2237340 王飞阳

编译第七次作业.

练习4-1:

1.

(1). E. T. F. P的四个产生式均无共同前缀，

故无需提取左公因子

(2) 不能，因为存在左递归

(3) E: $E \to TE'$

$E' \to +TE' \mid \varepsilon$

T: $T \to FT'$

$T' \to FT' \mid \varepsilon$

F: $F \to PF'$

$F' \to *F' \mid \varepsilon$

$P \to a \mid b$

(4) 消除左递归后，既无左递归，又不含左公因子，因此适用于

自顶向下分析.

练习4-2:

1. (1) 需满足：①对每个形如 $A \to x1 \mid x2 \mid \cdots \mid xn$ 的产生式，要求 FIRST(xi) 与 FIRST(xj) 的交集为空集（对）

原因：含左公因子会导致分析过程要穷尽所有可能的推导，会产生回溯

③对每个形如 $A \to x_1|x_2|\cdots|x_n$ 的产生式，若 $x_i \Rightarrow^* \varepsilon$，则要求 FIRST($x_i$) 与 FOLLOW(A) 的交集为空集(不相)

原因：防止产生回溯

④不含左递归　　原因：防止一直循环递归下去。

(2). 自顶向下的分析方法可以处理其他类型的递归文法，是因为这些递归文法不会导致无限循环的情况。

2. 消除左递归：

$A ::= (B) \mid dBe$

$B ::= cfc$.

分析程序见下面：

```
public class Parser {

    private static char CLASS;  // 当前的符号类型
    private static char[] input;  // 输入的符号流
    private static int index;  // 用于跟踪当前符号的位置

    public static void main(String[] args) {
        input = new char[]{'c', 'c', '(', 'c', 'd', 'e', ')'};
```

```
        index = 0;
        CLASS = input[index];

        try {
            G();
            System.out.println("Parsing completed successfully.");
        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
    }

    // 获取下一个符号
    private static void nextsym() {
        index++;
        if (index < input.length) {
            CLASS = input[index];
        } else {
            CLASS = '\0';  // 结束符号
        }
    }


    // 程序 G
    private static void G() throws Exception {
        A();
    }

    // 过程 A
    private static void A() throws Exception {
        if (CLASS == '(') {
            nextsym();
            B();
            if (CLASS == ')') {
                nextsym();
            }
            else {
                error();
            }
        }
        else if (CLASS == 'd') {
            nextsym();
            B();
            if (CLASS == 'e') {
                nextsym();
            }
            else {
                error();
            }
        }
        else {
            error();
        }
    }

    // 过程 B
    private static void B() throws Exception {
```

```
        if (CLASS == 'c') {
            nextsym();
            while (CLASS == 'c') {
                nextsym();
            }
        }
        else {
            error();
        }
    }

    // 错误处理
    private static void error() throws Exception {
        throw new Exception("Syntax error");
    }
}
```

3. 由题：FIRST(A)={c}, FIRST(B)={a}, FIRST(Z)={a, c}.

∴ FIRST(AcB)={c}, FIRST(Bd)={a}, FIRST(AaB)={c}

FIRST(c)={c}, FIRST(aB)={a}, FIRST(a)={a}.

(2). 不行，因为存在左递归.

(3). 改写文法：

Z::=AcB | Bd

A::=cA'

A'::=aBA' | ε

B::=aBA' | ε

程序如下：

```
public class Parser {

    private static char CLASS;  // 当前的符号类型
    private static char[] input;  // 输入的符号流
    private static int index;  // 用于跟踪当前符号的位置

    public static void main(String[] args) {
        input = new char[]{'c', 'c', '(', 'c', 'd', 'e', ')'};
        index = 0;
        CLASS = input[index];
```

```
        try {
            G();
            System.out.println("Parsing completed successfully.");
        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
    }

    // 获取下一个符号
    private static void nextsym() {
        index++;
        if (index < input.length) {
            CLASS = input[index];
        } else {
            CLASS = '\0';  // 结束符号
        }
    }


    // 程序 G
    private static void G() throws Exception {
        Z();
    }

    // 过程 Z
    private static void Z() throws Exception {
        if (CLASS == 'c') {
            A();
            if (CLASS == 'c') {
                nextsym();
                B();
            }
            else {
                error();
            }
        }
        else if (CLASS == 'a') {
            B();
            if (CLASS == 'd') {
                nextsym();
            }
            else {
                error();
            }
        }
        else {
            error();
        }
    }

    // 过程 A
    private static void A() throws Exception {
        if (CLASS == 'c') {
            nextsym();
            while (CLASS == 'a') {
                nextsym();
```

```
                B();
            }
        }
        else {
            error();
        }
    }

    // 过程 B
    private static void B() throws Exception {
        if (CLASS == 'a') {
            nextsym();
            if (CLASS == 'c') {
                A();
            }
        }
        else {
            error();
        }
    }

    // 错误处理
    private static void error() throws Exception {
        throw new Exception("Syntax error");
    }
}
```