MicroCaml Formal Type Inference Rules

November 11, 2025

1 Introduction

This document presents the type inference rules for MicroCaml.

2 Preliminaries

2.1. Context. Type inference rules define judgments that make use of a context Γ (Stylized as G). A context is a mapping between variable names and their type. $G(x)$ means to look up the value of x maps to in the context G. This operation is undefined if there is no mapping for x in G. The second operation is written $G[x : t]$. It defines a new context that is the same as G but maps x to t. It thus overrides any prior mapping for x in G.

2.2. Constraints. In addition to the context, we also will need to keep track of the constraints we should use when we try and infer types of variables.
A constraint set is a set of bindings of from one type to another. We will use the syntax $t_1 : t_2$ to indicate that type t_1 should be the same as type t_2. To keep types consistent, we may use a placeholder type to refer to an expression. You can see more in the below constraints rules. These placeholder types will look like t or t_n. If two types use the same n value, then they are considered the same type. Type inference fails if the final constraint set has any contradictions.

2.3. Make. Whenever you see $\texttt{make}(t_x)$, this means you should create a fresh polymorphic type (recall polymorphic = $'a$, $'b$, etc.) which will fill in for value t_x.

2.4. Syntax. In this document, we have simplified the presentation of the syntax, so it may not correspond exactly to the files that your interpreter will read in. Hopefully, the connection between what we show here and the AST is clear enough from context.

2.5. Error conditions. The semantics here define only *correct* evaluations. It says nothing about what happens when, say, you have a type error. For example, for the rules below, there is no type t for which you can prove the judgment $\Gamma \vdash 1 + \textbf{true} : t$. In your actual implementation, erroneous programs will cause an exception to be raised, as indicated in the project README.

3 Type Inference Rules

3.1. Expression Axioms.

$$\text{var-lookup}\frac{G(x) = t}{G \vdash x : t \dashv \{\}}$$

$$\text{int}\frac{}{G \vdash n : \mathit{int} \dashv \{\}} \qquad \text{bool}\frac{}{G \vdash b : \mathit{bool} \dashv \{\}}$$

3.2. Non-Axiom Expressions.

$$\text{op-add-sub-div-mult}\frac{G \vdash e_1 : t_1 \dashv C_1 \quad G \vdash e_2 : t_2 \dashv C_2}{G \vdash e_1 \ \texttt{op}\ e_2 : \mathit{int} \dashv \{t_1 : t_2, t_1 : \mathit{int}\} \cup C_1 \cup C_2}$$

$$\text{op-and-or}\frac{G \vdash e_1 : t_1 \dashv C_1 \quad G \vdash e_2 : t_2 \dashv C_2}{G \vdash e_1 \ \texttt{op}\ e_2 : \mathit{bool} \dashv \{t_1 : t_2, t_1 : \mathit{bool}\} \cup C_1 \cup C_2}$$

$$\text{op-comparison}\frac{G \vdash e_1 : t_1 \dashv C_1 \quad G \vdash e_2 : t_2 \dashv C_2}{G \vdash e_1 \ \texttt{op}\ e_2 : \mathit{bool} \dashv \{t_1 : t_2\} \cup C_1 \cup C_2} \qquad \text{not}\frac{G \vdash e : t \dashv C}{G \vdash \texttt{not}\ e : \mathit{bool} \dashv \{t : \mathit{bool}\} \cup C}$$

$$\text{if}\frac{G \vdash e_1 : t_1 \dashv C_1 \quad G \vdash e_2 : t_2 \dashv C_2 \quad G \vdash e_3 : t_3 \dashv C_3}{G \vdash \texttt{if}\ e_1\ e_2\ e_3 : t_2 \dashv \{t_1 : \mathit{bool}, t_2 : t_3\} \cup C_1 \cup C_2 \cup C_3}$$

$$\text{fun}\frac{\texttt{make}(t_x) \quad G[x : t_x] \vdash e : t \dashv C}{G \vdash \texttt{fun}\ x \to e : (t_x \to t) \dashv C} \qquad \text{app}\frac{\texttt{make}(t_x) \quad G \vdash e_1 : t_1 \dashv C_1 \quad G \vdash e_2 : t_2 \dashv C_2}{G \vdash e_1\ e_2 : t_x \dashv \{t_1 : (t_2 \to t_x)\} \cup C_1 \cup C_2}$$

$$\text{record}\frac{\forall i \in [1, n],\ G \vdash e_i : t_i \dashv C_i}{G \vdash \{l_1 : e_1, \ldots, l_n : e_n\} : \{l_1 : t_1, \ldots, l_n : t_n\} \dashv C_1 \cup \ldots \cup C_n}$$

$$\text{select}\frac{G \vdash e : \{l_1 : t_1, \ldots, l_n : t_n\} \dashv C \quad \exists i \in \{1, \ldots, n\}\ \texttt{s.t.}\ l_i = l_x}{G \vdash e.l_x : t_i \dashv C}$$

3.3. Let Bindings.

$$\text{let-no-rec}\frac{G \vdash e_1 : t_1 \dashv C_1 \quad G[x : t_1] \vdash e_2 : t_2 \dashv C_2}{G \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : t_2 \dashv C_1 \cup C_2}$$

$$\text{let-yes-rec}\frac{\texttt{make}(t_x) \quad G[x : t_x] \vdash e_1 : t_1 \dashv C_1 \quad G[x : t_1] \vdash e_2 : t_2 \dashv C_2}{G \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : t_2 \dashv C_1 \cup C_2}$$