A Theory of Type Polymorphism in Programming

Robin Milner

June 20, 2025

1 Illustrations of the Type Discipline

The constructs

$$\begin{aligned} &\textit{let } x = e \textit{ in } e' \\ &\textit{let } f(x_1, \ldots, x_n) = e \textit{ in } e' \end{aligned}$$

The fully determined types of ML are built from a set of basic types (*int*, *bool*, etc) by the binary indexed operators $\times$, $+$ (disjoint sum) and $\rightarrow$, and the unary postfixed operator *list*. Polymorphic types (polytypes) are obtained by admitting **type variables**, which here are represented by $\alpha, \beta, \gamma, \ldots$. We represent arbitrary types by ρ, σ, τ.

Example 1.1. Mapping a function over a list

$$\begin{aligned} \textit{letrec } (f, m) = \textit{if } &\mathrm{null}(m) \textit{ then } \mathrm{nil} \\ &\textit{else } \mathrm{cons}(f(hd(m)), \mathrm{map}(f, tl(m))) \end{aligned}$$

2 Problems

3 References