School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213–3891
Electronic mail: `rwh@cs.cmu.edu`

July 27, 1995

Information Processing Letters
Elsevier Science B.V.
Molenwerf 1
Postbus 211
1000 AE Amsterdam
NETHERLANDS

To the editor:

In the paper "A Simplified Account of Polymorphic References" (IPL v.51 pp.201–206) a proof of the soundness of type inference for a functional language combining polymorphism and references is presented. The main result is stated as follows:

> **Theorem** If $\mu \vdash e \Rightarrow v, \mu'$, $\lambda \vdash e : \tau$, $\mu : \lambda$, and λ is imperative, then there exists $\lambda' \supseteq \lambda$ such that $\mu' : \lambda'$, and $\lambda' \vdash v : \tau$.

The theorem establishes a type preservation property for evaluation that ensures that the result of a program may be ascribed the same type as the program itself. (A similar result was obtained by Tofte [3] using rather different techniques.)

The sense in which this theorem establishes soundness merits further clarification. This may be achieved by extending the evaluation relation with transitions of the form $\mu \vdash e \Rightarrow \mathsf{wrong}$, where wrong is a distinguished ill-typed token representing a run-time error. For example, the following rule expresses that it is an error to attempt to apply a value other than a functional abstraction:

$$\frac{\mu \vdash e \Rightarrow v, \mu'}{\mu \vdash e\, e_1 \Rightarrow \mathsf{wrong}} \quad (v \neq \lambda x.e') \qquad \text{(APP-WRONG)}$$

The proof of the theorem may be extended to cover these additional rules, with the consequence that the final value of a well-typed expression cannot be wrong since by design wrong is ill-typed. The extension of the proof to account for wrong transitions relies on a *canonical forms* lemma [1] characterizing the shapes of closed values of a type. In particular if v is a closed value of functional type, then v must be a

λ-abstraction. Consequently, the rule APP-WRONG cannot apply if the expression $e\,e_1$ is well-typed.

Since the proof of impossibility of wrong transitions is routine, an explicit treatment of them was omitted from the Tofte's and my own work. An alternative approach, advocated by Felleisen and Wright [4], is to work with single-step operational semantics. In this case the canonical forms lemma is used to establish that a well-typed program is either a value or can make progress by a single-step transition. By taking the informal notion "go wrong" to mean "unable to make progress", it follows that well-typed programs do not go wrong. This approach has the advantage that explicit transitions to wrong are not needed, but at the expense of requiring a separate progress lemma.

Sincerely,

Robert Harper
Associate Professor

1. Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.
2. Robin Milner. "A Theory of Type Polymorphism in Programming Languages". *Journal of Computer and System Sciences*, vol. 17, pp. 348–375, 1978.
3. Mads Tofte. "Type Inference for Polymorphic References". *Information and Computation*, vol. 89, pp. 1–34, November, 1990.
4. Andrew Wright and Matthias Felleisen. "A Syntactic Approach to Type Soundness". *Information and Computation*, vol. 115, no. 1, pp. 38–94, November, 1994.