UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS 264 **S. L. Graham**
Fall 1995 **A. B. Downey**

Assignment 4: Attribute Grammars

Due: Friday, 3 November 1995

The purpose of this assignment is to strengthen your knowledge of the use of attribute grammars. Please give a hardcopy of your solution to Allen.

1 Write an Attribute Grammar

Write a context-free grammar for parenthesized arithmetic expressions that contains the operators '+', '-', '*', '/', and $\uparrow$, where $\uparrow$ is right-associative and has highest precedence, the other operators are left associative and have their usual precedences. Parentheses are used to override the built-in associativities and precedences. Use the simple ambiguous grammar and assume that the bison-like parser "does the right thing" in building a parse tree.

Write an attribute grammar that prints out the expression in infix form and that checks each of the following properties, generating an error message if the property is not satisfied. You may invent the side-effect-free attribution language, or take it from one of the papers. Be sure to 'declare' the attributes, indicating for each whether it is inherited or synthesized. To clarify your answer, explain how each of the checks is carried out (i.e. provide some intuition).

1. The maximum depth of nested parentheses is 3.
2. The maximum number of operators used in an expression is 10.
3. If an expression has more divisions than multiplications, then it may not have any subtractions. (Don't ask me why!)
4. All the parentheses are necessary; i.e., the associativity or precedence would be different if they were omitted.

Note: the first 3 conditions are taken from a homework exercise in a textbook by Slonneger and Kurtz.

2 Evaluate the Use of Attribution Methods in your Cadaver

Sketch an attribute grammar, over an idealized abstract grammar, for doing the static semantic analysis done in your cadaver. What is needed here is the high-level design, not a specific grammar and rules. If you need to refer back to things you said in your solution to assignment 3, restate them briefly so that this document is self-contained.

More precisely:

1. Describe the analysis (in English). If there isn't very much, i.e. if it's deferred to runtime, describe some analyses that could be added (what things are done, or could be done: name-resolution? type-checking? overloading-resolution? type-inference? use-analysis? What sort of error messages can be generated?)

2. Sketch how the same task could be done in an attribute grammar. What attributes would be necessary? What form would the dependencies take?

3. If you were to use remote attribution, what sort of remote dependencies would exist? In what order could operations be done?

4. Would the result be L-AG? multi-sweep? or more complicated? Why?

Note: this question is based on a suggestion from John Boyland
Note from Allen: in previous assignments, many of you have gone to great lengths to avoid insulting my intelligence by providing unnecessary background on the source language of your compilers. Although I appreciate the implied faith in my abilities, actually it would be very helpful if you would include with your description enough information about the source language that someone — and I'm not naming names — completely unfamiliar with the language would be able to read your homework.