CS 350: Programming Language Design

Lecture 6

Attribute grammars

Context-free grammars with the addition of

* For each symbol, there is a set of attribute values consisting of
  + Synthesized attributes
  + Inherited attributes
  + Intrinsic attributes on the leaves
    - Symbol table -> declaration
* Each rule has
  + A set of functions that define certain attributes of the non-terminals in the rule
  + A set of predicates (Booleans) to check for attribute consistency
    - A false predicate function value indicates a violation of the syntax or static semantics rules of the language

Rule -> the name on the end of an Ada Procedure must match the procedure’s name

Syntax rule: <proc\_def> -> procedure <proc\_name>[1]

<proc\_body> end <proc\_name> [2];

Predicate: <proc\_name> [1].string == <proc\_name>[2].string

Why must we use 1 and 2 when the name will be the same?

Since, we can use multiple procedure names we can differentiate using the subscripts indicated above.

When we evaluate rules syntactically, we only have valid expressions when the value from our declaration matches the variable from our right hand expression.

Semantic rules:

* When there are two variables on the right side of an assignment
  + If, they have the same type, the expression type is that of the operands
  + If the operand types are not the same is always real
* The type of the left side assignment must match the type of the right side.

Attributes

* Actual\_type: synthesized for <var> and <expr>
  + Used to store the actual type, int or real, of a variable or expression
  + Variable -> actual type is intrinsic
  + Expression -> determined from the actual types of the child node or children nodes of the <expr> non-terminal
* Expected\_type: inherited for <expr>
  + Used to store the type, int or real, that is expected for the expression
  + Determined by the type of the variable on the left side of the assignment statement.

Decorating parse trees

* How are attribute values computed?
  + If all attributes were inherited
  + The tree is decorated top-down
* If all were synthesized
  + The tree is decorated bottom-up
* In many cases, both kinds are used
  + Some combination must be used.

Static semantics is vital for compilers of any type

When used to describe modern PLs, attribute grammar becomes large in size and complex

Decorating parse trees is an expensive process.

Dynamic Semantics

* Dynamic semantics