Attribute Grammar – Part 1

```
<prog>     ::= <block>
               <block>.alltbl := emptystack
<block>    ::= begin <declist>; <stmtlist> end
               <stmtlist>.alltbl := push(<declist>.tbl, <block>.alltbl)
<declist>1 ::= <decl>
               <declist>1.tbl := <decl>.tbl
             | <decl> ; <declist>2
               <declist>1.tbl := <decl>.tbl ∪ <declist>2.tbl
               Cond: ids(<decl>.tbl)∩ids(<declist>2.tbl)={}
<decl>     ::= int <id>
               <decl>.tbl := { (id.lexval, INT) }
             | bool <id>
               <decl>.tbl := { (id.lexval, BOOL) }
<stmtlist>1::= <stmt>
               <stmt>.alltbl := <stmtlist>1.alltbl
             | <stmt> ; <stmtlist>2
               <stmt>.alltbl := <stmtlist>1.alltbl
               <stmtlist>2.alltbl := <stmtlist>1.alltbl
<stmt>     ::= <assign>
               <assign>.alltbl := <stmt>.alltbl
             | <block>
               <block>.alltbl := <stmt>.alltbl
```

```
begin
 bool i;
 int j;
 begin
  int x;
  int i;
  x := i + j;
 end
end
```

[
{("x",INT), ("i",INT)},
{("i",BOOL), ("j",INT)}
]

bottom of stack