Attribute Grammar Examples and Symbol Tables

66.648 Compiler Design Lecture (02/23/98)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- Examples
- Symbol Tables
- Administration

Example

Question 5.5, 5.4 and 5.8 in the text book.

Qn: 5.5 Give a syntax directed translation scheme to differentiate expressions.

D/dx (x*x + x) = 1*x + x* 1 + 1.

How do we do it?

Yet Another Example

qn 5.8 To get the value of a binary number:

s --> L.L | L

L --> L B | B

B --> 0 | 1

We can do the problem in two steps:

S--> L and S --> .L . Then how do we combine

Yet Another Example

Qn. 5 .4 Give a syntax directed translation scheme to remove extra parentheses.

((x * x)) = x * x

(x*x) +x x * x +x

(x+x) *x = (x+x) * x

How do we do this in terms of attributes?

Simple Declarative language

In this example, we will explore how we take care of identifiers and types.

Symbol Table

The symbol tables - repository of all information within a compiler. All parts of a compiler communicate thru these table and access the data-symbols. Symbol tables are also used to hold labels, constants, and types.

Symbol tables map names (constants, identifiers, label numbers) into sets of symbols. Different names have different attributes. - for identifiers, it could be its type, its location in a stack frame, its storage class etc.

Symbol Tables- Contd

Symbols are collected into symbol tables. The symbol-table module manages symbols and tables.

Symbol Management should also deal with scope and visibility that is imposed by the language.

Symbol Tables are usually lists of hash tables, one for each scope.

Symbol Table- Contd

A typical table (see Fraser and Hanson book page 40)

```
typedef struct table *Table;
struct table {
int level ; /* scope value */
Table previous;
struct entry { struvt symbol sym;
               struct entry *link; } *buckets[256];
Symbol all; } ;
```

Symbol Table- Contd

The buckets field is an array of pointers to the hash chains. Previous field points to the table of the enclosing scope. In each table structure all heads a list of symbols in this and enclosing scopes.

Symbol Table entries are (can be) non-uniform (variable sized). Depending upon what the identifier represents. (We will see how a symbol is defined soon).

Symbol Table look-up is performed in the context of a current scope , global, local, structured type.

Symbol Table Interface

Create_scope(parent_scope) - return index for a new scope contained in a parent scope.

Insert_scope(scope,name) - insert identifier into the specified scope of a symbol table.

Lookup(scope,name) - look up identifier in specified scope.

Delete_scope(scope) - delete specified scope and all symbol table entries that it contains

Symbols

```
Typedef struct symbol *Symbol;
struct symbol { char *name;
                int scope;
                Coordinates src; Symbol up;
                List uses; int sclass; /*storgeclass*/
                float ref;
                union  { constants,,function
   symbols, globals, temporaries} u;
   Xsymbol x; /* debugger infmtion */ } ;
```

U field

The u field supplies additional data for labels, structure and union types, static variables and temporary variables.

We will discuss this in later classes.

Comments and Feedback

Please read chapter 6 and look at the sample compilers in the course home page to get an over all picture. All the programs discussed today are also in the home page.