Principles of Programming Languages - Midterm (Solutions)

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Problems

1. Enumerate the different times at which the attributes of a name can be bound. Give one example for each.

   Solution: The different times at which the attributes of a name can be bound are as follows:

   (a) Language Definition Time (Predefined identifiers),
   (b) Language Implementation Time ($maxint$ in Pascal),
   (c) Language Translation Time (Static Declarations),
   (d) Link Time (externally defined functions),
   (e) Load Time (global variables),
   (f) Execution Time (Reading input data).

   □

2. Briefly explain with examples, the concepts of visibility, scope and scope holes, associated with bindings.

   Solution: Consider the following fragment of a **C** program.

```
int x;

main()
{
   char x;
}
```

   The variable x has been declared to be an integer globally, whereas in the function "main", it is a variable of type *char*. Both declarations result in different bindings for the name x. The scope of a binding is the region of the program over which the binding is maintained. The binding "int x" is maintained within the function "main", but it is not visible, since within "main", x has a different set of attributes, i.e., the binding "int x" suffers from a scope hole.

   □

3. Consider the following Expression Grammar:

$$\begin{aligned} expr &\rightarrow expr * expr \mid expr / expr \mid number \\ number &\rightarrow digit \ \ number \mid digit \\ digit &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

   Identify the problems with the above grammar.

   Solution: There are two problems with the above grammar, viz.,

(a) Ambiguity between operators - The string $2/3 * 5$ has two different parse trees, one evaluating to $\frac{2}{15}$ and the other evaluating to $\frac{10}{3}$.

(b) Ambiguity in the same operator (Associativity) - The string $2/3/4$ has two different parse trees, one evaluating to $\frac{2}{12}$ and the other evaluating to $\frac{8}{3}$.

□

4. Write a new set of rules for the above Expression Grammar such that the problems that you have identified are fixed.

   Solution: There are two ways to approach the problem of removing ambiguity. We could enforce a precedence rule, viz., that division takes precedence over multiplication. Note that this does not correspond to the natural rules of mathematics. This would result in the following grammar:

$$\begin{aligned} expr &\rightarrow expr * term \mid term \\ term &\rightarrow term \ / \ number \mid number \\ number &\rightarrow digit \ \ number \mid digit \\ digit &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

   Alternatively, we could use the following grammar:

$$\begin{aligned} expr &\rightarrow expr * number \mid expr \ / \ number \mid number \\ number &\rightarrow digit \ \ number \mid digit \\ digit &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

   Both grammars result in unique parse trees for every terminal string.

   □

5. Enumerate the different ways in which Computer Scientists have tried to specify semantics of programming languages, over the years. Identify a shortcoming for each method employed.

   Solution: The three principal ways of specifying semantics are as follows:

   (a) Language Reference Manuals. The drawback is that there does not exist a universal interpretation of the English language (or any other language, for that matter!)

   (b) Defining Translators. The drawback is that the meaning of a code fragment is understood only after executing it on the translator, i.e., there is no *a priori* understanding of the semantics.

   (c) Formal Definitions. All attempts at axiomatizing semantics through formal methods have resulted in extremely complex procedures for specifying the simplest of structures.

   □