Chapter 2

讨论

0.1

> 请简要说明文法在编译中的作用

`文法` 在编译中的作用是 `定义` 编程语言的 `语法规则`, 从而实现了对 `编程语言` 中各种 `语法结构` 组成的 `具体描述`

0.2

> C 语言的宏是在编译的什么阶段处理的? 说明理由

C 语言的 `宏` 在 `预处理` 阶段 (位于 `词法分析` 阶段前, 专门用于 `宏的展开` 以及 `拷贝头文件中的代码`) 被处理的

首先可以明确的是, `宏` 被展开后生成的代码, 是会经过语法分析的, 因此它应该位于 `语法分析` 阶段前

```c
#define FOO (x + 1)

int main() {
  int y = FOO;
  return 0;
}
```

上述代码, 编译后会报错:

```
<source>: In function 'main':
<source>:1:14: error: 'x' undeclared (first use in this function)
    1 | #define FOO (x + 1)
```

```
      |               ^
<source>:4:11: note: in expansion of macro 'FOO'
    4 |   int y = FOO;
      |           ^~~
<source>:1:14: note: each undeclared identifier is reported only once for each function it appears in
    1 | #define FOO (x + 1)
      |              ^
<source>:4:11: note: in expansion of macro 'FOO'
    4 |   int y = FOO;
      |           ^~~
Compiler returned: 1
```

显然, 编译器认定, x 未定义, 报出 `语法错误`

其次, 通过一个简单的实验可以得知, `宏` 被展开后生成的代码, 也是要经过 `词法分析` 的, 实验如下:

```c
#define GREETING_IN_JP_VAR_NAME こんにちは,
// attention: `,` in the end

int main() {
  char *GREETING_IN_JP_VAR_NAME = "Hello!";
}
```

报出错误:

```
<source>: In function 'main':
<source>:4:33: error: expected identifier or '(' before '=' token
    4 |   char *GREETING_IN_JP_VAR_NAME = "Hello!";
      |                                 ^
Compiler returned: 1
```

`before '=' token` 中的 `token` 说明了一切问题: 在词法分析之前, 展开了宏定义. 假如说, 词法分析之前没有展开宏定义, 编译器就会把 `GREETING_IN_JP_VAR_NAME` 解析为一个合法的 `标识符`, 而不是 `こんにちは` 和 `,` 两个连续的 `token`

0.3

> 已知 一上下文无关文法 和 一该文法的句型, 该句型 是否存在一 最左推导或最右推导?

事实上, 不一定存在 `最左/右推导`, 我们只能确定, 任意文法的任意句型, 一定存在 `推导`, 因为 `任意句型` 本身就是 `某个文法` 按照 `某种推导方式` 得出的结果

作业

课本作业

P36 T7

> 写一个文法, 使其语言为奇数集, 且每个奇数不得以 0 开头

一个可行的方案是:

$$S \to O|OS|ES|OZS|EZS$$
$$O \to 1|3|5|7|9$$
$$E \to 2|4|6|8$$
$$Z \to 0$$

注意对 `开头` 的 `特殊处理`

P36 T8

> 题目略, 见课本

$$E \to T|E+T|E-T$$
$$T \to F|T*F|T/F$$
$$F \to (E)|i$$

是一个经典的 `算术表达式文法`, 描述了所有合法的 `四则运算表达式`

首先, 分别给出 `i + i * i` 和 `i * (i + i)` 的 `最左/右推导`

`i + i * i`:

$$\begin{aligned} E &\Rightarrow E+T \\ &\Rightarrow T+T \\ &\Rightarrow F+T \\ &\Rightarrow i+T \\ &\Rightarrow i+T*F \\ &\Rightarrow i+F*F \\ &\Rightarrow i+i*F \\ &\Rightarrow i+i*i \ \ldots \text{最左推导} \end{aligned}$$

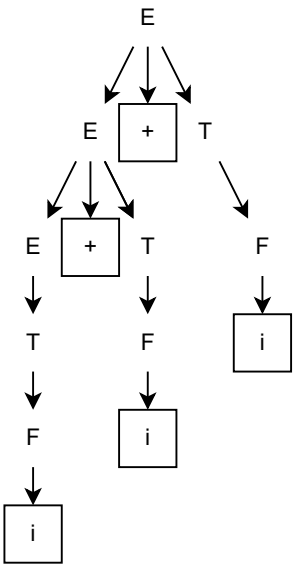
$$\begin{aligned} E &\Rightarrow E+T \\ &\Rightarrow E+T*F \\ &\Rightarrow E+T*i \\ &\Rightarrow E+F*i \\ &\Rightarrow E+i*i \\ &\Rightarrow T+i*i \\ &\Rightarrow F+i*i \\ &\Rightarrow i+i*i \ \ldots \text{最右推导} \end{aligned}$$

`i * (i + i)`:

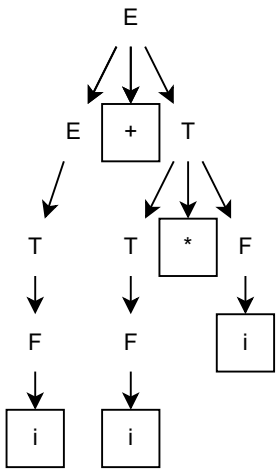
$$
\begin{aligned}
E &\Rightarrow T \\
&\Rightarrow T * F \\
&\Rightarrow F * F \\
&\Rightarrow i * F \\
&\Rightarrow i * (E) \\
&\Rightarrow i * (E + T) \\
&\Rightarrow i * (T + T) \\
&\Rightarrow i * (F + T) \\
&\Rightarrow i * (i + T) \\
&\Rightarrow i * (i + F) \\
&\Rightarrow i * (i + i) \quad \ldots \text{最左推导}
\end{aligned}
$$

$$
\begin{aligned}
E &\Rightarrow T \\
&\Rightarrow T * F \\
&\Rightarrow T * (E) \\
&\Rightarrow T * (E + T) \\
&\Rightarrow T * (E + F) \\
&\Rightarrow T * (E + i) \\
&\Rightarrow T * (T + i) \\
&\Rightarrow T * (F + i) \\
&\Rightarrow T * (i + i) \\
&\Rightarrow F * (i + i) \\
&\Rightarrow i * (i + i) \quad \ldots \text{最右推导}
\end{aligned}
$$

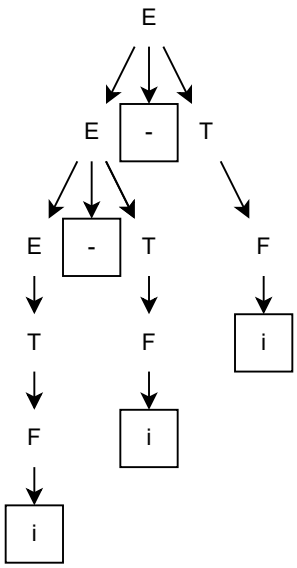
然后, 画出给定 表达式 的 语法树:



i+i+i



i+i*i



i-i-i

P36 T11

> 给出下面语言的相应文法:
> $L_1 = \{a^n b^n c^i | n \geq 1, i \geq 0\}$
> $L_2 = \{a^i b^n c^n | n \geq 1, i \geq 0\}$
> $L_3 = \{a^n b^n a^m b^m | n, m \geq 0\}$
> $L_4 = \{1^n 0^m 1^m 0^n | n, m \geq 0\}$

对于 L1, 一个可行的方案是:

$$S \to AC$$
$$A \to aAb|ab$$
$$C \to cC|\epsilon$$

对于 L2, 一个可行的方案是:

$$S \to AB$$
$$A \to aA|\epsilon$$
$$B \to bBc|bc$$

对于 L3, 一个可行的方案是:

$$S \to AA$$
$$A \to aAb|\epsilon$$

对于 L4, 一个可行的方案是:

$$S \to 1S0|T$$
$$T \to 0T1|\epsilon$$

补充作业

2.1

> 写一个上下文无关的文法 G, 使其语言为:
> $L(G) = \{a^n c^i b^n | n \geq 1, i \geq 1\} \cup \{b^n c^i a^n | n \geq 1, i \geq 1\}$

一个可行的方案是:

$$S \to M|N$$
$$M \to aMb|aCb$$
$$N \to bNa|bCa$$
$$C \to cC|c$$

三个注意点:

1. 语言中不允许出现 `abcab` 这种 `a` 和 `b` 交错出现的情况, 全部各出现在 `c` 一侧, 所以不能通过合并, 消除 `M` 和 `N` 这两个非终结符
2. 需要引入 `C` 这个非终结符, 用于推导 `ccc...ccc` 子串
3. 注意 `n > 0` 的条件, 也就是说 `a/b` 至少要出现一次, 所以说 `C` 不能直接成为 `M/N` 的 候选式

2.2

> 写一个上下文无关文法, 使其语言为:
> $L(G) = \{a^n c^m | n > m > 0\}$

一个可行的方案是:

$$S \to MN$$
$$M \to aM|a$$
$$N \to aNc|ac$$

应对这个问题, 一种较好的思路是:

$$\begin{aligned} L(G) &= \{a^n c^m | n > m > 0\} \\ &\simeq \{a^{y+x} c^x | x, y > 0\} \\ &= \{a^y a^x c^x | x, y > 0\} \end{aligned}$$

然后, 分别考虑 `a^y` 和 `a^x c^x` 子项 的 `推导`