HW7 – Type Inference

CS 476, Fall 2018
Due Nov. 9 at 2 PM

1 Instructions

Begin by downloading the file `hw7-base.ml` from the course website and renaming it to `hw7.ml`. This file contains the functions that you will use and modify in the homework. You will need to add cases to `gather_constraints`, but you should not need to modify any of the other predefined functions. Submit your completed `hw7.ml` via Gradescope. As always, please don't hesitate to ask for help on Piazza (`https://piazza.com/class/jkh8q52qrh06v`).

2 Type Inference for a Functional Language

The file `hw7-base.ml` defines a type `exp` of expressions for a simple OCaml-like language. It also defines a function `get_constraints` that performs (the first half of) type inference for this language: `get_constraints gamma e` returns either `None`, if `e` fails to typecheck, or `Some (t, c)`, where `t` is a type with type variables, and `c` is a list of constraints (pairs of types with variables) that must be satisfied for `t` to be a correct type for `e`. The following problems will ask you to extend the `get_constraints` function to handle the rest of the language.

You can test your code using the `infer_type` function, which takes an `exp` and infers its type using your `get_constraints` function and a pre-written `unify` function. The inferred type may have variables in it, if there is more than one possible type for the expression: for instance, `infer_type (Fun ("x", Var "x"))` might return something like `Some (Tarrow (Tvar 3, Tvar 3))`, indicating that `fun x -> x` could have type `a -> a` for any type `a`.

3 Problems

1. (2 points) Extend the provided `get_constraints` function to handle integer and boolean literals, according to the following rules:

$$\frac{(i \text{ is an integer})}{\Gamma \vdash i : \text{int} \mid \{\}} \qquad \frac{(b \text{ is a boolean})}{\Gamma \vdash b : \text{bool} \mid \{\}}$$

Once you have completed this problem, `infer_type (Int 3)` should return `Some Tint`.

2. (5 points) Extend the provided `get_constraints` function to handle addition and comparison, according to the following rules:

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid C_1 \qquad \Gamma \vdash e_2 : \tau_2 \mid C_2}{\Gamma \vdash e_1 + e_2 : \text{int} \mid \{\tau_1 = \text{int}, \tau_2 = \text{int}\} \cup C_1 \cup C_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid C_1 \qquad \Gamma \vdash e_2 : \tau_2 \mid C_2}{\Gamma \vdash e_1 = e_2 : \text{bool} \mid \{\tau_1 = \tau_2\} \cup C_1 \cup C_2}$$

Once you have completed this problem, `infer_type (Fun ("x", Fun ("y", Add (Var "x", Var "y"))))` should return `Some (Tarrow (Tint, Tarrow (Tint, Tint)))`.

3. (5 points) Extend the provided `get_constraints` function to handle if-then-else expressions, according to the following rule:

$$\frac{\Gamma \vdash e : \tau \mid C \qquad \Gamma \vdash e_1 : \tau_1 \mid C_1 \qquad \Gamma \vdash e_2 : \tau_2 \mid C_2}{\Gamma \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 : \tau_1 \mid \{\tau = \text{bool}, \tau_1 = \tau_2\} \cup C \cup C_1 \cup C_2}$$

Once you have completed this problem, `infer_type (Fun ("x", Fun ("y", If (Eq (Var "x", Var "y"), Var "x", Int 3))))` should return `Some (Tarrow (Tint, Tarrow (Tint, Tint)))`.

4. (6 points) Extend the provided `get_constraints` function to handle expressions involving tuples, according to the following rules:

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid C_1 \qquad \Gamma \vdash e_2 : \tau_2 \mid C_2}{\Gamma \vdash (e_1, e_2) : \tau_1 * \tau_2 \mid C_1 \cup C_2}$$

$$\frac{\Gamma \vdash e : \tau \mid C \qquad \tau_1, \tau_2 \text{ fresh}}{\Gamma \vdash \text{fst } e : \tau_1 \mid \{\tau = \tau_1 * \tau_2\} \cup C} \qquad \frac{\Gamma \vdash e : \tau \mid C \qquad \tau_1, \tau_2 \text{ fresh}}{\Gamma \vdash \text{snd } e : \tau_2 \mid \{\tau = \tau_1 * \tau_2\} \cup C}$$

You can use the function `fresh_tident : unit -> tident` to create a new fresh type variable. Once you have completed this problem, `infer_type (Fun ("x", Add (Fst (Var "x"), Snd (Var "x"))))` should return `Some (Tarrow (Ttuple (Tint, Tint), Tint))`.

5. (7 points) Extend the provided `get_constraints` function to handle expressions involving sum types and pattern matching, according to the following rules:

$$\frac{\Gamma \vdash e : \tau \mid C \qquad \tau_2 \text{ fresh}}{\Gamma \vdash \text{inl } e : \tau + \tau_2 \mid C} \qquad \frac{\Gamma \vdash e : \tau \mid C \qquad \tau_1 \text{ fresh}}{\Gamma \vdash \text{inr } e : \tau_1 + \tau \mid C}$$

$$\frac{\Gamma \vdash e : \tau \mid C \qquad \Gamma[x_1 \mapsto \tau_l] \vdash e_1 : \tau_1 \mid C_1 \qquad \Gamma[x_2 \mapsto \tau_r] \vdash e_2 : \tau_2 \mid C_2 \qquad \tau_l, \tau_r \text{ fresh}}{\Gamma \vdash (\text{match } e \text{ with inl } x_1 \to e_1 \mid \text{inr } x_2 \to e_2) : \tau_1 \mid \{\tau = \tau_l + \tau_r, \tau_1 = \tau_2\} \cup C \cup C_1 \cup C_2}$$

Once you have completed this problem, `infer_type (Fun ("x", Match (Var "x", "a", Eq (Var "a", Int 3), "b", Var "b")))` should return `Some (Tarrow (Tsum (Tint, Tbool), Tbool))`.