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

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

Assignment 2: Syntax

Due: Tuesday, 26 September 1995

The purpose of this assignment is to increase your experience with parsers as they are used in practice. Please mail your solution, in plain text form, to downey@cs.

1 Evaluate Cadaver Syntax Analyzer

Study the way syntax analysis is done in your Cadaver compiler and relate it to the material discussed in class. Following are some examples of issues that would be interesting to discuss. You should add any others that you discover or that you think are relevant. The purpose of these questions is to motivate you to understand details about how the source language is handled, not just find out (superficially) what category of parser is used.

- What parsing method is used? If it is table-driven, what is the table-building tool? Is the parser itself provided by the tool? If it is hand-coded, is there any formal specification of the syntax that is parsed?
- Is there a clean interface between parser and semantics? If so, how can you tell? If not, why not?
- Does the parser use semantic information to make parsing decisions? If so, explain how and why.
- If the parser is LALR or full LR, are there some grammar situations in which non-SLR lookahead is needed? Explain how you determined that (no matter which way the answer goes). If yes, give at least one example. If you can, show what the corresponding SLR grammar would look like.
- If the parser is hand-written, describe its organization. Does it parse in a deterministic context free manner (i.e. does it mimic a table-driven parser)? If so, explain what would be entailed in replacing the parser with a Bison or Yacc parser. If not, explain in what ways the programmed solution deviates from the equivalent formal solution (e.g. give an example in which the programming flexibility is exploited).
- What is the design of the error handling? Is there evidence that it is tuned to this source language? Give some examples, or explain why not.
- How robust is the compiler in the face of errors? How good are the diagnostics? How did you determine that?

2 Learn about the Other Side

If we had the time, I would ask each of you whose cadaver has a table-driven parser to rewrite the parser in recursive descent and vice-versa. To attempt to provide the same experience, choose a partner in the class whose cadaver uses the opposite technology from yours (i.e. if your parser is hand written, team up with someone whose parser is table-driven). Review with your partner his or her answers to part I of the assignment. (You can also provide feedback on what that person proposes to hand in!) Write a brief summary contrasting the pros and cons of how syntax analysis is done in your compiler with how it is done in your partner's.

I strongly suggest that you answer part 1 (and perhaps part 3) in advance of the normal class meeting on the 26th, and use that time to meet with your partner.

3 Modify the Language Syntactically

Modify the syntax used by your cadaver by (1) adding a new operator and a new data type and (2) changing the procedure/function interface so that all procedures/functions must have at least two arguments.

- The purpose of the assignment is to get experience with syntax. Modify the syntax in place (i.e. modify the compiler) if that is feasible. However, if that is difficult for compilation or other engineering reasons, just extract the scanner and parser and modify them.
- Choose your own addition, since it's impossible to give one proposal for all the languages. You might wish to float your idea to the newsgroup. The goal is to make a nontrivial syntactic extension, whether or not it is good language design.
- Discuss what was involved in making the change. How easy or difficult is it to understand where the changes need to be? To insure that the resulting analyzer is correct? To debug the analyzer?

4 Design an Error Handling Strategy for an Incremental Parser

Note: this part of the assignment has nothing to do with your cadaver compiler

In class, we (will) have discussed error recovery in compilers, and also the use of syntax analysis in an interactive incremental setting. In the case of a syntax error, the situation is different than in compilation in that

- The user may choose not to repair the error right away, but will still want the structure-based services to continue.
- After the initial stages, most of the program will be analyzed before the error recovery strategy is invoked. In particular, the left-to-right bias is much less in evidence.

Design an error handling strategy for a bottom up incremental parser. The design goals should be to limit the error as much as possible (i.e. to maintain as much valid structure as possible) and to confuse the user as little as possible. Consider both what the internal mechanisms are and what is visible to the user.