Type Systems for Programming Languages

Assessed Course Work

Hand-out November 21 2011, Hand-in December 7 2011

Question Show that ML-substitution is a sound operation in the ML type assignment system. You can use $(S\phi)[(S\sigma)/\phi] = S(\phi[\sigma/\phi])$, and can assume that free and bound (by for-all) type-variables are distinct.

(This is the only time I will ask for a proof.)

Question Extend the Lambda Calculus with the *conditional* language construct '`if` M `then` P `else` Q'. Add any other constant or construct that might be needed to make this work correctly and extend, if needed, the notion of reduction to deal with this addition.

Give the natural extension to Curry's type assignment system, by presenting -if necessary- new types and rules, for the expressions that deal with this new construct.

Extend the principal context algorithm with a case for the new construct.

Question Assume the existence of characters, numbers, test for zero and (pre-fix) addition, as well as the abstract data type *List* in the ML language. Let $[\phi] = \mathit{List}\ \phi$, *Tail* be a function of type $\forall\phi.[\phi]\to[\phi]$, and '$=[\,]\ l$' be a test if l is the empty list. Build the ML expression that calculates the length of a list, and build a derivation that types this ML expression.

The derived type for length might be small, but notice that you would, formally, need to repeat the WHOLE derivation when you use length. Instead, you can abbreviate that (and write just the conclusion in the bottom line) by D.

Question Assume that *Cons* has type $\forall\phi.\phi\to[\phi]\to[\phi]$; using your answer to the previous question, give a derivation for the (translation of the) expression

$$\mathit{Length}\ (\mathit{Cons}('a',\mathit{Cons}('b',[\,]))) + \mathit{Length}\ (\mathit{Cons}(1,\mathit{Cons}(2,\mathit{Cons}(3,[\,]))))$$

that types this ML-expression (do not forget `let`!).

Notice that length is used on objects of different type here: i.e. polymorphic, so this calls for the use of `let`. It is possible to type it without, but that would breach the idea of polymorphism, since you would be specifying length twice.

I want to see the whole derivation (just this once).

Question Show that $(\lambda xy.y(xxy))(\lambda xy.y(xxy))$ is a fixed-point constructor for the Lambda Calculus.

Extending ML with equi-recursive types, and using the recursive type $B = \mu X.X\to(\phi\to\phi)\to\phi$, show that $(\lambda xy.y(xxy))(\lambda xy.y(xxy))$ is typeable with $(A\to A)\to A$, for all A. (Hint: first derive $\vdash\lambda xy.y(xxy) : B\to(\phi\to\phi)\to\phi$).