アルゴリズムとデータ構造III
2回目：10月15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	10/01	スタック（後置記法で書かれた式の計算）
2	10/15	文脈自由文法，構文解析，CYK法
3	10/22	構文解析　CYK法
4	10/29	構文解析　CYK法
5	11/12	構文解析（チャート法），グラフ（ダイクストラ法）
6	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/	グラフ（DPマッチング，A*アルゴリズム）
8	11/19	グラフ（A＊アルゴリズム），前半のまとめ
9	11/26	中間試験

10/08, 11/05の代わりの補講日は後日相談

授業の予定（中間試験以降）

10	12/03	全文検索アルゴリズム（simple search, KMP)
11	12/10	全文検索アルゴリズム（BM, Aho-Corasick)
12	12/17	全文検索アルゴリズム（Aho-Corasick)，データ圧縮
13	01/07	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	01/14	テキスト圧縮　（zip)， 音声圧縮　（ADPCM，MP3，CELP)， 画像圧縮（JPEG）
15	01/21	**期末試験**

７２３＋ ー 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(７２３＋ ー)をメモリ（データ領域）に書き込まれている

1. データ領域から1文字読み込む
    1. 数字（アスキーコード：30H～39H）なら
        1. 数値に変換し，数値をスタックにプッシュ
    2. 演算子なら
        1. 一旦スタックにプッシュし，ポップする．
        2. スタックからポップし，数値をBレジスタに取り込む
        3. 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
        4. 演算子が＋なら
            1. A + B を計算し，Aレジスタに計算結果を格納
        5. 演算子がーなら
            1. A －B を計算し，Aレジスタに計算結果を格納
        6. Aレジスタの内容をスタックにプッシュ
2. データ領域すべてを読み終えるまで続ける．

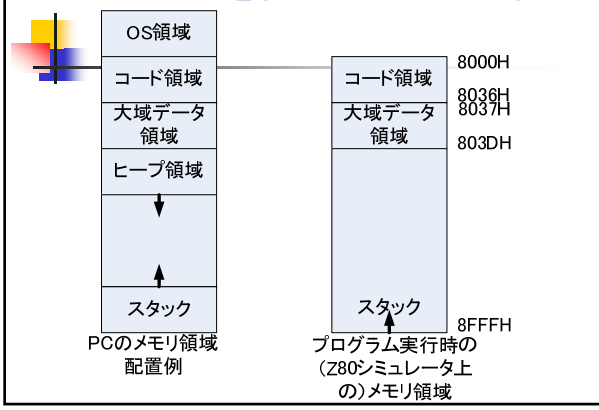
簡単な計算の例　７２３＋ ー

```
; 後置記法　７２３＋ - 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り
            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, LOOPS ; ー 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ッシュ
            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
LOOPS:      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
;
OWARI: HALT
;入力データ
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）

8

「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

9

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

10

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
    - $u=v=\varepsilon$ のとき（$\alpha \rightarrow \beta$）文脈自由プロダクションとなる

11

文脈自由文法の例(例題5.9)

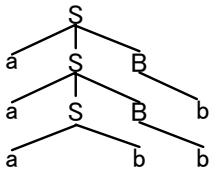
- CFG G=(N,T,P,S)
  - N（非終端記号）={B,S}
  - T（終端記号）={a,b}
  - P:
    - S→aSB | ab
    - B→b
  - S: S
- 語　aaabbb の導出過程
- L(G)はどのような言語か

12

例題5.9の解答例

- CFG G=(N,T,P,S)
  - N={B,S}
  - T={a,b}
  - P: S→aSB | ab, B→b
  - S: S

- S⇒aSB⇒aaSBB⇒aaabBB⇒aaabbB⇒aaabbb
- 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



13

練習問題1
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→aSBA | abA, AB→BA, bB→bb, bA→ba, aA→aa
- S: S

- 語　aabbaa　の導出過程
- L(G)　はどのような言語か

14

練習問題1　解答
例題5.10 aabbaa

- CSG G=(N,T,P,S)
- N={A,B,S}
- T={a,b}
- P: S→aSBA | abA, AB→BA, bB→bb, bA→ba, aA→aa
- S: S

- 語　aabbaa　の導出過程
- S⇒aSBA⇒aabABA⇒aabBAA⇒aabbAA
- ⇒aabbaA⇒aabbaa
- L(G)　はどのような言語か
- L(G): $a^n b^n a^n$

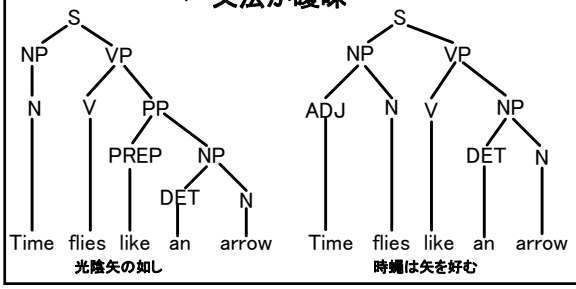
--

構文木（導出木）

- Time flies like an arrow.

S→NP VP
NP→N|DET N|ADJ N
VP→V PP|V NP
PP→PREP NP
N→Time|arrow | flies
V→flies | like
PREP→like
DET→an
ADJ →Time

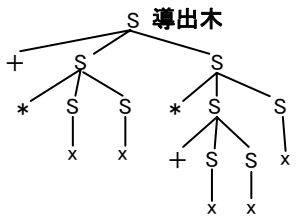
2種類の導出木
→　文法が曖昧



16

例題5.11

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

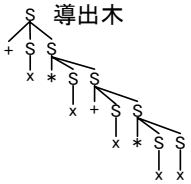


- 解答例
- 導出:
  S⇒+SS⇒+*SSS⇒+*xSS⇒+*xxS⇒+*xx*SS⇒+*xx*+SSS⇒+*xx*+xxx

17

例題5.12 ①

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



- 解答例
- 前置記法　+x*x+x*xx
- S⇒+SS⇒+xS⇒+x*SS⇒+x*xS⇒+x*x+SS
  ⇒+x*x+xS⇒+x*x+x*SS⇒+x*x+x*xS ⇒+x*x+x*xx

18

練習問題2　例題5.12 ②

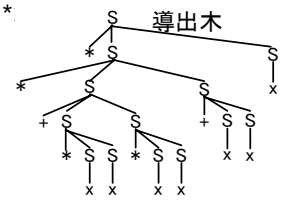
- 問題
- 文法　N={S},T={x,+,*},P={S→+SS|*SS|x}
- 中置記法　(x*x+x*x)*(x+x)*x
- 前置記法
- 最左導出
- 構文木

19

練習問題2　例題5.12 ②の解答例

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

導出木



- 解答例
- 前置記法　**+*xx*xx+xxx
- S⇒*SS⇒**SSS ⇒**+SSSS ⇒**+*SSSSS ⇒**+*xSSSS ⇒**+*xxSSS ⇒**+*xx*SSSS ⇒**+*xx*xSSS ⇒**+*xx*xxSS ⇒**+*xx*xx+SSS
- ⇒**+*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
- ⇒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
- ⇒文法Gは曖昧である

21

例題5.26

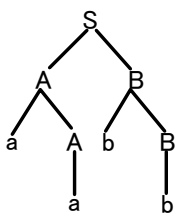
- 文法G=(N,T,P,S)において，N={S,A,B},T={a,b},
- P：S→AB|aAB,　A→aA|a,　B→bB|b
- この文法が曖昧であることを示せ

22

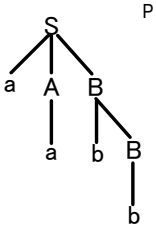
例題5.26 解答例 同一文字列に対して2種類の導出木が構成可能→曖昧である

- 1. S→AB→aAB→aAbB→aabB→aabb
- 2. S→aAB→aaB→aabB→aabb

1.



2.



P：S → AB | aAB
A → aA | a
B → bB | b

23

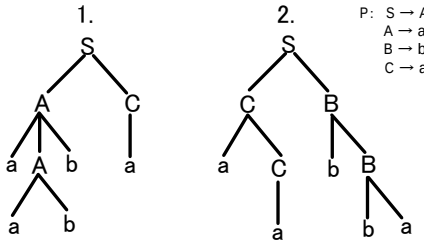
練習問題3 例題5.27

- 文法G=(N,T,P,S)において，
- N={S,A,B,C},T={a,b},
- P：S→AC|CB,　A→aA|a,　A→aAb|ab,　B→bB|ba
- C→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

24

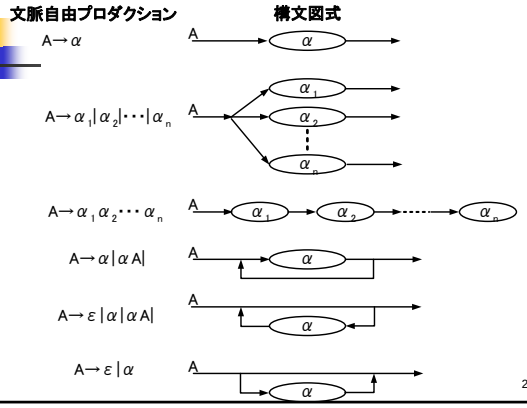
練習問題3　例題5.27　解答例
同一文字列に対して2種類の導出木が構成可能 → 曖昧である

- 1. S→AC→aAbC→aAba→aabba
- 2. S→CB→aCB→aCbB→aabB→aabba



25

CFGの構文図式



26

ここまで

構文解析　CYK法

- 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.

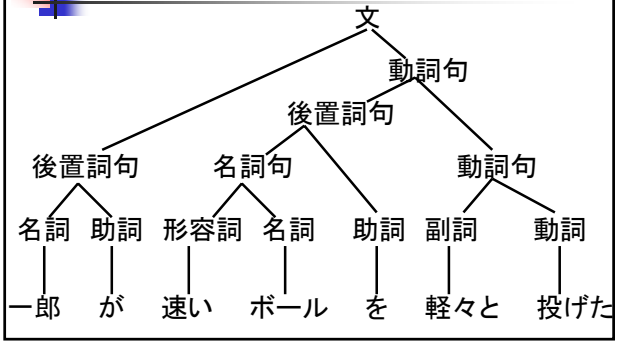
構文解析とは(Wikipediaより)

- 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→BC
    - A→a
- 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→BC （A,B,C∈Vn)
  - A→a (A∈Vn, a∈Vt）

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