言語理論及びコンパイラ

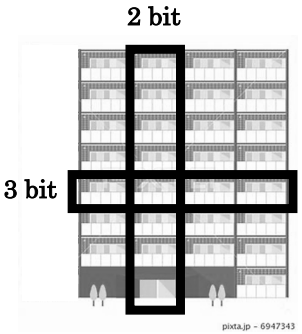
オートマトンと言語理論

情報科学の基礎

1. 情報理論
   情報の質と量
2. 論理代数
   論理の設計と検証
3. 言語理論
   コンピュータの動作

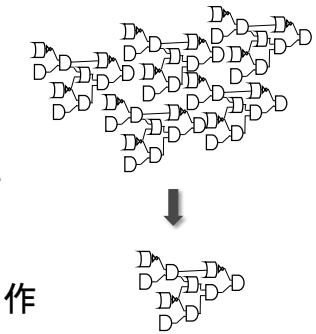
情報科学の基礎

1. 情報理論
   情報の質と量
2. 論理代数
   論理の設計と検証
3. 言語理論
   コンピュータの動作

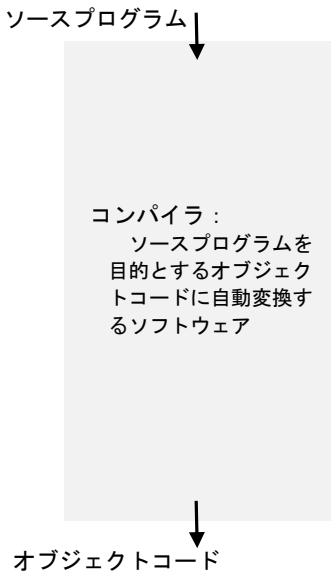


情報科学の基礎

1. 情報理論
   情報の質と量
2. 論理代数
   論理の設計と検証
3. 言語理論
   コンピュータの動作

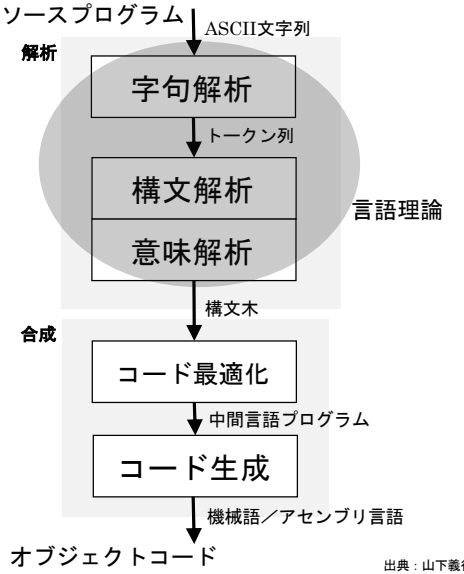


コンパイラ



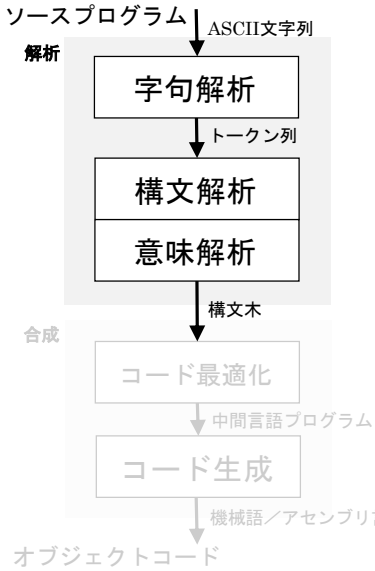
出典：山下義行，「コンパイラ入門」サイエンス社，2008

コンパイラの内部構造

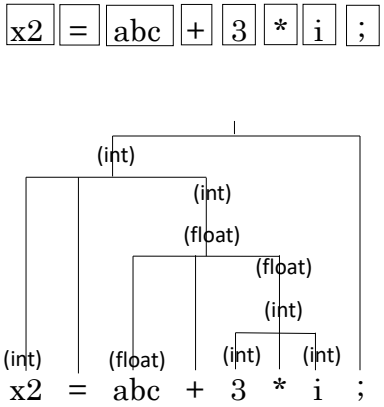


出典：山下義行，「コンパイラ入門」サイエンス社，2008

コンパイラの内部構造



x2=abc+3*i;



出典：山下義行，「コンパイラ入門」サイエンス社，2008

授業内容（言語理論）

1. 計算と言語
2. コンピュータが実行する計算
3. コンピュータが受理する言語

授業のポイント（言語理論）

1. 計算とはなにか，が理解できること
   オートマトンで計算を定義

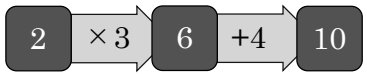
2. 言語とはなにか，が理解できること
   オートマトンと形式文法で言語を定義

1. 計算と言語
2. コンピュータが実行する計算
3. コンピュータが受理する言語

計算の例

① 数値計算

$2 \times 3+4=10$



データ　入力

	+1	+2	+3	+4	+5
1	2	3	4	5	6
2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10

	*1	*2	*3	*4	*5
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25

参考：時原，オートマトンと言語理論，森北出版，2015

計算の例

② かな変換

tokei ⟶ とけい

データ　入力

	a	i	u	e	o	k	s	t	n
ひらがな	あ	い	う	え	お	k	s	t	n
k	か	き	く	け	こ				
s	さ	し	す	せ	そ				
t	た	ち	つ	て	と				
n	な	に	ぬ	ね	の				

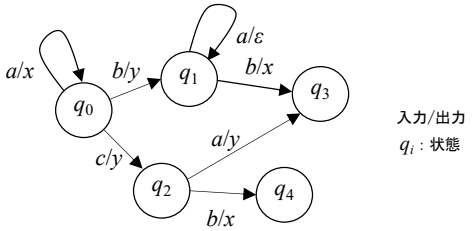
参考：時原，オートマトンと言語理論，森北出版，2015

計算とは

コンピュータで行う計算とは，保存されているデータと与えられた入力によって，データが書き換えられる操作を言う．

オートマトン

(1) オートマトンは，コンピュータで行う計算を単純化・抽象化した数学モデルである．
(2) オートマトンは，計算の仕組みを入力，出力，状態で記述する．

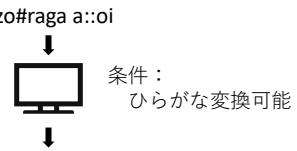


オートマトンを表す状態遷移図

変換機械と認識機械

オートマトンは，次の2つに分類できる．
変換機械：入力が加えられるたびに，対応する出力を行うオートマトン
認識機械：入力終了時に，入力列が条件を満たすかどうかを判断するオートマトン

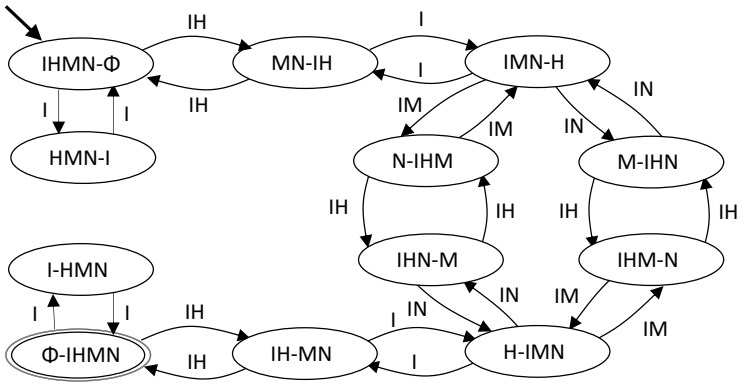
変換機械



認識機械

オートマトンの例

問題：私(I)，彼(H)，みさえ(M)，なつみ(N)が島に取り残された．そこには2人乗りのボートがあるが，運転できるのは私だけ．全員を岸まで移動させたいが，彼は浮気っぽいので，彼とみさえ，彼となつみを2人だけにするのは嫌．どのように全員を岸まで運べばよいか．



状態：島ー岸にいる人物

参考：宮野，東京大学理学部情報学科，授業紹介「形式言語理論」，2017

言語の例

自然言語

社会のなかで自然に発生し用いられている言語

■日本語，英語，・・・

“ 私はリンゴが好き ”

人工言語

明確な目的のために人為的に作られた言語

■プログラミング言語，エスペラント，・・・

“ X=rand()%XMAX ”

言語とは

言語とは，特定の条件を満たす記号列の集合を言う．

記号集合：言語を構成する最小要素の集合

$\sum = \{$あ，い，う，・・・，私，・・・$\}$

$\sum = \{ a, b, c,$ ・・・， $+, -,$ ・・・$\}$

記号列：記号集合に含まれる記号を並べた列

■特定の条件をどう定めるかが重要

形式文法

記号列を別の記号列に書き換える規則の集まりを形式文法と言う．書き換え規則は

$$A \rightarrow BC$$

の形をした規則で，これは記号Aを記号列BCに書き換えることを示す．形式文法で生成される言語を形式言語という．

形式文法の例

① 簡単な英語文

〈文〉 → 〈主語〉〈動詞〉〈目的語〉 (1)
〈主語〉 → I (2)
〈動詞〉 → like (3)
〈動詞〉 → play (4)
〈目的語〉 → tennis (5)
〈目的語〉 → soccer (6)

適用 (1)(2)(4)(5) ： I play tennis

形式文法の例

② 簡単な数式

〈数式〉→〈数式〉〈演算子〉〈数式〉　(1)
〈数式〉 →　(〈数式〉)　(2)
〈数式〉 →　“数”　(3)
〈演算子〉 →　＋　(4)
〈演算子〉 →　－　(5)

適用 (1)(2)(1)(3)(4)(5) ：“数”＋(“数”－“数”)

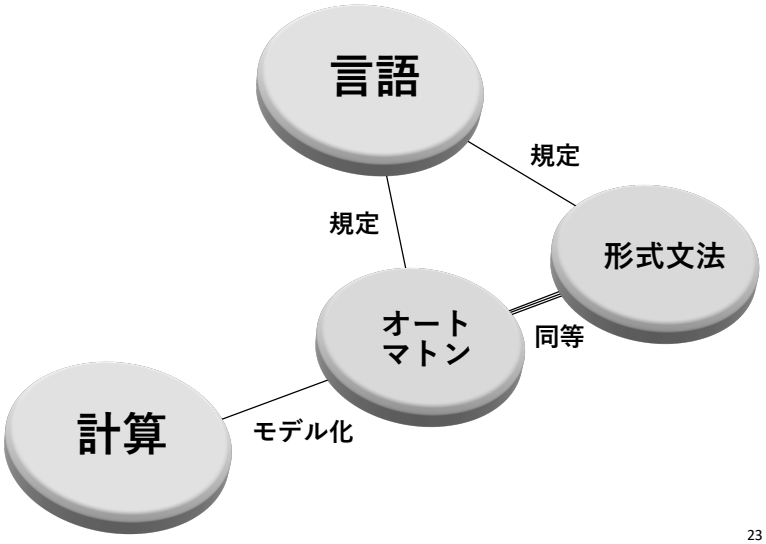
オートマトンと形式文法

■オートマトンと形式文法はいずれも言語を定める道具である.

■オートマトンと形式文法は表現形式が異なるだけで，言語を定めるうえでは全く同じ能力をもつ.

■オートマトンと形式文法にはいくつかの階層が存在する.

復習

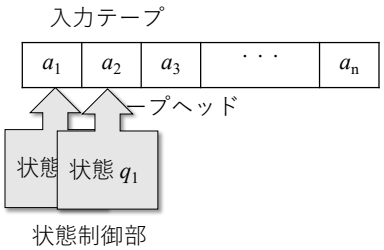


1. 計算と言語
2. コンピュータが実行する計算
3. コンピュータが受理する言語

有限オートマトン

■状態数が有限な認識機械

■入力列によって状態が変化し，入力終了時に yes または no を出力



有限オートマトンの定義

$M = (Q, \Sigma, \delta, q_0, F)$

Q：状態の集合
Σ：入力記号の集合
δ：状態と入力から次の状態を定める関数
q_0：初期状態
F：受理状態の集合

認識する言語

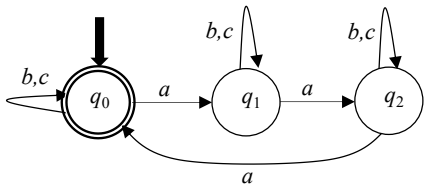
■オートマトンが受理する記号列
認識機械としてのオートマトンにおいて，出力が yes となる記号列

■オートマトンが認識する言語
特定の条件を表すオートマトンが受理する記号列の集合

有限オートマトンの例

① 記号の個数を認識
0個または3の倍数個の a を含む記号列のみを受理するオートマトン

■受理される記号列：*abbbaa, abacca, aaa, bbb, bbcbb*
■受理されない記号列：*aabaa, abaaacc, bbbac, bab*



参考：時原，オートマトンと言語理論，森北出版，2015

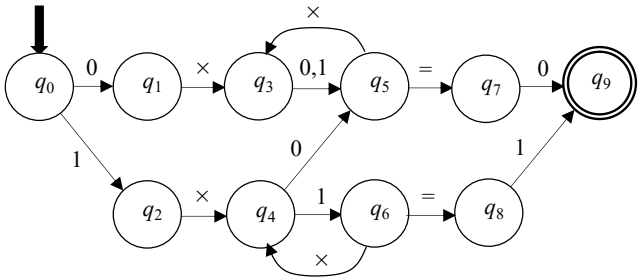
有限オートマトンの例

② 単純な掛け算を認識

0と1の掛け算の式を入力し，正しい等式のみを受理するオートマトン

■受理される記号列：0×0=0, 0×1=0, 1×1×1=1

■受理されない記号列：0×0=1, 1×0×1=1, 1×0= =0



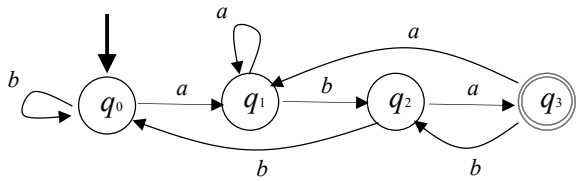
参考：時原，オートマトンと言語理論，森北出版，2015.

追加：有限オートマトンの作り方

■ 特定の記号列を検索するオートマトン

例：*a,b*から構成される記号列から“*aba*”を検出

babaabbabbabab



q_0：初期状態　　q_1：最初の a が入力

q_2：a に続いて b が入力　　q_3：b に続いて最後の a が入力

拡張された有限オートマトン

(1) 非決定性有限オートマトン

状態と入力の１つの組に対して，複数の状態遷移が定められる有限オートマトン

決定性有限オートマトン
非決定性有限オートマトンに対して，これまでのオートマトンを決定性有限オートマトンと言う.

(2) 空動作をもつ有限オートマトン

何も入力がないのに状態遷移が発生することもある有限オートマトン

有限オートマトンの等価性

非決定性有限オートマトンは，同じ言語を認識する決定性有限オートマトンに変換できる.
同様に，空動作をもつ有限オートマトンも，同じ言語を認識する決定性有限オートマトンに変換できる.

したがって，2つの拡張された有限オートマトンは表現方法が異なるだけで，いずれも決定性有限オートマトンと等価である.

正規表現（有限オートマトンの応用）

■正規表現は，特定の条件をもつ記号列の集合を，一つの記号列で表す簡便な表現方法

■正規表現は，プログラミング言語における字句の定義や検索する文字の表現などに活用

■任意の正規表現が表す言語を受理する有限オートマトンが存在．逆に，任意の有限オートマトンに対して，その受理言語を表す正規表現が存在．このため，正規表現は有限オートマトンのコンピュータへの応用の一つ

有限オートマトンの限界

有限オートマトンの能力は限られており，多くの認識できない言語が存在する．

■有限オートマトンが認識できない言語の例

① 正しい括弧の記号列

入力記号集合Σ={(,)}に対して，左括弧(と右括弧)の正しい組合せからなる記号列のみからなる言語

含まれる記号列：(()), ()(()), ()()()

含まれない記号列：((), ())(), ((()

有限オートマトンの限界

■有限オートマトンが認識できない言語の例

② 回文

Σ={*a*,*b*}に対して，右から読んでも左から読んでも同じ記号列のみからなる言語

含まれる記号列：*aba, aabaa, ababa*

含まれない記号列：*ab, ababab, aaabbb*

③ 同じ文字が同数並ぶ記号列

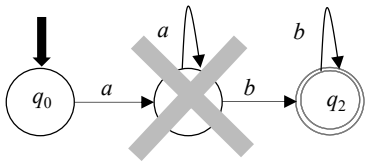
Σ={*a*,*b*}に対して，*a*と*b*が同数連続して並ぶ記号列のみからなる言語

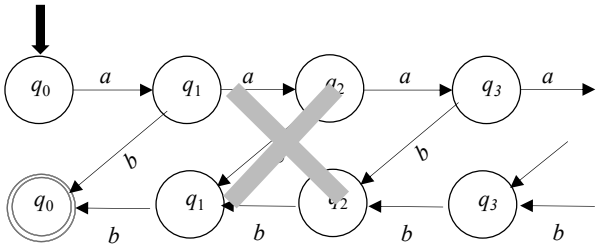
含まれる記号列：*ab, aaabbb*

含まれない記号列：*a, abb, aaabb*

追加：有限オートマトンの限界

同じ文字が同数並ぶ記号列



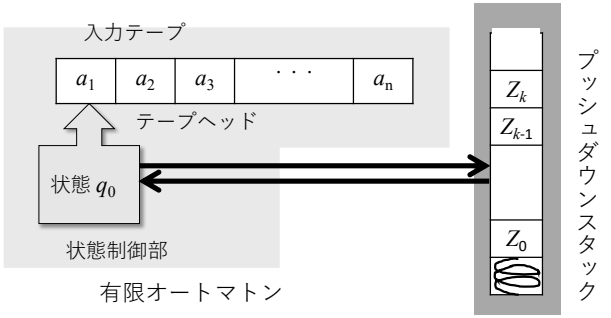


有限オートマトンの限界

■有限オートマトンが①，②，③の言語を認識できない理由
①においては左括弧(の個数を，②においては左から全体の長さの1/2の記号列を，③においてはaの個数を記憶しなければならない．しかし，有限オートマトンには，コンピュータのメモリに相当する記憶装置が備えられていない．これが有限オートマトンの限界である．

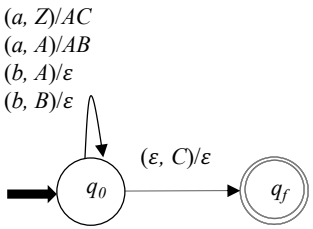
プッシュダウンオートマトン

■プッシュダウンオートマトンは有限オートマトン（CPUに対応）にプッシュダウンスタック（メモリに対応）を付けた機械
■その動作は，現在の状態，入力およびスタックの先頭から削除された記号によって定まる新たな状態に遷移し，スタックの先頭に新たにいくつかの記号を書き込む．



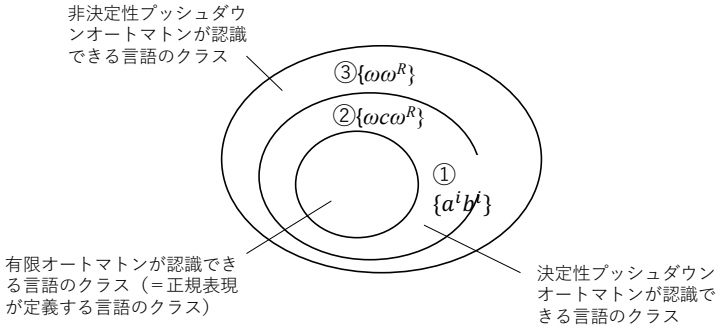
追加：プッシュダウンマトンの活用

同じ文字が同数並ぶ記号列



プッシュダウンオートマトンの言語認識能力

■プッシュダウンオートマトンは，有限オートマトンの場合と異なり，動作の決定性・非決定性により，言語認識能力に差がある．



プッシュダウンオートマトンの限界

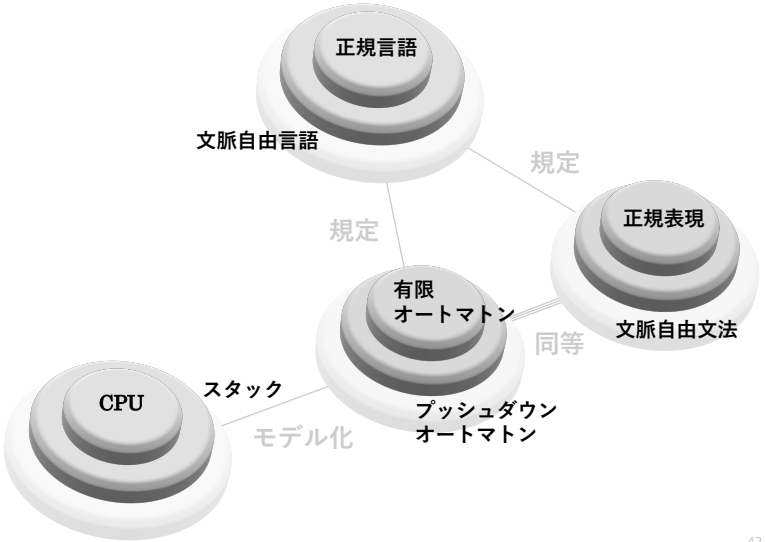
■プッシュダウンオートマトンは有限オートマトンより能力は優れているが，それでも認識できない多くの言語が存在する．

認識できない言語の例

$L_1=\{a^ib^ic^i \mid i\geq 1\}$, $L_2=\{a^{2^i} \mid i\geq 0\}$, $L_3=\{\omega\omega \mid \omega\in\{a,b\}^*\}$

■プッシュダウンオートマトンの限界は，直感的には，先頭のデータしか利用できないため，下に保存されているデータや同時に複数のデータを利用できないことにある．

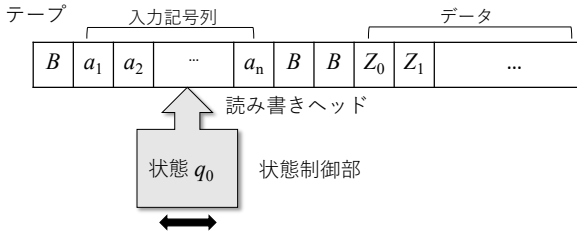
復習



チューリング機械

■チューリング機械は，状態制御部，入力記号列とデータの双方を収めた1本の半無限テープ，およびテープに対する読み出し・書き込みを行う読み書きヘッドから構成される．

■チューリング機械はプッシュダウンオートマトンと異なり，テープの任意の場所からデータを読み出し，任意の場所にデータを書き込むことができる．



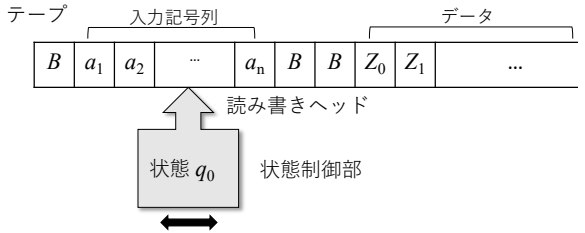
チューリング機械

■チューリング機械は次の動作を繰り返す．

(1) 読み書きヘッドが指すセルのデータ（入力記号を含む）を読み出す．

(2) 現在の状態と読み出したデータに従って，次の状態に遷移（演算）する．

(3) セルに指定されたデータを書き込み，読み書きヘッドを指定されたとおりに動かす．



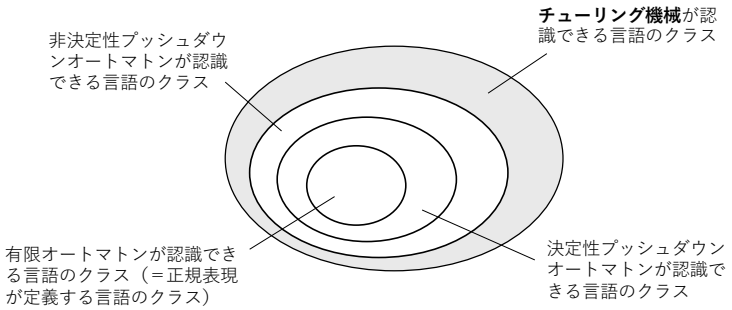
追加：コンピュータとチューリング機械

コンピュータ	チューリング機械
1. メモリーからCPUにデータの読み出し	1. ヘッドが示すセルからデータを読み出し
2. データをもとにCPUが演算を実行	2. データをもとに状態遷移を実行
3.メモリーに演算結果の書き込み	3. データを書き込みヘッドを移動

チューリング機械の能力

■チューリング機械は，CPUとメモリを備えた現在のコンピュータの基本モデルであり．その計算能力は現在のコンピュータと等しい．

■チューリング機械は，有限オートマトンやプッシュダウンオートマトンが認識できない言語も認識することができる．



1. 計算と言語
2. コンピュータが実行する計算
3. コンピュータが受理する言語

形式文法

■考え方：すべての文は，ある一つの要素から始めて，それに言語を定める規則を繰り返し適用することによって生成

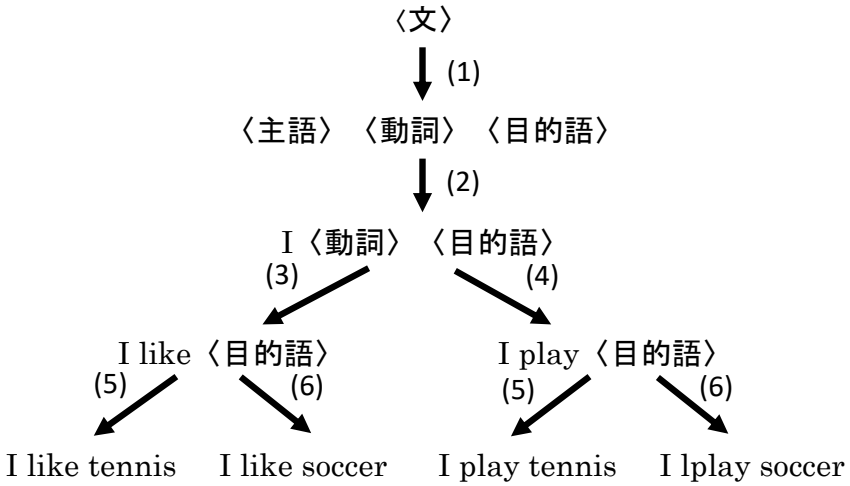
■形式文法の能力はオートマトンと同等

■形式文法はコンピュータで使われる数式やプログラミング言語を容易に定義

簡単な英語文

〈文〉→〈主語〉〈動詞〉〈目的語〉 (1)
〈主語〉 → I (2)
〈動詞〉 → like (3)
〈動詞〉 → play (4)
〈目的語〉 → tennis (5)
〈目的語〉 → soccer (6)

追加：導出例



形式文法の定義

$G = (N, T, P, S)$

N：非終端記号の集合
T：終端記号の集合
P：生成規則の集合
S：開始記号

簡単な英語文

G ={N, T, P, 〈文〉 }
　N ={ 〈文〉 , 〈主語〉 , 〈動詞〉 , 〈目的語〉 }
　T ={I, like, play, tennis, soccer }
　P ={ 〈文〉 → 〈主語〉 〈動詞〉 〈目的語〉 ,
　　〈主語〉 → I, 〈動詞〉 → like, 〈動詞〉 → play,
　　〈目的語〉 →tennis, 〈目的語〉 →soccer}

形式文法と言語

■文法Gにより記号列 α が記号列 β に書き換えられるとき，G は α から β を導出するといい，

$$\alpha \Rightarrow \beta$$

で表す.

■文法Gが導出する記号列 γ が

$$\gamma \in T^*$$

を満たすとき，γ をG で導出される文という.

■記号列の集合

$$L(G) = \{ \gamma \in T^* \mid S \Rightarrow \gamma \}$$

をG 生成する言語という.

有限オートマトンと形式文法

有限オートマトン

$M = (Q, \Sigma, \delta, q_0, F)$
Q：状態の集合
Σ：入力記号の集合
δ：状態遷移関数数
q_0：初期状態
F ：受理状態の集合

形式文法

$G = (N, T, P, S)$
N：非終端記号の集合
T：終端記号の集合
P：生成規則の集合
S：開始記号

要素の意味	有限オートマトン	形式文法
始まりを表す要素	初期状態	開始記号
言語を表す記号列	受理される記号列	導出される文
状態を表す要素	状態	生成途中の記号列
状態変化を表す要素	状態遷移関数	生成規則

代表的な形式文法

形式文法　$G=(N,T,P,S)$

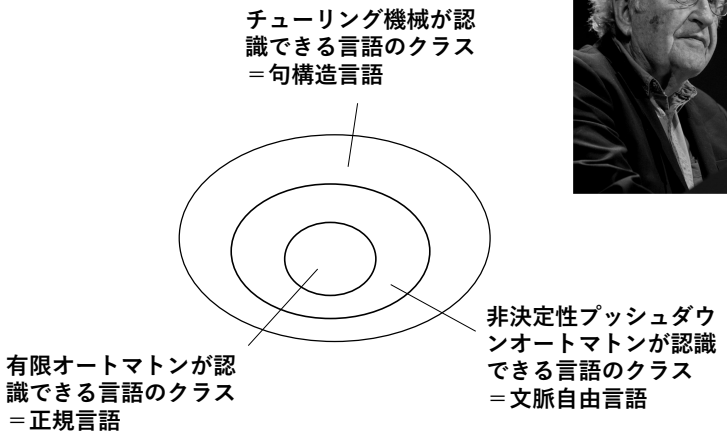
N：非終端記号の集合，　T：終端記号の集合

P：生成規則の集合，　S：開始記号

において，

文法	P
正規文法	$S \to \varepsilon, A \to a, A \to aB \quad (A,B \in N, a \in T)$
文脈自由文法	$A \to \alpha \quad (\alpha \in (N \cup T)^*)$
句構造文法	$\alpha \to \beta \quad (\alpha \in (N \cup T)^* N (N \cup T)^*, \beta \in (N \cup T)^*)$

チョムスキーの言語階層



参考書・参考資料

1) 山下 義行：「コンパイラ入門」サイエンス社，2008

2) 大川 知 ら：「オートマトン・言語理論入門」共立出版，2013.

3) 藤原 暁宏：「はじめて学ぶ オートマトンと言語理論」，森北出版，2015.

4) 岡留 剛：「例解図説 オートマトンと形式言語入門」森北出版，2015.

5) 東京大学理学部情報科学科授業紹介，
http://www.is.s.u-tokyo.ac.jp/vu/vu_lesson_entry.php?eid=00019&when=1