

第6回 構文解析の基本

広島大学工学部第二類
藤田 聡

今日の授業内容

- **構文解析が語の生成の逆過程であることを理解する**
- **「導出木」を常に頭の中においておく**
 - **「構文木」という別の木構造があとで出てくる**
- **上昇型解析と下降型解析の考え方を理解する**
- **再帰下降解析について理解する**

構文解析の目的

(syntax analysis, parsing)

- ✓ 与えられた記号列(具体的には字句解析のおわったソースプログラム)の文法構造を調べること
- ✓ 終端記号列から導出木をつくる(生成過程の逆)

対象とする構文解析法

- 条件1: 文字列は左から右へ順に一度だけみる
- 条件2: 決定的に動作し、バックトラックはしない
- 条件3: 右側の k (≥ 0) 個の記号の先読み (look-ahead) を許す (多くの場合 $k = 1$)



構文解析法の分類(1)

下降型解析 (top-down parsing)

- 例) 再帰下降型解析, LL解析など

上昇型解析 (bottom-up parsing)

- 例) 演算子順位解析 (operator precedence parsing), LR解析など

構文解析法の分類(2)

下降型解析 (top-down parsing)

- ・ 「**最左導出**」を「**順に**」おこなう
- ・ **導出木の根と葉の左端を結ぶ線から順に解析木を確定**

例) 再帰下降型解析(今週), LL解析(来週)

構文解析法の分類(3)

上昇型解析 (bottom-up parsing)

- ・ 「最右導出」を「逆順に」おこなう
- ・ 葉の左端から順に解析木を確定

例) 演算子順位解析 (operator precedence parsing),
LR解析(再来週)

例

識別子リスト ::= 識別子 | 識別子, 識別子リスト
識別子 ::= a | b | c

⇒ 入力記号列 a,b,b の解析

下降型解析 (最左導出)

識別子リスト

- 識別子, 識別子リスト
- a, 識別子リスト (aを確定)
- a, 識別子, 識別子リスト
- a, b, 識別子リスト (bを確定)
- a, b, 識別子
- a, b, b (bを確定)

最右導出

識別子リスト

- 識別子, 識別子リスト
- 識別子, 識別子, 識別子リスト
- 識別子, 識別子, 識別子
- 識別子, 識別子, b
- 識別子, b, b
- a, b, b

上昇型解析 (最右導出の逆順)

a, b, b

- 識別子, b, b (aを確定)
- 識別子, 識別子, b (bを確定)
- 識別子, 識別子, 識別子 (bを確定)
- 識別子, 識別子, 識別子リスト
- 識別子, 識別子リスト
- 識別子リスト

再帰下降解析(1)

(recursive descent parsing)

Pascal,Cなど多くの処理系で採用されている
「実用的」な構文解析法

○ アイデア

- ・ 各構文要素(非終端記号)ごとにひとつの解析ルーチンを用意
- ・ 解析ルーチンは、構文規則の再帰的構造に対応してお互いに再帰的に呼びだし合う。

再帰下降解析(2)

- ・ 各ルーチンの内容は、その構文要素を左辺とする構文規則の右辺に対応
- ・ 構文規則の右辺にあらわれる非終端記号はそれに対応する解析ルーチンの呼びだしに対応
- ・ 終端記号は入力記号列におけるその記号の読み込みに対応

例(1)

while 文 ::= while (式) 文 を解析するルーチン

```
if (sy == WHILE) {  
    insymbol();  
    if (sy == LPAR) insymbol(); else error();  
    expression();  
    if (sy == RPAR) insymbol(); else error();  
    statement();  
}
```

例(2)

while 文 ::= while (式) 文 を解析するルーチン

読んだ字句がwhileのとき、 {
シンボルをよむ → シンボルが `(` でなければエラー;
シンボルをよぶ → expression (); /* 式の評価 */
シンボルをよむ → シンボルが `)` でなければエラー;
シンボルをよぶ → statement (); /* 文の評価 */
}

再帰下降解析がうまくいかない場合(1)

左再帰 (left recursion)

$A ::= Aa|b$

⇒ 無限ループ

$A ::= bA' \quad A' ::= \varepsilon | aA'$

のように生成規則を修正 (空列 ε をいれる)

再帰下降解析がうまくいかない場合(2)

展開に複数の可能性がある場合

$$A ::= s \mid t$$

⇒ バックトラックが必要?

左くりだし (left factoring)

⇒ s と t の共通のprefix u をつかって

$$(s = uv, t = uw)$$

$A ::= uA' \quad A' ::= v \mid w$ のように修正

左くりだしの例

たとえば

if 文 ::= if 式 then 文 | if 式 then 文 else 文 は
if 文 ::= if 式 then 文 E, E ::= ε | else 文

のように修正すればよい

ここまでのまとめ

- **構文解析が語の生成の逆過程であることを理解する**
- **「解析木」を常に頭の中においておく**
- **上昇型解析と下降型解析の考え方を理解する**
- **再帰下降解析について理解する**