Chapter 6

作业

1.1

> 已知文法:
>
> $$G(S):S\to aAbA$$
> $$A\to aSb|bSa|a$$

> 给出一种翻译方案, 统计输入句中 `a` 和 `b` 的个数

分别为 `S` 和 `A` 两个 `非终结符`, 添加两个属性:

- num_a
- num_b

对 `4` 个产生式, 依次定义语义规则:

产生式	语义规则
$S\to aA_1bA_2$	$\begin{cases}S.num_a=A_1.num_a+A_2.num_a+1\\S.num_b=A_1.num_b+A_2.num_b+1\end{cases}$
$A\to aSb$	$\begin{cases}S.num_a=A_1.num_a+A_2.num_a+1\\S.num_b=A_1.num_b+A_2.num_b+1\end{cases}$
$A\to bSa$	$\begin{cases}A.num_a=S.num_a+1\\A.num_b=S.num_b+1\end{cases}$
$A\to a$	$\begin{cases}A.num_a=1\\A.num_b=0\end{cases}$

最终, 由输入句规约后得到的 `AST` 上 `根节点` 的 $S.num_a$ 和 $S.num_b$, 即为句中 `a` 和 `b` 的个数

1.2

> 已知文法:
>
> $$G(P):P\to D$$
> $$D\to D;\ D|id:\ T|proc\ id;\ D;\ S$$

> 1. 给出一种翻译方案, 统计该程序一共声明了多少个 `id`
> 2. 给出一种翻译方案, 统计该程序中每个变量 `id` 的 `嵌套深度`

1.2.1

为 `P` 和 `D` 分别添加 num_id 属性, 对相关产生式, 依次定义语义规则:

产生式	语义规则
$P\to D$	$P.num\_id=D.num\_id$
$D_0\to D_1;\ D_2$	$D_0.num\_id=D_1.num\_id+D_2.num\_id$
$D\to id:\ T$	$D.num\_id=1$
$D_0\to proc\ id;\ D_1;\ S$	$D_0.num\_id=1+D_1.num\_id$

最终, 由程序规约后得到的 `AST` 上 `根节点` 的 $P.num\_id$, 即为程序中声明的 `id` 的个数

1.2.2

首先为 `id` 添加 $name$ 属性, 用于描述 `id` 的 `名称`

再为 `P` 和 `D` 分别添加 id_table 属性, 它是一张维护了 `id.name` 到 `integer` 的 `映射表` (类似于 `HashMap<id, depth>`), `integer` 部分即为 `id.name` 对应的 `变量` 的 `嵌套深度`

随后, 为相应的产生式, 依次定义语义规则:

- $P\to D$

$$P.id\_table = D.id\_table$$

- $D_0 \to D_1;\ D_2$

$$\forall (\_, depth) \in D_2.id\_table :: depth = depth + 1$$
$$D_0.id\_table = \text{UNION}(D_1.id\_table\ ,\ D_2.id\_table)$$

- $D \to id:\ T$

$$D.id\_table = \{id.name : 0\}$$

- $D_0 \to proc\ id;\ D_1;\ S$

$$\forall (\_, depth) \in D_1.id\_table :: depth = depth + 1$$
$$D_0.id\_table = \text{UNION}(\{id.name : 0\}\ ,\ D_1.id\_table)$$

最终, 由程序规约后得到的 `AST` 上 `根节点` 的 $P.id\_table$, 包含了所有由 `id.name` 和 `depth` 组成的 `键值对`, 即为程序中每个变量 `id` 的 `嵌套深度`

2.1

> 课本 $P_{164}\ T_7$
>
> 下列文法由开始符号 S 产生一个二进制数, 令综合属性 `val` 给出该数的值:
>
> $$S \to L.L|L$$
> $$L \to LB|B$$
> $$B \to 0|1$$
>
> 已知 B 的综合属性 `c`, 给出由 B 产生的二进制位的值; 试设计求 $S.val$ 的 `属性文法`

这里, 我们首先需要为 L 设计一个继承属性 `is_frac`, 它是一个 `boolean` 值, 要么为 `true`, 要么为 `false`, 用于描述当前 `L` 是否为 `小数部分`

再为 L 设计一个综合属性 `v`, 给出由 L 产生的二进制数的值

还需要为 L 设计一个综合属性 `l`, 给出由 L 产生的二进制数的长度

随后, 为相应的产生式, 依次定义语义规则:

- $S \to L_0.L_1$

$$
\begin{aligned}
L_0.is\_frac &= false \\
L_1.is\_frac &= true \\
S.val &= L_0.v + L_1.v
\end{aligned}
$$

- $S \to L$

$$
\begin{aligned}
L.is\_frac &= false \\
S.val &= L.v
\end{aligned}
$$

- $L_0 \to L_1B$

$$
\begin{aligned}
L_1.is\_frac &= L_0.is\_frac \\
L_0.l &= L_1.l + 1 \\
L_0.v &= \begin{cases} L_1.v \times 2 + B.c & (L_0.is\_frac = false) \\ L_1.v + B.c \times 2^{-L_0.l} & (L_0.is\_frac = true) \end{cases}
\end{aligned}
$$

- $L \to B$

$$
\begin{aligned}
L.l &= 1 \\
L.v &= \begin{cases} B.c & (L.is\_frac = false) \\ B.c \times 2^{-1} & (L.is\_frac = true) \end{cases}
\end{aligned}
$$

这样, 就可以求出 `S.val` 的值

2.2

> 课本 P_{165} T_{11}
>
> 下列文法, 可以生成变量的类型声明:

$$
\begin{aligned}
&D \to id\ L \\
&L \to , id\ L \mid :T \\
&T \to integer \mid real
\end{aligned}
$$

> 构造一个翻译模式, 把每个标识符的类型存入符号表 (可以参考, 课本本章 `例 6.2`)

$$
\begin{aligned}
&D \to id\ L\ \{\mathbf{addtype}(id.entry, L.type)\} \\
&L \to , id\ L_1\ \{L.type = L_1.type; \mathbf{addtype}(id.entry, L.type)\} \\
&L \to : T\ \{L.type = T.type\} \\
&T \to integer\ \{T.type = integer\} \\
&T \to real\ \{T.type = real\}
\end{aligned}
$$

其中, `addtype(id, type)` 把对应 `标识符` 的 `类型` 填入 `符号表` 的对应项中 (符号表的每个 `入口` 由属性 `entry` 指明)

该文法中, 标识符表的最后一个元素始终为 `类型`, 成功地实现了 `用综合属性替代继承属性`