラフのオイラー路存在判定問題$\in \mathcal{P}$　∵)オイラーの定理があるから
グラフのハミルトン路存在判定問題$\in \mathcal{NP}$
- ハミルトン路を非決定的に選択できれば、それがハミルトン路であることの検証は多項式時間で可能
- $\mathcal{P}$に属すか否かは知られていない

$\mathcal{NP}$問題の別表現
証拠が与えられれば、その答えがYesであることを、**問題(入力)サイズの**多項式時間内に検証できる問題　（**注意：証拠のサイズではない！**）

パズル．１．$\mathcal{NP}$は何の略か。
２．上の注意にあるが、証拠のサイズでよければ停止問題は$\mathcal{NP}$であることを示せ。

$\mathcal{NP}$完全問題

$\mathcal{NP}$問題Pが$\mathcal{NP}$完全：問題Pを解くプログラムM_Pから、すべての$\mathcal{NP}$問題Qに対して、Qを解くプログラムM_Qを多項式時間で作れる

$\mathcal{NP}$完全問題のどれか１つでも$\mathcal{P}$問題であれば、$\mathcal{P} = \mathcal{NP}$
つまり、$\mathcal{NP}$問題の中で、もっとも難しいと考えられる問題

Cookは充足可能性判定問題が$\mathcal{NP}$完全であることを示した(1971)。

充足可能性判定問題
与えられた論理式の値を真にする変数値の割り当てが存在するか否か判定する問題

パズル．$\mathcal{P} = \mathcal{NP}$であれば、どのようなことが言えるか。

証明のアイディア

M_P：$\mathcal{NP}$問題Pを解く非決定性多項式$p(n)$時間チューリング機械
　M_Pはサイズnの入力xに対し、$N-1 := p(n)$ステップで問題Pの答$P(x)$ =yes/noを(例えば)最初のセルに書き込んで停止する
M_Pの入力xに対する動作列を記述し答がyesかnoかを判定する論理式$L(x)$をNの多項式時間で作り、$L(x)$が充足可能⇔$P(x)$ =yes　となるようにする

変数	=真　の意味	個数
S_{tq}	時点tのM_Pの内部状態がq	N×状態数
T_{tka}	時点tのテープセルkの内容が記号a	N^2×記号数
H_{tk}	時点tのヘッド位置がk	N^2

これらの変数に対する真偽の割り当てが、M_Pの正しい計算になって(動作規則を満たして)いて出力がyesのときにのみ真となるような論理式をつくればよい。

$\mathcal{NP}$完全問題の例

充足性判定問題
ハミルトン(閉)路問題：与えれれたグラフがハミルトン(閉)路を持つか
巡回セールスマン問題：与えれた都市をすべてめぐるコストw以下の路があるか
3彩色問題：与えれたグラフの頂点を3色で塗り分けることができるか

$\mathcal{P}$問題ではなさそうな$\mathcal{NP}$問題
素因数分解：与えられた数の素因数分解を求める
（素数判定は最近(2002年)$\mathcal{P}$問題であることが示された）

公開鍵暗号

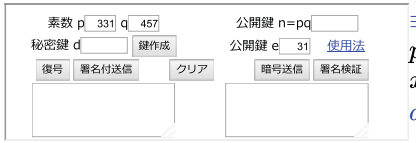
送信者　　通信路　　受信者
平文[暗号化]⇒　暗号　⇒[復号]平文

共通鍵暗号：送信者と受信者で実質的に同じ鍵を用いる
　鍵(と手順)を秘密裡に共有し(ネット上で可能？)
　送られた暗号をその鍵で復号
　例 IBM[1文字前]⇒ HAL ⇒[1文字後]IBM

公開鍵暗号：送信者(暗号化)と受信者(復号)で異なる鍵
　受信者が暗号化の鍵(と手順)を公開
　送られた暗号を秘密にしている鍵で復号
　公開(暗号化)鍵から秘密(復号)鍵が分からないのか？
　　RSA暗号、楕円曲線暗号

パズル．通信路の盗聴者が、公開鍵暗号を破るための方法を考えよ。

RSA公開鍵暗号の作成実験



$\equiv_n$：法nで（nで割った余りが）等しい
p,qが素数のとき
$x \equiv_{pq} x^{1+(p-1)(q-1)} \equiv_{pq} x^{1+2(p-1)(q-1)} \equiv_{pq} \cdots$より
$de \equiv_{(p-1)(q-1)} 1 \Rightarrow (x^d)^e = (x^e)^d = x^{de} \equiv_{pq} x$

$n = pq$を法(mod)としてd乗とe乗が互いに逆関数⇒暗号化と復号に利用

パズル．上のアプリと右のmod電卓を使って以下を確かめよ
1. 「暗号送信」と「複号」がうまく働く
2. 実験で使った e,p,q から、$d := 1 \div e \mod (p-1)(q-1)$で$d$が求まる
3. 実験で使った e,d,n に対し、適当なxで$(x^d)^e \equiv_n (x^e)^d \equiv_n x$となる

RSA暗号の安全性

RSA暗号を破る（考えうる）方法：公開鍵(e,n)は既知
1. 暗号(の値)を、法nで1乗、2乗、3乗、…して元に戻るか否かを調べる。d乗で元に戻るから、秘密鍵dをいつかは知ることができる
2. nの素因数分解でp,qを求め、法$(p-1)(q-1)$で$1 \div e$を計算してdを求める
3. 平文(の一部)xを推測し、法nでe乗して盗聴暗号(の一部)yと一致するか見る

注．前頁の実験は簡単のため換字暗号になっており、3. に対しては安全でない。

すべて虱潰しの方法であり、素朴ベキ乗計算の実習でみたように、nの値や秘密鍵e、素数p,qを十分おおきくすれば安全と考えてよい。
　ところで、x^dの計算は復号計算でもあるから、その計算にxを$d-1$回掛ける素朴な方法しかなければ、復号にとんでもない時間がかかってしまい、RSA暗号は機能しない。幸い、dの値がわかっていれば、d乗はBinary法で高速計算できる

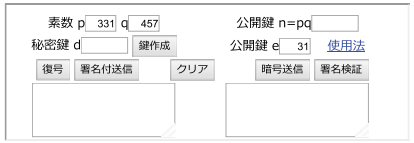
暗号の安全性と$\mathcal{P} \neq \mathcal{NP}$

　以上のことから、復号は、秘密鍵dを知っている本人は高速実行できるが、たとえ暗号を入手(盗聴)できたとしても、秘密鍵を知らない第3者が可能な素朴な方法では、現実的な時間内に実行できない ことがわかる。
　現在のところRSA暗号を現実的な時間内に破る方法は知られていない。
１．について言えば、与えられた x,y,nに対し、$x =_n y^d$を満たすdを求める効率的な方法は知られていないし、
２．についていえば、nからp,qを求める効率的な因数分解の方法は知られていない。

　しかし、何か巧妙な方法が、まだ見つかっていないだけで、今後見つかる可能性が0とは今のところわかっていない。そのような方法は存在しえず、それゆえRSA暗号は安全であると広く信じられてはいるが、そのことの数学的に厳密な証明は得られていない。
　このことは、$\mathcal{P} \neq \mathcal{NP}$問題というクレイ数学研究所から100万ドルの賞金がかけられている有名な未解決問題の一つに関わる問題である。

電子署名

署名＝文書を秘密鍵で暗号化したもの
文書＋署名の受取人は公開鍵で署名を復号して
文書との一致を検証する



公開鍵で復号できる暗号(＝署名)を作れるのは秘密鍵の持ち主だけ
⇒文書の改ざん、署名の偽造(なりすまし)が防げる

パズル. 右上のアプリを使って、以下のことを確かめよ
1. 「署名付き送信」と「署名検証」がうまく働く
2. 「署名検証」は文書の改竄を発見できる
3. 「署名検証」は署名の偽造を発見できる

署名者は、秘密(鍵)を明かさずに本人（秘密(鍵)の持ち主）であることを証明
⇔ゼロ知識証明

マジックプロトロコル

コイントス：A,B 両者が信頼できる第三者を介さずに通信線上で行う
- AはBに投げたコインの情報を送信し（以後Aはコインの裏表を変えられない）
- Bは裏表を当てる（Bが得た「情報」からはコインの裏表が分からない）
- Bが答えた後、その当否を両者(特にB)が確認できる

ことを実現する
コイントスプロトコル(通信手順)：Aの公開鍵を利用
1. Aは乱数 r をBに送り、r のAによる署名の偶奇をBに問う
2. Bは偶奇を答える
3. Aは r の署名をBに送り、Bは(Aの公開鍵で)それを検証する

パズル. 上のプロトコルがうまく働くことを示せ

他のマジックプロトコル：いずれも通信線上で第三者を介さずに行う
金持ち比べ：互いの資産額を知らせずに、その大小を比べる
カード配り：トランプのカードを配る
電子投票：投票の秘密を保って投・開票を行う
参考：情報セキュリティの科学