

アルゴリズムとデータ構造III 2回目:10月14日

文脈自由文法, CYK法

授業資料 <http://ir.cs.yamanashi.ac.jp/~ysuzuki/algorithm3/index.html>

授業の予定(中間試験まで)

1	10/07	スタック(後置記法で書かれた式の計算)
2	10/14	チューリング機械, 文脈自由文法
3	10/21	構文解析 CYK法
4	10/28	構文解析 CYK法
5	11/04	構文解析(チャート法), グラフ(ダイクストラ法)
6	11/11	構文解析(チャート法), グラフ(ダイクストラ法, DPマッチング)
7	11/18	グラフ(DPマッチング, A*アルゴリズム)
8	11/25	グラフ(A*アルゴリズム), 前半のまとめ
9	12/02	中間試験

10/28の代わりに補講日は後日相談

授業の予定(中間試験以降)

10	12/09	全文検索アルゴリズム(simple search, KMP)
11	12/16	全文検索アルゴリズム(BM, Aho-Corasick)
12	01/06	全文検索アルゴリズム(Aho-Corasick), データ圧縮
13	01/13	暗号(黄金虫, 踊る人形) 符号化(モールス信号, Zipfの法則, ハフマン符号)テキスト圧縮
14	01/20	テキスト圧縮(zip), 音声圧縮(ADPCM, MP3, CELP), 画像圧縮(JPEG)
15	02/03	期末試験

今日のメニュー

- 文脈自由文法
- 構文木, 構文解析
- CYK法

7 2 3 + - を計算してみよう (アセンブリ言語でプログラミング)

数式(7 2 3 + -)をメモリ(データ領域)に書き込まれている

- データ領域から1文字読み込む
 - 数字(アスキーコード:30H~39H)なら
 - 数値に変換し, 数値をスタックにプッシュ
 - 演算子なら
 - 一旦スタックにプッシュし, ポップする
 - スタックからポップし, 数値をBレジスタに取り込む
 - スタックからポップし, 数値をAレジスタ(アキュムレータ)に取り込む
 - 演算子が+なら
 - A + B を計算し, Aレジスタに計算結果を格納
 - 演算子が-なら
 - A - B を計算し, Aレジスタに計算結果を格納
 - Aレジスタの内容をスタックにプッシュ
- データ領域すべてを読み終えるまで続ける.

簡単な計算の例 7 2 3 + -

```

:後置記法 7 2 3 + - の計算
ORG 8000H:
LD HL, DATA: 数式の先頭番地を指定
LOOP: LD A, (HL)
CP 00H
JP Z, OWARI: 数式を全部読み込んだら終わ
LD E, (HL)
LD D, 0H
LD A, (HL)
INC HL
CP 2BH
JP Z, LOOPA: +なら加算処理へ
CP 2DH
JP Z, LOOPB: -なら減算処理へ
LD A, E
SUB 30H: 数字なら数値に変換
: Aレジスタの内容をスタックへプッシュ
STPUSH: LD E, A
LD D, 0H
PUSH DE: 読み込んだ数値をスタックへプ
JP LOOP: つぎの文字読み込みへ

:加算
LOOPA: PUSH DE: 演算子をスタックへプッシュ
POP DE: 演算子をスタックからポップ
POP DE: 数値をスタックからポップ
LD B, E: スタックトップの値をBレジスタへ
POP DE: 数値をスタックからポップ
LD A, E: スタックトップの値をAレジスタへ
ADD A, B: 加算(A <= A + B)
JP STPUSH

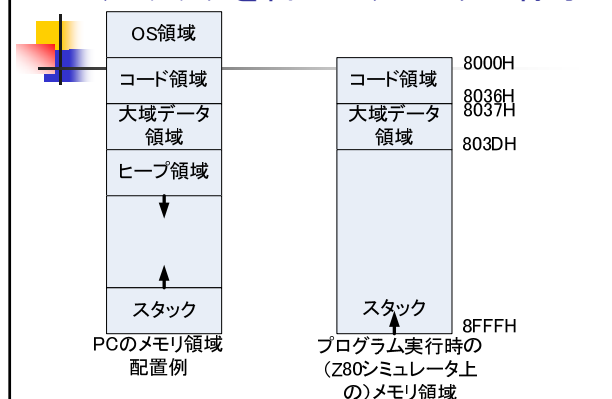
:減算
LOOPB: PUSH DE: 演算子をスタックへプッシュ
POP DE: 演算子をスタックからポップ
POP DE: 数値をスタックからポップ
LD B, E: スタックトップの値をBレジスタへ
POP DE: 数値をスタックからポップ
LD A, E: スタックトップの値をAレジスタへ
SUB B: 減算(A <= A - B)
JP STPUSH

:入力データ
OWARI: HALT
DATA: DEFB 37H:7
DEFB 32H:2
DEFB 33H:3
DEFB 2BH:+
DEFB 2DH:-
DEFB 00H:END
END
    
```

Z80 シミュレータ

- Z80シミュレータ for Win32
 - <http://www.game3rd.com/soft/z80edit/index.htm>

(スタックを含めた)メモリの様子



4.5.3 オートマトンと計算理論

オートマトンの受理する言語クラス

オートマトン	受理言語型	言語クラス
チューリング機械	第0型言語	句構造言語(PSL)
線形拘束チューリング機械	第1型言語	文脈依存言語(CSL)
プッシュダウンオートマトン	第2型言語	文脈自由言語(CFL)
有限オートマトン	第3型言語	正規言語(RL)

RL ⊂ CFL ⊂ CSL ⊂ PSL (チョムスキーの言語階層)
(⊂は包含関係を表す)

9

「形式言語と有限オートマトン入門」 5 形式言語理論入門

- 5.1 形式言語理論
- 5.2 文脈自由文法
- 5.3 線形文法と正規言語
- 5.4 形式言語のクラス階層とオートマトン
- 5.5 言語処理への応用

10

形式文法G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: プロダクション
 - S: 開始記号

11

5.2 文脈自由文法

- 文脈自由文法(CFG: Context Free Grammar)
 - 文脈自由プロダクションのみから構成される
 - 文脈自由プロダクション
 - $\alpha \rightarrow \beta$
 - ただし, $\alpha \in N, \beta \in V^*$
 - N: 非終端記号の集合, T: 終端記号の集合, V: NとTの直和
 - 左辺が変数1つ
- 文脈依存文法(CSG: Context Sensitive Grammar)
 - 文脈依存プロダクションを含むプロダクションから構成される
 - 文脈依存プロダクション
 - $u\alpha v \rightarrow u\beta v$ ただし, $\alpha \in N, u, v \in V^*, \beta \in V^+$
 - N: 非終端記号の集合, T: 終端記号の集合, V: NとTの直和
 - $u=v=\epsilon$ のとき($\alpha \rightarrow \beta$)文脈自由プロダクションとなる

12

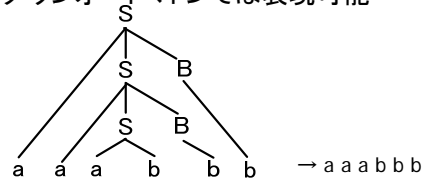
文脈自由文法の例(例題5.9)

- CFG $G=(N,T,P,S)$
 - N (非終端記号) = $\{B,S\}$
 - T (終端記号) = $\{a,b\}$
 - P (生成規則):
 - $S \rightarrow aSB \mid ab$
 - $B \rightarrow b$
 - S (始端記号): S
- 語 $aaabbb$ の導出過程

13

例題5.9の解答例

- CFG $G=(N,T,P,S)$
 - $N=\{B,S\}$
 - $T=\{a,b\}$
 - $P: S \rightarrow aSB \mid ab, B \rightarrow b$
 - $S: S$
- $S \Rightarrow aSB \Rightarrow aaSBB \Rightarrow aaabBB \Rightarrow aaabbbB \Rightarrow aaabbbb$
- $L(G): a^n b^n$
- 正規表現では表せない
- プッシュダウンオートマトンでは表現可能
- 構文木



14

例題5.10 文脈依存文法の例

- CSG $G=(N,T,P,S)$
 - $N=\{A,B,S\}$
 - $T=\{a,b\}$
 - $P: S \rightarrow aSBA \mid abA, AB \rightarrow BA, bB \rightarrow bb, bA \rightarrow ba, aA \rightarrow aa$
 - $S: S$
- 語 $aabbaa$ の導出過程
- $L(G)$ はどのような言語か

15

ここまでオートマトンと言語

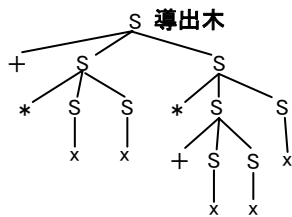
例題5.10 aabbaa 解答

- CSG $G=(N,T,P,S)$
 - $N=\{A,B,S\}$
 - $T=\{a,b\}$
 - $P: S \rightarrow aSBA \mid abA, AB \rightarrow BA, bB \rightarrow bb, bA \rightarrow ba, aA \rightarrow aa$
 - $S: S$
- 語 $aabbaa$ の導出過程
- $S \Rightarrow aSBA \Rightarrow aabABA \Rightarrow aabBAA \Rightarrow aabbbAA$
- $\Rightarrow aabbaA \Rightarrow aabbaa$
- $L(G)$ はどのような言語か
- $L(G): a^n b^n a^n$

16

例題5.11

- 問題:
- 文法 $N=\{S\}, T=\{x, +, *\}, P=\{S \rightarrow +SS \mid *SS \mid x\}, S=S$
- 語 $w = +*xx*+xxx$ を導出せよ
- 語 w の導出木

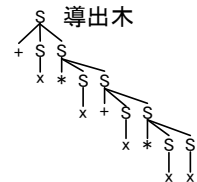


- 解答例
- 導出:
 $S \Rightarrow +SS \Rightarrow +*SSS \Rightarrow +*xSS \Rightarrow +*xxS \Rightarrow +*xx*SS \Rightarrow +*xx*+SS \Rightarrow +*xx*+*SSS \Rightarrow +*xx*+*xxx$

17

例題5.12 ①

- 問題
- 文法 $N=\{S\}, T=\{x, +, *\}, P=\{S \rightarrow +SS \mid *SS \mid x\}$
- 中置記法 $x+x*(x+x*x)$



- 解答例
- 前置記法 $+x*x+x*x*x$
- $S \Rightarrow +SS \Rightarrow +xS \Rightarrow +x*SS \Rightarrow +x*xS \Rightarrow +x*x*xS \Rightarrow +x*x*x*xS \Rightarrow +x*x*x*x*xS \Rightarrow +x*x*x*x*x*xS \Rightarrow +x*x*x*x*x*x*xS$

18

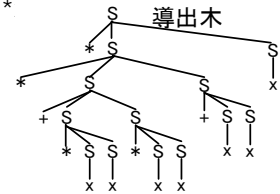
練習問題1 例題5.12 ②

- 問題
- 文法 $N=\{S\}, T=\{x, +, *\}, P=\{S \rightarrow +SS | *SS | x\}$
- 中置記法 $(x*x+x*x)^*(x+x)^*x$
- 前置記法
- 最左導出
- 構文木

19

練習問題1 例題5.12 ②の解答例

- 問題
- 文法 $N=\{S\}, T=\{x, +, *\}, P=\{S \rightarrow +SS | *SS | x\}$
- 中置記法 $(x*x+x*x)^*(x+x)^*x$



- 解答例
- 前置記法 $**+*xx*xx+xxx$
- $S \Rightarrow *SS \Rightarrow **SSS \Rightarrow **+SSSS \Rightarrow **+*SSSSS$
- $\Rightarrow **+*xSSSS \Rightarrow **+*xxSSS \Rightarrow **+*xx*SSSS$
- $\Rightarrow **+*xx*xSSS \Rightarrow **+*xx*xxSS \Rightarrow **+*xx*xx+SSS$
- $\Rightarrow **+*xx*xx+xxx$

20

文脈自由文法の曖昧性

- どのような導出を行っても同じ導出木が得られる
- \Rightarrow 文法Gは曖昧でない
- 複数の異なった導出木が構成できるような語を含む
- \Rightarrow 文法Gは曖昧である

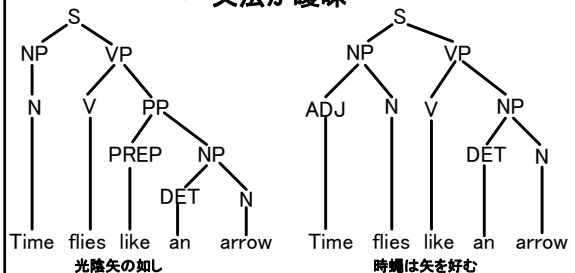
21

構文木(導出木)

- Time flies like an arrow.

$S \rightarrow NP VP$
 $NP \rightarrow N | DET N | ADJ N$
 $VP \rightarrow V | PP | V NP$
 $PP \rightarrow PREP NP$
 $N \rightarrow Time | arrow | flies$
 $V \rightarrow flies | like$
 $PREP \rightarrow like$
 $DET \rightarrow an$
 $ADJ \rightarrow Time$

2種類の導出木
 \rightarrow 文法が曖昧



22

例題5.26

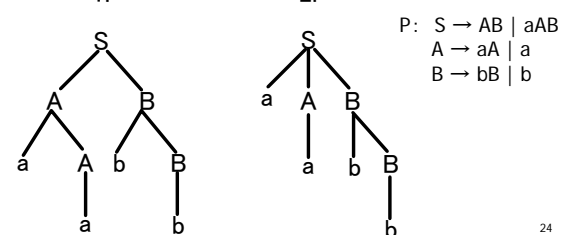
- 文法 $G=(N, T, P, S)$ において,
 $N=\{S, A, B\}, T=\{a, b\},$
- $P: S \rightarrow AB | aAB, A \rightarrow aA | a, B \rightarrow bB | b$
- この文法が曖昧であることを示せ

23

例題5.26 解答例

同一文字列に対して2種類の導出木が構成可能 \rightarrow 曖昧である

- 1. $S \rightarrow AB \rightarrow aAB \rightarrow aAbB \rightarrow aabB \rightarrow aabb$
- 2. $S \rightarrow aAB \rightarrow aaB \rightarrow aabB \rightarrow aabb$



24

練習問題2

例題5.27

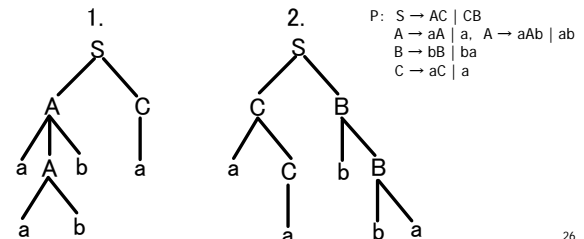
- 文法 $G=(N,T,P,S)$ において,
 - $N=\{S,A,B,C\}, T=\{a,b\}$,
 - $P: S \rightarrow AC|CB, A \rightarrow aA|a, A \rightarrow aAb|ab, B \rightarrow bB|ba$
 - $C \rightarrow aC|a$
- この文法が曖昧であることを, $aabba$ の導出木を構成して示せ

25

練習問題2 例題5.27 解答例

同一文字列に対して2種類の導出木が構成可能 → 曖昧である

1. $S \rightarrow AC \rightarrow aAbC \rightarrow aAba \rightarrow aabba$
2. $S \rightarrow CB \rightarrow aCB \rightarrow aCbB \rightarrow aabB \rightarrow aabba$



26

CFGの構文図式

文脈自由プロダクション

$A \rightarrow \alpha$

$A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$

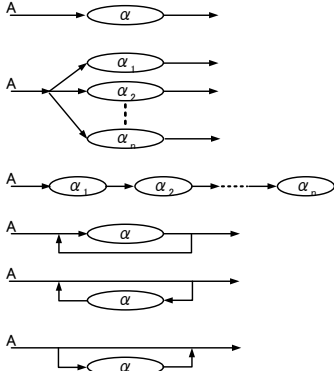
$A \rightarrow \alpha_1 \alpha_2 \dots \alpha_n$

$A \rightarrow \alpha | \alpha A$

$A \rightarrow \epsilon | \alpha | \alpha A$

$A \rightarrow \epsilon | \alpha$

構文図式



27

構文解析

- 文脈自由文法で生成された文から自動的に構文木を生成する。

構文解析とは(Wikipediaより)

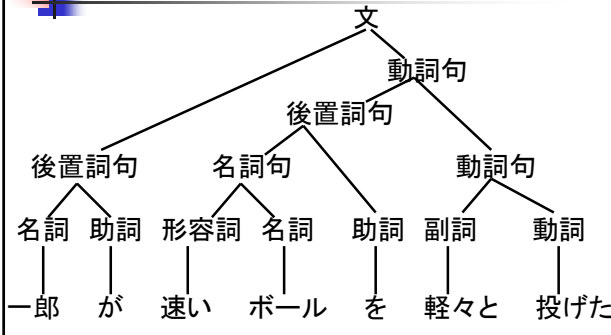
- ある文章の文法的な関係を説明すること(*parse*)。計算機科学の世界では、構文解析は字句解析 (*Lexical Analysis*) とともに、おもにプログラミング言語などの形式言語の解析に使用される。また、自然言語処理に応用されることもある。
- コンパイラにおいて構文解析を行う機構を構文解析器 (Parser) と呼ぶ。
- 構文解析は入力テキストを通常、木構造のデータ構造に変換し、その後の処理に適した形にする。字句解析によって入力文字列から字句を取り出し、それらを構文解析器の入力として、構文木や抽象構文木のようなデータ構造を生成する。

構文解析

- 入力文(記号列)が与えられたとき、文法によってその文を解析し、その構造を明らかにする
- 代表的な構文解析アルゴリズム
 - CYK法
 - チャート法
 - アーリー法
 - LR法

構文木

(一郎が速いボールを軽々と投げた)



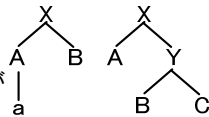
CYK (Cocke-Younger-Kasami) 法

- ボトムアップアルゴリズム
- 扱える文法
 - チョムスキーの標準形
 - $A \rightarrow BC$
 - $A \rightarrow a$
 - $A \rightarrow BCD$, $A \rightarrow aB$ などは直接は扱えない
- CYK表
 - 構文解析の途中経過を保持するための表

CYKアルゴリズム

- チョムスキーの標準形で表される文脈自由文法を対象とした構文解析法
- チョムスキーの標準形
 - $A \rightarrow BC$ ($A, B, C \in V_n$)
 - $A \rightarrow a$ ($A \in V_n, a \in V_t$)

$X \rightarrow aB$ はチョムスキーの標準形ではないが
 $X \rightarrow AB, A \rightarrow a$ に分解できる
 $X \rightarrow ABC$ はチョムスキーの標準形ではないが
 $X \rightarrow AY, Y \rightarrow BC$ に分解できる



チョムスキーの標準形の例 「急いで走る一郎を見る」

A→BC型

A→a型

- (1) $s \rightarrow pp v$
- (2) $s \rightarrow adv vp$
- (3) $vp \rightarrow pp v$
- (4) $vp \rightarrow adv v$
- (5) $np \rightarrow vp n$
- (6) $np \rightarrow v n$
- (7) $pp \rightarrow np p$
- (8) $pp \rightarrow n p$
- (9) $adv \rightarrow 急いで$
- (10) $n \rightarrow 一郎$
- (11) $p \rightarrow を$
- (12) $v \rightarrow 走る$
- (13) $v \rightarrow 見る$

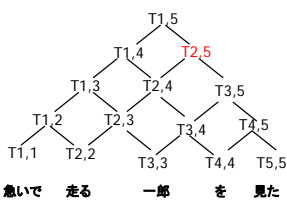
CYK構文解析の概要

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで				
2. 走る		T2,2	T2,3	T2,4
3. 一郎				T3,5
4. を				T4,5
5. 見た				T5,5

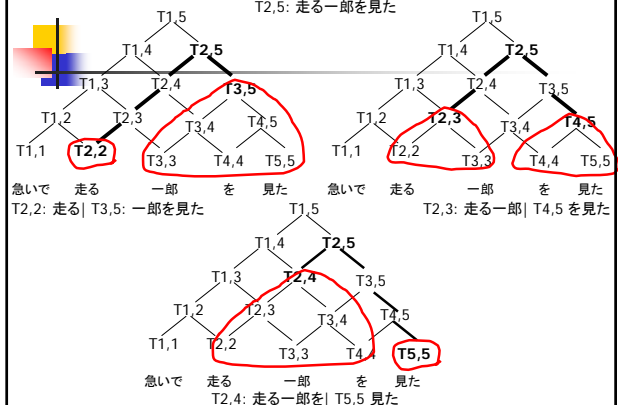
T2,5: 走る一郎を見た

T2,2: 走る | T3,5: 一郎を見た
 T2,3: 走る一郎 | T4,5: を見た
 T2,4: 走る一郎を | T5,5: 見た



CYK表は構文木を表している

T2,5までの部分木 (3種類)



CYKアルゴリズム

1. $A \rightarrow a$ の生成規則を用いて、主対角線上の要素を計算
for $i=1$ to N
 $T_{i,i} = \{A \mid A \rightarrow w_i\}$
2. $A \rightarrow BC$ の生成規則を用いて、2番目以降の対角線上の要素を計算
for $n=1$ to $N-1$
for $i=1$ to $N-n$
$$T_{i,i+n} = \bigcup_{j=1}^n \{A \mid A \rightarrow BC, B \in T_{i,i+j-1}, C \in T_{i+j,i+n}\}$$
3. $S \in T_{1,N}$ であれば、 $w_1 \dots w_N$ は開始記号 S から導出可能

CYK構文解析表(完成)

	1. 急いで	2. 走る	3. 一郎	4. を	5. 見た
1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	vp→pp v s→pp v s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎			n→一郎	pp→n p	vp→pp v s→pp v
4. を				p→を	
5. 見た					v→見た

- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p
- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

今日のまとめ

- 文脈自由文法
- 構文木, 構文解析
- CYK法