第3章 下向き構文解析 (教科書 p.50)

3.1 下向き構文解析の特徴

演算子順位法は if ～ else などの制御構造の部分には使いにくい
LR 法は　　人手での作成に向かない

アイデア

左端のトークン（if, whileなど）を見て生成規則を選ぶ（ 予測型構文解析 ）

例 教科書 p.14 図 1.4 の Statement（改）

$$
\begin{aligned}
Statement \rightarrow\ & \texttt{if (}\ ConditionExp\ \texttt{)}\ Statement\ ElsePart \\
\mid\ & \texttt{\{}\ StatementSeq\ \texttt{\}} \\
\mid\ & Id\ \texttt{=}\ Expression\ \texttt{;} \\
\mid\ & Id\ \texttt{(}\ ExpressionList\ \texttt{);} \\
\mid\ & \cdots
\end{aligned}
$$

各非終端記号について次の疑似コードで示すような（再帰的な）関数を定義する

```
int Statement(void) {
    switch (次のトークン) {
    case IF: {
        IF を消費;
        '('  〃 ;
        c = ConditionExp();
        ')'  〃 ;
        s = Statement();
        e = ElsePart();
        return c, s, e を使った式
    }
    case '{': {
      '{' を消費;
      s = StatementSeq();
      '}' を消費;
      return s を使った式;
    }

    /* ⋮ */

    }
}
```

3.2 再帰下降構文解析 (recursive decent parsing)

- 下向き構文解析 の一種 … 幹から葉へ解析木ができていく
- 左再帰 があるとまずい

$$
\begin{aligned}
expr \rightarrow\ & const \\
\mid\ & \underline{expr}\ \texttt{*}\ const
\end{aligned}
$$

$$StatementSeq \to \varepsilon \\ \quad | \quad \underline{StatementSeq}\ Statement$$

下線部のところが、一番左端の再帰的出現である。これをプログラムにしようとすると、

```
int StatementSeq(void) {
    switch (次のトークン) {
      case IF: case WHILE: … {
        ss = StatementSeq();  /* ← */
        s1 = Statement();
        return …;
      }

       /* ⋮ */

    }
}
```

← のところで入力が変わらないので、止まらなくなる。

必要な準備

- 先頭の共通部分をくくりだす

  $$S \to A\,B \quad | \quad A\,C$$

  は

  $$S \to A\,T \\ T \to B \mid C$$

  に書き換える

- 左再帰を除去 する

- BNF の各右辺 α に対して $First(\alpha)$ を求める
  （$First(\alpha)$ は α の 最初に現れうる終端記号の集合 ）

- $A \overset{*}{\Rightarrow} \varepsilon$となりうる非終端記号 A に対して $Follow(A)$ を求める
  （$Follow(A)$ は A の 直後に現れうる終端記号の集合 ）

再帰下降構文解析のプログラムの作り方

各非終端記号に対して関数を定義する

I. $N \to X_{11}X_{12}\ldots X_{1n_1} \quad | \quad \ldots \quad | \quad X_{m1}X_{m2}\ldots X_{mn_m}$に対して、次のトークンがどの $First(X_{i1}X_{i2}\ldots X_{in_i})$ に属するかによって分岐する（ $X_{i1}X_{i2}\ldots X_{in_i} \overset{*}{\Rightarrow} \varepsilon$ となる場合は $Follow(N)$ も考慮する）

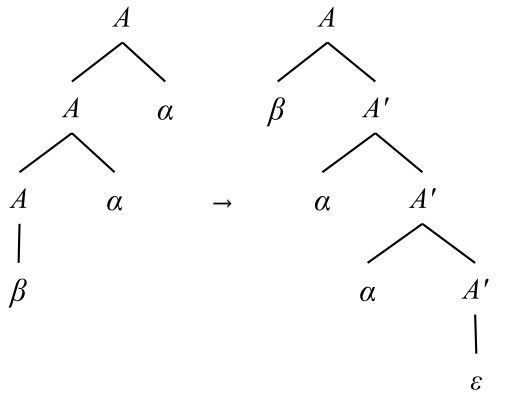
II. 右辺の $X_{i1}X_{i2}\ldots X_{in_i}$ に対して `Xi1(); Xi2 (); … Xin_i();` のように続けて関数を呼出す（ただし、X_{ij} が終端記号のときは単にトークンを消費する）
そのあと 末尾再帰 を繰り返しに書き換える（効率のため）

3.3 アルゴリズム（左再帰の除去） (教 p.57)

$$A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n$$

とする（α_i, β_j は構文記号列で β_j の先頭の記号は A ではない。 β_j の先頭以外には A は出現しても構わない）
↓
先頭が $\beta_1 \sim \beta_n$ でそのあとに $\alpha_1 \sim \alpha_m$ が 0 回以上繰り返すというかたちになる
↓
次のように書き換えることができる

$$A \to \beta_1 A' \mid \ldots \mid \beta_n A'$$
$$A' \to \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \varepsilon$$



注:

- 最後の ε を忘れない
- 間接的な左再帰（教科書 p.54）があるともう少しや
やこしくなるが、そのよう
な場合でも除去可能である
ことが知られている

問 **3.3.1**

$$L \to L\,;\,C \mid C$$

の左再帰を除去せよ。

First と Follow の求め方の例 (詳しい説明は教科書 pp.60-61)

例 **5.3**

$$E \to T\,E'$$
$$E' \to +\,T\,E' \mid \varepsilon$$
$$T \to F\,T'$$
$$T' \to *FT' \mid \varepsilon$$
$$F \to (\,E\,) \mid \mathbf{id}$$

$First(TE') = First(T) = First(FT') = First(F)$
$\leftarrow First(F)$ に ε が入っていないので T' や E' は考慮しなく
$First(FT') = First(F) = \{(, \mathbf{id}\}$
$Follow(E') = Follow(E) \cup Follow(E')$
$Follow(E) = \{), \$\} \leftarrow$ 開始記号の $Follow$ には $ （入力の終）を追加する
$Follow(T') = Follow(T) \cup Follow(T')$
$Follow(T) = First(E') \setminus \{\varepsilon\} \cup Follow(E) \cup Follow(E')$
$= \{+,), \$\}$
$First(E') = \{+, \varepsilon\} \leftarrow$ ε になりうる場合、 ε を加える
（$First(T')$ と $Follow(F)$ は求める必要はない）

3.4 予測型構文解析表 (教科書 表5.1 （p.63）)

- $First$ と $Follow$ の結果をまとめて表にまとめたもの
- 構文解析すべき非終端記号 A と入力の先頭の終端記号 a に対して 選択すべき生成規則 を示す

LL(1) 文法

予測型構文解析表のエントリーに重複がない文法のことを LL(1) 文法 という

- エントリーに重複があると構文解析中に 後戻り が必要となる（通常のプログラミング言語では記述しにくい … Prolog の出番?）
- LL(1) は Left-to-Right Leftmost derivation (1) に由来する。 Leftmost（最左導出）はあとで説明する

表 **5.1** 教科書 **p.63**

	id	*	+	(	)	**$**
$E \rightarrow$	TE' ①			TE' ①		
$E' \rightarrow$			$+TE'$ ②		ε ③	ε ③
$T \rightarrow$	FT' ④			FT' ④		
$T' \rightarrow$		$*FT'$ ⑤	ε ⑥		ε ⑥	ε ⑥
$F \rightarrow$	$\mathbf{id}$ ⑦			(E) ⑧		

① $\because First(TE') = \{(, \mathbf{id}\}$
② $\because First(+TE') = \{+\}$
③ $\because Follow(E') = \{), \$\}$
④ $\because First(FT') = First(F) = \{(, \mathbf{id}\}$
⑤ $\because First(*FT') = \{*\}$
⑥ $\because Follow(T') = \{+,), \$\}$
⑦ $\because First(\mathbf{id}) = \{\mathbf{id}\}$
⑧ $\because First((E)) = \{(\}$

入力例 x + y * z ただし、x, y, z は **id** に属するトークンである。

予測	入力		動作