Computer Science Department
Stanford University
Comprehensive Examination in Compilers
Autumn 1995

October 19, 1995

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. Be sure you have all the pages of this exam. There is 1 page.

   The exam takes 30 minutes.

3. This exam is OPEN BOOK. You may use notes, articles, or books—but no help from other sentient agents such as other humans or robots.

4. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

Compilers

There are a number of essay questions on this exam. You should, of course, keep your answers clear and concise.

1. (5 minutes) Answer true or false. In all questions, assume that the context-free grammar has no useless symbols.

   F (a) There exists a context-free grammar with that is LL(1) but not LR(1).

   T (b) There exists an context-free language that is LR(1) but not LL(1).

   T (c) YACC can correctly parse all ambiguous context-free grammars by using precedence and associativity hints.

   (d) There exists an unambiguous grammar that is not LR(1).

   F (e) LALR(1) parser generation (*not parsing*) is linear in the size of the input grammar.

2. (10 minutes) Consider the following grammar:

   $$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow \epsilon \\ B &\rightarrow b \\ B &\rightarrow SC \\ C &\rightarrow c \end{aligned}$$

   (a) Compute the FIRST and FOLLOW sets for the nonterminals of the grammar.

   (b) Based on the FIRST and FOLLOW sets, is it LL(1)? Explain your answer.

   (c) What is the language of this grammar? From inspecting the language, is it obvious whether there exists an LL(1) answer (if you say it's obvious, explain [obviously]).

3. (10 minutes) These questions are about the handling of variables by a compiler for a simple language like C. In your answers, address only the compiler behavior that is *necessary for code generation*. Do not address type checking or other aspects of semantic analysis are not strictly necessary to emit code for correct programs.

   (a) Describe the compiler's processing of a global variable g of type `int`, both at the point of *declaration* and at the point of *use*.

   (b) How is the handling of a local variable declaration of type `int` different?

   (c) What information does the compiler have to maintain in the symbol table to generate code for `A[i].f[j]`?

4. (5 minutes) Imagine a language in which the types of expressions can be determined in a single bottom-up traversal of the syntax tree of a program. The language also has implicit type conversions (also called *coercions*), such as `int` to `float`.

   How could a compiler

   - recognize when coercions need to occur, and
   - generate code to perform them

   in a single bottom-up traversal of an expression?

(18)