Final Project: Unifier

CS 440: Programming Languages and Translators

Due Wed May 1, 11:59 pm

Unifier

The final project is a program to find the most general unifier for a set of textual equations. (See Lecture 18 for a discussion of unification.) Your (Haskell) program should define a function `solve` :: `String -> String` that (1) Takes a string representation of a unification problem, (2) Parses it using a recursive descent parser, (3) Uses the unification algorithm to solve the problem, and (4) Returns a pretty-printed version of the solution (or "`FAIL`" if there is no solution).

- The syntax for a unification problem is:

  Problem → { *Equation EquationTail* }

  Equation → *Term* = *Term*

  EquationTail → `,` *Equation EquationTail* | ε

  An example: `{ f(X, b) = f(Y, Z), g(c) = Y }`
- The syntax for Term should be the one from Homework 6 with some small changes:
  - A *Variable* includes only upper-case letters. As a regular expression, a *Variable* is `[A-Z][A-Z0-9_]*`
  - Similarly, an *Identifier* includes only lower-case letters: `[a-z][a-z0-9_]*`
  - We'll also include nonnegative integer constants: *Constant* has the form `[0-9][0-9]*`
- The datatype for `Term` should include `CONST Int` (in addition to `Var`, `ID`, and `FCN`).
- The unification algorithm is the one discussed in Lecture 18.
- To pretty-print the solution, follow the syntax above for Problem, with the left-hand *Term* of each equation restricted to be a variable. For example, a solution to the problem `{ f(X, b) = f(Y, Z)), g(c) = Y }` might be "`{X = g(c), Z = b, Y = g(c)}`". (You're encouraged to add a space or two in places to make the solution more readable.)

Collaboration

This is a one-person project. (Sorry, no teams.) On the other hand, if there's code you can use from a homework assignment, you can use it, even if you did the homework assignment as part of a team. This includes the parser from Homework 5 and the substitution functions from Homework 6. (The unification algorithm is basically the one-person part of the assignment.)

With the discussion in Lecture 18 and the code in Homeworks 5 and 6, this should be enough information for you to do the whole project, but we'll discuss it more in class.

[Added 4/9: There's a Haskell framework attached to this handout -- it includes the basic types and data structure declarations, the names of some functions you should implement (and the implementation of a few basic routines). All the actual parsing / unifying / pretty-printing code has to be written, of course.]