

CMCS 321: Programming Languages

Homework #7

Guest Instructor: Derek Dreyer (dreyer@tti-c.org)

Assigned: Thursday, November 16, 2006

Due: Thursday, November 30, 2006

The goal of this homework assignment is to give you some hands-on experience with ML type inference. Below is the abstract syntax of a subset of ML, represented using ML `datatype`'s.

```
type var = string
datatype typ = Tyvar of var
             | Int | Bool | Arrow of typ * typ
             | List of typ
datatype ptyp = Poly of var list * typ
datatype exp = Var of var
             | Fix of var * var * exp
             | App of exp * exp
             | Num of int
             | Plus of exp * exp
             | Eq of exp * exp
             | True
             | False
             | If of exp * exp * exp
             | Nil
             | Cons of exp * exp
             | Listcase of {list : exp,
                           nilcase : exp,
                           conscase : var * var * exp}
             | Let of var * exp * exp
```

We use here an explicit representation of variables as strings. For example, the polytype $\forall(\alpha_1, \dots, \alpha_n). \tau$ is represented as `Poly(["alpha_1", ..., "alpha_n"], τ)`, where references to the α_i in τ are written `Tyvar("alpha_i")`. The particular strings used to represent the variables are arbitrary. You can assume for the purpose of this assignment that all bound variables in the abstract syntax of the source program are distinct strings.

1 Extending the Declarative and Algorithmic Typing Judgments

The language described above has more features and types in it than the implicitly-typed language studied in class. In particular, the language here has integers, booleans and lists (in addition to functions). The new operations for these data types are fairly self-explanatory, but just to clarify a few points:

- `Fix(f,x,e)` corresponds to `fix f(x:τ1):τ2 is e end` from Homework #5, except that τ_1 and τ_2 are to be inferred.
- `Eq(e1,e2)` tests whether two integers are equal and returns a boolean.
- `If(e1,e2,e3)` corresponds to the ML expression `if e1 then e2 else e3`.
- `Listcase {list=e1,nilcase=e2,conscase=(h,t,e3)}` corresponds to the ML expression `case e1 of [] => e2 | (h::t) => e3`.

Show how to extend the declarative typing judgment, $\Gamma \vdash e : \tau$, and the Algorithm \mathcal{W} judgment, $\Gamma \vdash e \Rightarrow (\tau; \theta)$, defined in class in order to handle the new features of the language (in other words, all cases of the `exp` datatype except `Var`, `App`, and `Let`).

Note: Your new rules should only mention monotypes τ , not polytypes σ . Polytypes only show up explicitly in the rules for `Var` and `Let`, which are unchanged from the ones given in class.

2 Implementing Algorithm \mathcal{W}

Implement your extended version of the Algorithm \mathcal{W} judgment in Standard ML. You can implement it however you like—your solution need not be particularly efficient. All I require in order to test your code is that you provide a function

```
val typecheck : exp -> ptyp option
```

with the behavior that `typecheck e = SOME(σ)` iff e is well-typed in the empty context and σ is the principal polytype of e , and `typecheck e = NONE` iff e is not well-typed in the empty context.