University of Massachusetts Lowell — Comp 3010: Organization of Programming Languages

Assignment 8

DUE: Tuesday, Dec 3, 2024, 11:59PM

Collaboration Policy: You may collaborate with others but must submit work that is your own. Submissions of collaborators should be different and each submission must demonstrate your own work. Please refer to collaboration policy on the course page in Blackboard.

Submission instructions: Please submit your homework via GradeScope. Some points to note:

- You need to use LaTeX for the non-coding parts of the assignment. If you produce your assignment using a different tool, please make sure your assignment is legible and follows these submission instructions. Scanned copies of hand-writen assignments are not acceptible.
- If you collaborate with your classmates, please note who they are when you submit your answers.
- We will provide (through Blackboard) a LaTeX template you can use to write the answers to your assignment.
- **Don't forget to submit both your PDF and check.ml**

1 Type Inference (30 points)

In this question you will figure out how to extend the constraint-based type inference judgment we saw in class to additional language constructs. Then, in Question 2, you will implement type inference for this language.

Consider a typed λ-calculus with pairs, integers, booleans (including a test for integer equality), let-definitions, and recursive functions.

$$
\begin{aligned}
\tau &::= X \mid \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2 \mid \textsf{int} \mid \textsf{bool} \\
e &::= x \mid \lambda x{:}\tau.\, e \mid e_1\ e_2 \\
&\mid (e_1, e_2) \mid \#1\ e \mid \#2\ e \\
&\mid n \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2 \\
&\mid \textsf{true} \mid \textsf{false} \mid e_1 = e_2 \mid \textsf{if}\ e_1\ \textsf{then}\ e_2\ \textsf{else}\ e_3 \\
&\mid \textsf{let}\ x = e_1\ \textsf{in}\ e_2
\end{aligned}
$$

Recall the constraint-based type inference judgment has the form

$$\Gamma \vdash e : \tau \triangleright C$$

where Γ is a typing context, e is an expression, τ is a type, and C is a set of constraints, meaning that if substitution σ unifies constraint set C, then expression e is well typed, with type $\sigma(\tau)$.

In the following question, you will provide inference rules for constraint-based type-inference. Use constraint-based type-inference rules from Lecture 10 (slides 10-12).

(a) Provide inference rules for the constraint-based type-inference judgment for pairs and projection (i.e., for expressions (e_1, e_2) and $\#1\ e$ and $\#2\ e$).

(b) Provide inference rules for the constraint-based type-inference judgment for conditionals and the integer equality expression (i.e., for expressions $\textsf{if}\ e_1\ \textsf{then}\ e_2\ \textsf{else}\ e_3$ and $e_1 = e_2$). Note that the integer equality expression $e_1 = e_2$ is defined only when expressions e_1 and e_2 evaluate to integers.

(c) Provide inference rules for the constraint-based type-inference judgment for let i.e., for expression $\textsf{let}\ x = e_1\ \textsf{in}\ e_2$.

2 Implementing Type Inference (70 points)

Now you get to implement type inference for the language in Question 1! Please *carefully* read the following instructions and follow them.

- Download the file `hw8.zip` from the assignment page on Blackboard, and expand it. You will now have a directory `lam`. This directory contains OCaml code for a partial implementation of type inference for a simple lambda calculus.
- Read the `README` file first and look through the files to familiarize yourself with the code.
- Make the executable. (Run `make`. Also, read the `README`.)
- Implement the functions `unify` and `check` in the file `check.ml`, where it says `(* FILL IN HERE FOR QUESTION 2 *)`. **Modify only the file `check.ml`!** Do not modify any other ML file.

  The function `check` implements the constraint-based type-inference judgment $\Gamma \vdash e : \tau \triangleright C$ that you defined in Question 1 above. The function `unify` implements the unification algorithm.
- Don't forget to upload your completed version of `check.ml` when you submit your assignment.

HINTS.

(a) First understand what the program is doing, and how it is calling the functions you are implementing. The code in the file `main.ml` calls the functions you are implementing.

(b) We have provided several helper functions in `helper.ml` which you will need to implement the functions. Look at this file, and understand what each of the functions do.

- To generate fresh type variables, use the `next_tvar` function, defined in `helper.ml`.
- To calculate the set of free type variables for a type, use the `ftvs` function, defined in `helper.ml`.

(c) Constraints are represented using the OCaml type `constr` (defined in `ast.ml` to be pairs of types, i.e., `(typ * typ)`). Values of type `constr` can be manipulated using the `Constr` module (defined in `ast.ml`) which contains all of the functions listed in the OCaml module `Set.S` where `elt` is `(typ * typ)` and `t` is `constr`.

(d) Type substitutions are represented using the OCaml type `subst` (defined in `ast.ml`). Values of type `subst` can be manipulated using the `VarMap` module (also defined in `ast.ml`) which contains all of the functions listed in the OCaml module `Map.S` where `key` is `var` and `'a` is instantiated to `typ`.

(e) To represent sets of variables, use the OCaml type `varset`. Values of type `varset` can be manipulated using the `VarSet` module (defined in, you guessed it, `ast.ml`) which contains all of the functions listed in the OCaml module `Set.S` where `elt` is `var` and `t` is `varset`.

(f) Contexts map program variables to types and are represented by the OCaml type `context`.

(g) We have provided a number of example programs together with the corresponding output. If you implement the functions correctly your output should be the same as the `.out` files, up to renaming of type variables, and symmetry of constraints (e.g., **int** $= X$ instead of $X =$ **int**). Make sure to check the intermediate output as well as the final answers. (HINT: the grading will consist of running similar programs and comparing the output against a gold standard.)

(h) Note that the code we are providing you with in this assignment uses OCaml's standard library. Look online for descriptions of OCaml's standard library, e.g., via `https://caml.inria.fr/pub/docs/manual-ocaml/libref/`.