CS51 FINAL PROJECT

Dante Lacuadra

May 2020

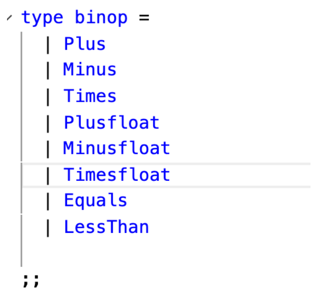
1 Extension

1.1 Introduction

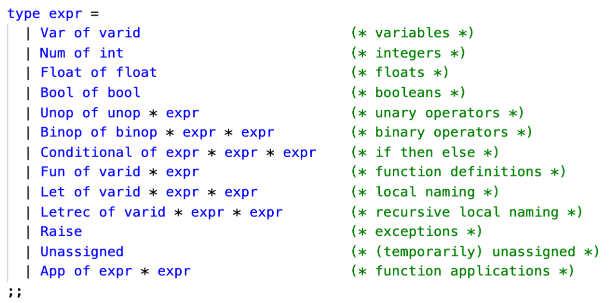
While I will agree that my writeup is particularly bad. I first want to say I put a lot of time into completing expr.ml, and evaluation.ml. I pretty didn't have time to do most of the extension. I spent over 20 hours doing this final project at least. Given the circumstances I hope you guys understand, I'm really did like this class, but I have issues going on at home so yeah.

1.2 Addition

My addition was to allow for float operators. First what I had to do was edit my definition of expressions and binary operators in the expr.ml and corresponding mli file.

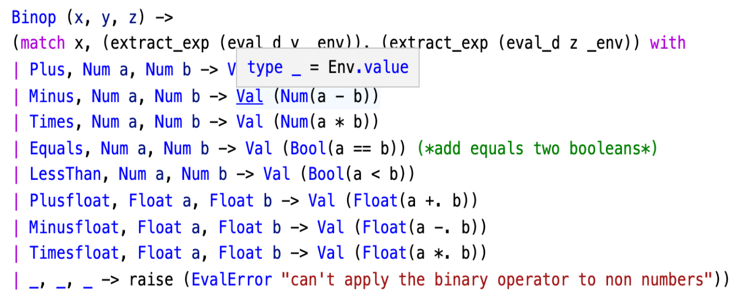


```
type binop =
  | Plus
  | Minus
  | Times
  | Plusfloat
  | Minusfloat
  | Timesfloat
  | Equals
  | LessThan

;;
```

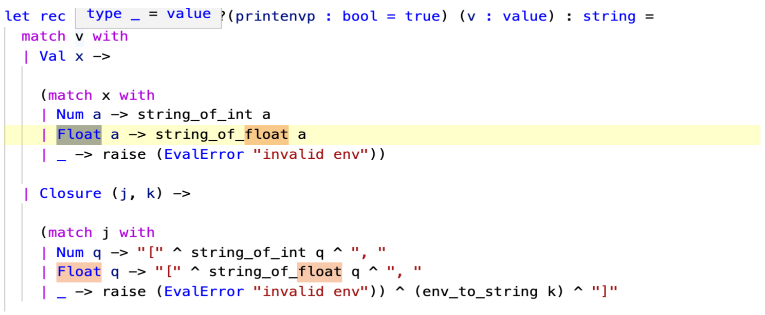


```
type expr =
  | Var of varid                         (* variables *)
  | Num of int                           (* integers *)
  | Float of float                       (* floats *)
  | Bool of bool                         (* booleans *)
  | Unop of unop * expr                  (* unary operators *)
  | Binop of binop * expr * expr         (* binary operators *)
  | Conditional of expr * expr * expr    (* if then else *)
  | Fun of varid * expr                  (* function definitions *)
  | Let of varid * expr * expr           (* local naming *)
  | Letrec of varid * expr * expr        (* recursive local naming *)
  | Raise                                (* exceptions *)
  | Unassigned                           (* (temporarily) unassigned *)
  | App of expr * expr                   (* function applications *)
;;
```

Then what I had to do is edit the Binop match statement in evaluation.ml and change some of the definitions to include floats in the Env module.

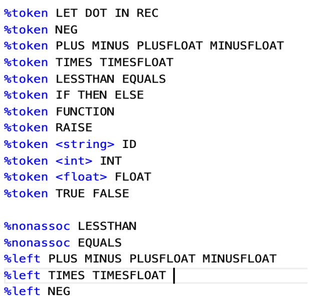


```
Binop (x, y, z) ->
(match x, (extract_exp (eval_d y _env)), (extract_exp (eval_d z _env)) with
| Plus, Num a, Num b -> V type _ = Env.value
| Minus, Num a, Num b -> Val (Num(a - b))
| Times, Num a, Num b -> Val (Num(a * b))
| Equals, Num a, Num b -> Val (Bool(a == b)) (*add equals two booleans*)
| LessThan, Num a, Num b -> Val (Bool(a < b))
| Plusfloat, Float a, Float b -> Val (Float(a +. b))
| Minusfloat, Float a, Float b -> Val (Float(a -. b))
| Timesfloat, Float a, Float b -> Val (Float(a *. b))
| _, _, _ -> raise (EvalError "can't apply the binary operator to non numbers"))
```

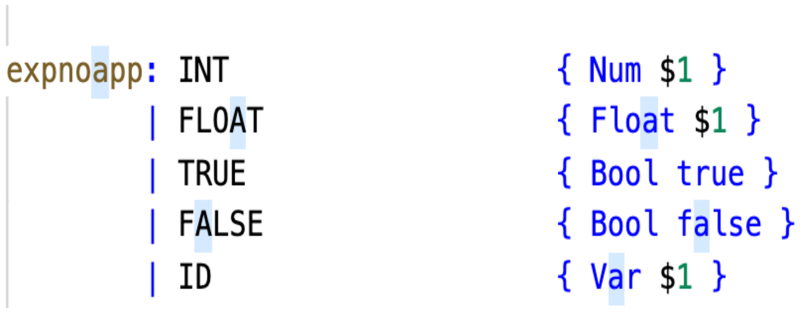


```
let rec  type _ = value ?(printenvp : bool = true) (v : value) : string =
  match v with
  | Val x ->

    (match x with
    | Num a -> string_of_int a
    | Float a -> string_of_float a
    | _ -> raise (EvalError "invalid env"))

  | Closure (j, k) ->

    (match j with
    | Num q -> "[" ^ string_of_int q ^ ", "
    | Float q -> "[" ^ string_of_float q ^ ", "
    | _ -> raise (EvalError "invalid env")) ^ (env_to_string k) ^ "]"
```

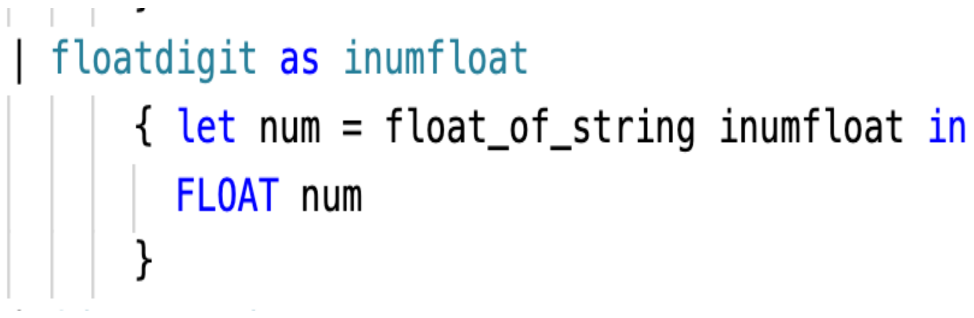
Then I had to make many changes to both miniml_lex.mll and miniml_parse.mly. In I've attached what I've changed. Basically I had to add a new token that included floats, I also had to add new expressions to expnoapp, amending the hash table in the former file and telling the parser what floats actually are.



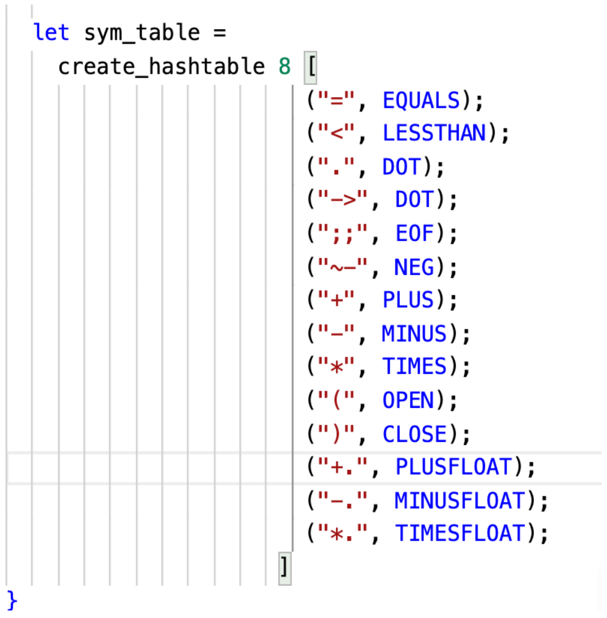
```
%token LET DOT IN REC
%token NEG
%token PLUS MINUS PLUSFLOAT MINUSFLOAT
%token TIMES TIMESFLOAT
%token LESSTHAN EQUALS
%token IF THEN ELSE
%token FUNCTION
%token RAISE
%token <string> ID
%token <int> INT
%token <float> FLOAT
%token TRUE FALSE

%nonassoc LESSTHAN
%nonassoc EQUALS
%left PLUS MINUS PLUSFLOAT MINUSFLOAT
%left TIMES TIMESFLOAT
%left NEG
```



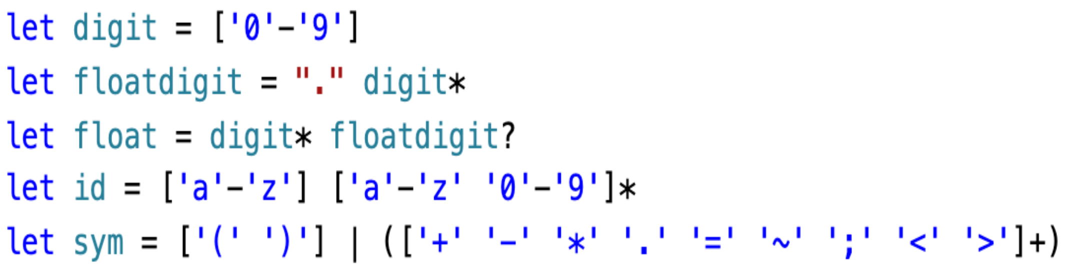
```
expnoapp: INT                   { Num $1 }
        | FLOAT                 { Float $1 }
        | TRUE                  { Bool true }
        | FALSE                 { Bool false }
        | ID                    { Var $1 }
```



```
| floatdigit as inumfloat
    { let num = float_of_string inumfloat in
      FLOAT num
    }
```



```
  let sym_table =
    create_hashtable 8 [
                         ("=", EQUALS);
                         ("<", LESSTHAN);
                         (".", DOT);
                         ("->", DOT);
                         (";;", EOF);
                         ("~-", NEG);
                         ("+", PLUS);
                         ("-", MINUS);
                         ("*", TIMES);
                         ("(", OPEN);
                         (")", CLOSE);
                         ("+.", PLUSFLOAT);
                         ("-.", MINUSFLOAT);
                         ("*.", TIMESFLOAT);
                        ]
}
```



```
let digit = ['0'-'9']
let floatdigit = "." digit*
let float = digit* floatdigit?
let id = ['a'-'z'] ['a'-'z' '0'-'9']*
let sym = ['(' ')'] | (['+' '-' '*' '.' '=' '~' ';' '<' '>']+)
```

Thank you for reading!