编译原理第 8 次作业

1

试构造符号串翻译文法，将由一般中缀表达式文法所定义的中缀表达式：

```
<E> ::= <E>'+'<T> | <T>
<T> ::= <T>'*'<F> | <F>
<F> ::= '('<E>')' | 'i'
```

翻译成前缀表达式。

答案解析

```
<E> ::= @+<E>'+'<T> | <T>
<T> ::= @*<T>'*'<F> | <F>
<F> ::= '('<E>')' | @i'i'
```

2

构造符号串翻译文法，使其接受由 0 和 1 组成的任意输入符号串，并分别产生下面的输出符号串：

（1）输入符号串的倒置

（3）输入符号串本身

答案解析

答案不唯一，如果不支持空串也可以

（1）输入符号串的倒置

G[<S>]:

```
<S> ::= @0<S>'0' | @1<S>'1' | ε
```

（3）输入符号串本身

G[<S>]:

```
<S> ::= <S>@0'0' | <S>@1'1' | ε
```

3

以下的符号串翻译文法能做什么?

```
<s> -> @C'EN'@HI'GL'@N'I'@E'S'@SE'H'
```

答案解析

接受输入串 `ENGLISH`，输出 `CHINESE`。

可能是个英文翻译到中文的冷笑话

4

由特殊的翻译文法产生的两个活动序列是

```
@x@y'b'@z
@q'a'@x@y'b'@z@x@x@y'b'@z@y
```

由这个翻译文法删掉诸动作符号得到的输入文法 G[<S>] 是：

```
<s> -> 'a'<s><s>
<s> -> 'b'
```

这是个什么翻译文法?

答案解析

翻译文法是

```
<s> -> @q'a'<s>@x<s>@y
<s> -> @x@y'b'@z
```

补充题（代码）

有属性翻译文法：

$$
\begin{array}{ll}
E_{\uparrow x} \to E_{\uparrow q} + T_{\uparrow r}\ @\mathtt{ADD}_{\downarrow y,z,p} & (x,p) := \mathtt{NEW}; y := q; z := r \\
E_{\uparrow x} \to T_{\uparrow p} & x := p \\
T_{\uparrow x} \to T_{\uparrow q} \times F_{\uparrow r}\ @\mathtt{MULT}_{\downarrow y,z,p} & (x,p) := \mathtt{NEW}; y := q; z := r \\
T_{\uparrow x} \to F_{\uparrow p} & x := p \\
F_{\uparrow x} \to (E_{\uparrow p}) & x := p \\
F_{\uparrow x} \to \mathtt{id}_{\uparrow p} & x := p
\end{array}
$$

其中，`id` 的综合属性 p 是数据区地址；`NEW` 是系统过程，返回数据区地址。

语义动作程序：

```
ADD ↓(y, z, p) => print("ADD {} {} {}", y, z, p)
MULT ↓(y, z, p) => print("MULT {} {} {}", y, z, p)
```

答案解析

题目写的挺不清楚的……以下是助教理解的内容

首先可以注意到，第 1 和 3 个产生式是左递归的，于是我们可以进行消除左递归的操作，变形为相应的 EBNF 格式：

(1, 2): $E_{\uparrow x} \rightarrow T_{\uparrow q}\{+T_{\uparrow r}\ @\texttt{ADD}_{\downarrow y,z,p}\}$，其中对于每个循环都有赋值 $(x, p) \leftarrow \texttt{NEW}; y \leftarrow q; z \leftarrow r; q \leftarrow x$（注意每次迭代后 q 会变）；

(3, 4): $T_{\uparrow x} \rightarrow F_{\uparrow q}\{+F_{\uparrow r}\ @\texttt{MULT}_{\downarrow y,z,p}\}$，其中对于每个循环都有赋值 $(x, p) \leftarrow \texttt{NEW}; y \leftarrow q; z \leftarrow r; q \leftarrow x;$

于是参考伪代码如下：

```
def E():
    var x, p, y, z, q
    q = T()
    while next_is("+"):
        next()
        z = T()

        (x, p) = NEW()
        y = q
        q = x
        ADD(y, z, p)
    return x

def T():
    var x, p, y, z, q
    q = F()
    while next_is("*"):
        next()
        z = F()

```

```
        (x, p) = NEW()
        y = q
        q = x
        MULT(y, z, p)
    return x

def F():
    if next_is("("):
        next()
        var x = E()
        expect(")")
        return x
    else if next_is("id"):
        id = next()
        var x = id.p
        return x
    else:
        error()

def main():
    E()
    expect(右界符)
    return

def ADD(y, z, p): print("ADD {} {} {}", y, z, p)

def MULT(y, z, p): print("MULT {} {} {}", y, z, p)
```