例题 6.1.1 下列文法对整型常数和实型常数施用加法运算符 + 生成表达式；当两个整型数相加时，结果仍为整型数，否则，结果为实型数：

E→E+T | T

T→num．num | num

(1) 试给出确定每个子表达式结果类型的属性文法；

(2) 扩充(1)的属性文法，使之把表达式翻译成后缀形式，同时也能确定结果的类型。应该注意使用一元运算符 inttoreal 把整型数转换成实型数，以便使后缀形如加法运算符的两个操作数具有相同的类型。

解题思路： 确定每个子表达式结果类型的属性文法是比较容易定义的。关键是如何扩充此属性文法，使之把表达式翻译成后缀形式。我们将不在 name 或 num.num 向 T 归约的时候输出该运算对象，而是把运算对象的输出放在 T 或 E+T 向 E 归约的时候。这是因为考虑输出类型转换算符 inttoreal 的动作可能在 E+T 向 E 归约的时候进行，如果这时两个运算对象都在前面 name 或 num.num 向 T 归约的时候已输出，需要为第一个运算对象输出类型转换算符时就已经为时太晚。

还要注意的是，在 E+T 向 E 归约时，该加法运算的第一个运算对象已经输出．所以 E→E + T 的语义规则不需要有输出 E 运算对象的动作。

解答:

(1) 为文法符号 E 和 T 配以综合属性 type，用来表示它们的类型。类型值分别用 int 和 real 来表示。确定每个子表达式结果类型的属性文法如下。

产生式　　语义规则

E->E1+T　　T.type:= if E1.type =int and T.type = int then int else real

E->T　　E.type:=T.type}

T->num.num　　T.type :=real}

T->num　　T.type:=int}

(2) 下面属性文法将表达式的后缀表示打印输出，其中 lexeme 属性表示单词的拼写。

产生式　　语义规则

E->E1+T　　if E1.type = real and T.type = int then

begin

E.type :=real;

```
                    print(T.lexeme);
                    print('inttoreal')
                  end
                else if E1.type = int and T.type=real then
                  Begin
                    E.type :=real;
                    print('inttoreal') ;
                    print(T.lexeme)
                  end
                else begin
                    E.type :=E1.type;
                    print(T.lexeme)
                  end;
    print ('+');
  E->T    E.type:=T.type;
                    print(T.lexeme)
  T->num1.num2  T.type:=real;
                    T.lexeme:=num1.lexeme|| ". "||num2.lexeme
T->num    T.type:=int;
                    T.lexeme:=num.lexeme
```