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

Assignment 8

Name: Harishwar Reddy Erri

UML ID: 02148304

Collaborators: None

Make sure that the remaining pages of this assignment do not contain any identifying information.

1 Type Inference (30 points)

(a) Inference Rules for Pairs and Projections

Pair Construction: The pair expression (e_1, e_2) combines two sub-expressions into a pair. The resulting type is a product type $\tau_1 \times \tau_2$, where τ_1 is the type of e_1 and τ_2 is the type of e_2.

$$\frac{\Gamma \vdash e_1 : \tau_1 \ \blacktriangleright\ C_1 \quad \Gamma \vdash e_2 : \tau_2 \ \blacktriangleright\ C_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2 \ \blacktriangleright\ C_1 \cup C_2}$$

Explanation:

- Each component e_1 and e_2 is evaluated in the current typing context Γ.
- The resulting type of the pair is $\tau_1 \times \tau_2$, with the constraints C_1 and C_2 merged.

Projections: Projections extract components from a pair. The type of the expression e must be a product type $\tau_1 \times \tau_2$.

First Projection ($\#1e$):

$$\frac{\Gamma \vdash e : X \ \blacktriangleright\ C \quad C' = \{X \equiv \tau_1 \times \tau_2\}}{\Gamma \vdash \#1e : \tau_1 \ \blacktriangleright\ C \cup C'}$$

Second Projection ($\#2e$):

$$\frac{\Gamma \vdash e : X \ \blacktriangleright\ C \quad C' = \{X \equiv \tau_1 \times \tau_2\}}{\Gamma \vdash \#2e : \tau_2 \ \blacktriangleright\ C \cup C'}$$

Explanation:

- The projection $\#1e$ extracts the first component, ensuring the type X of e unifies with $\tau_1 \times \tau_2$.
- Similarly, $\#2e$ extracts the second component under the same unification condition.

(b) Inference Rules for Conditionals and Integer Equality

Conditionals (if e_1 then e_2 else e_3):

Conditionals require:

- e_1 evaluates to `bool`.
- e_2 and e_3 evaluate to the same type.

The rule is:

$$\frac{\Gamma \vdash e_1 : \tau_1 \blacktriangleright C_1 \quad \Gamma \vdash e_2 : \tau_2 \blacktriangleright C_2 \quad \Gamma \vdash e_3 : \tau_3 \blacktriangleright C_3 \quad C' = \{\tau_1 \equiv \texttt{bool}, \tau_2 \equiv \tau_3\}}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : \tau_2 \blacktriangleright C_1 \cup C_2 \cup C_3 \cup C'}$$

Explanation:

- The guard e_1 must evaluate to `bool`, enforced by $\tau_1 \equiv \texttt{bool}$.
- The branches e_2 and e_3 must have the same type, enforced by $\tau_2 \equiv \tau_3$.
- Constraints C_1, C_2, C_3 from all components are combined.

Integer Equality ($e_1 = e_2$): Equality requires both operands to be integers, and the result is `bool`:

$$\frac{\Gamma \vdash e_1 : \tau_1 \blacktriangleright C_1 \quad \Gamma \vdash e_2 : \tau_2 \blacktriangleright C_2 \quad C' = \{\tau_1 \equiv \texttt{int}, \tau_2 \equiv \texttt{int}\}}{\Gamma \vdash e_1 = e_2 : \texttt{bool} \blacktriangleright C_1 \cup C_2 \cup C'}$$

Explanation:

- Both operands e_1 and e_2 must evaluate to `int`, enforced by $\tau_1 \equiv \texttt{int}$ and $\tau_2 \equiv \texttt{int}$.
- The result type is `bool`.

(c) Inference Rules for `let` Expressions

The expression `let` $x = e_1$ `in` e_2 assigns e_1 to x and evaluates e_2 in this context. The rule is:

$$\frac{\Gamma \vdash e_1 : \tau_1 \blacktriangleright C_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2 \blacktriangleright C_2}{\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 : \tau_2 \blacktriangleright C_1 \cup C_2}$$

Explanation:

- The type τ_1 of e_1 is added to the context for x.
- The type of the entire `let` expression is the type τ_2 of e_2.
- The constraints C_1 and C_2 are combined.