CS4610 Programming Languages HW1

Group: William Loving (wfl9zy), etc,

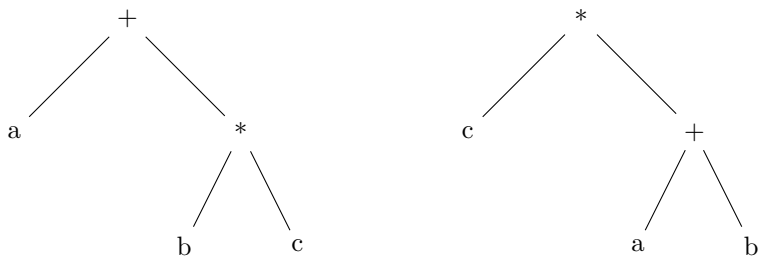
February 1, 2024

1. Show that the following grammar is ambiguous:

$$\text{<exp>} ::= \text{<exp>} + \text{<exp>} \mid \text{<exp>} * \text{<exp>} \mid (\text{<exp>}) \mid a \mid b \mid c \quad (1)$$

The grammar here is ambiguous because of the lack of operator precedence with regards to this grammar design. Without operator precedence, scenarios like **a + b * c** are ambiguous as we don't know if the language will first evaluate the multiplication or the addition. With operator precedence we can enforce certain operators to be evaluated before others such that the above example always evaluates **b * c** before **a + b** eliminating the ambiguity for both users of the language and parse-tree/code generators.

Two Possible Trees:



With this higher precedence set the language will always produce the first tree and the one liner would be something like **a + (b * c)** such that any ambiguity is eliminated.

2. In class, we discussed that different bindings take place at different times. One standard way of describing binding times is as follows:

- Language definition time
- Language implementation time
- Compile time
- Link time
- Load time
- Runtime

Label each of these scenarios with a binding time (C++):

- The location in memory of a local variable in a function.
  - Runtime
- The meaning of the keyword while.
  - Language definition time
- The size in memory of a variable of type int.
  - Language implementation time
- The location in memory of a global static variable.
  - Load time
- The code for the printf (or equivalent) function.
  - Link time (If using static linking, all of printf will be added to executable)
  - Load time (If using dynamic linking, addresses to printf resolved as executable is loaded into memory)
  - Runtime (If using lazy linking where printf will only be resolved when it is actually called within the code)
- The type of a local variable in a function.
  - Compile time
- The value(s) assigned to a variable.
  - Runtime
- The size in memory of a pointer.
  - Language implementation time

Label each of these scenarios with a binding time (Java):

- The location in memory of a local variable in a function.
  - Runtime
- The meaning of the keyword while.
  - Language definition time
- The size in memory of a variable of type int.
  - Language implementation time
- The location in memory of a global static variable.
  - Load time (Java does not have global "static" variables but will instead have static variables as members of classes, these are typically initialized at load time and can be called without instantiating the class itself)
- The code for the printf (or equivalent) function.
  - Runtime (Java uses dynamic linking and can load/modify functions when they are called allowing greater flexibility and OO features like polymorphism)
- The type of a local variable in a function.
  - Compile time
- The value(s) assigned to a variable.
  - Runtime
- The size in memory of a pointer.
  - Language implementation time (Java abstracts away pointers, but references still exist and would be defined at implementation time)