CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
Phases of a compiler

Figure 1.6, page 5 of text
Attribute grammars

- Attribute grammars provide a neater way of encoding such information.
- Each syntactic rule of the grammar can be decorated with:
  - a set of semantic rules/functions
  - a set of semantic predicates
Attributes

- We can associate with each symbol $X$ of the grammar a set of attributes $A(X)$. Attributes are partitioned into:

  synthesized attributes $S(X)$ – pass info up parse tree

  inherited attributes $I(X)$ – pass info down parse tree
Example

<assign> → <var> = <expr>
<expr>.expType ← <var>.actType

<expr>.actType ← if (var[2].actType = int) and 
      (var[3].actType = int)
      then int
      else real
<expr>.actType == <expr>.expType

<expr> → <var>
<expr>.actType ← <var>.actType
<expr>.actType == <expr>.expType

<var> → A | B | C
<var>.actType ← lookUp(<var>.string)
Let's see how these rules work in practice!

In this example A and B are both of type int.

Suppose:
A is int
B is int
Suppose:

A is int
B is int
A = A + B

Suppose:
A is real
B is int

This is the same example structure, but now assume A is of type real and B is of type int.
This is the same example structure, but now assume A is of type real and B is of type int.

Suppose:
A is real
B is int
This is the same example structure, but now assume A is of type real and B is of type int.

Suppose:
A is real
B is int
This is the same example structure, but now assume A is of type real and B is of type int.

Generate code to do conversion.

Suppose:
A is real
B is int
Suppose:

A is int
B is real
This is the same example structure, but now assume A is of type int and B is of type real.

Suppose:
A is int
B is real
Houston, we have a problem! Semantic predicate is **false**.

Suppose:
- **A** is int
- **B** is real

\[
\text{A = A}[2] + \text{A}[3]
\]
A = A + B

Suppose:
A is int
B is real
Project Questions?