Note on PCF Computable Functions

Peter Thiemann

01.06.2015

This note sketches a proof for the following statement:

> There is a PCF term that computes a total function that cannot be programmed in System T.

- Any (well-typed) System T program (i.e., a closed expression of type $\texttt{nat} \to \texttt{nat}$) can be encoded as a number. Some variation of Gödel numbering will work, so that there is a surjective mapping G from natural numbers to closed expressions of that type.
- There are PCF-computable functions that, given an encoding number of a System T expression, return the kind of the expression (variable, lambda, application, iterator, ...) and encoding numbers for the subexpressions.
- Given these functions, we can write an interpreter for System T in PCF. This interpreter is a PCF term for a function $I\ e\ n$ that computes the result of applying the System T expression $G(e)$ to input $n : \texttt{nat}$.
- The PCF term for I is terminating (because it is a System T interpreter and all System T programs terminate).
- Now we can construct a PCF term for a function $J : \texttt{nat} \to \texttt{nat}$ defined by
$$J(n) = I\ n\ n + 1$$
- The function J is terminating because it just invokes I, which is terminating.
- The function J cannot have a System T program. To derive a contradiction, suppose that J is computed by System T program $G(e)$, for some e. Then $J(e) = I\ e\ e + 1$ by definition of J. However, I is an interpreter so that $J = I\ e$ and $J(e) = I\ e\ e$. Contradiction to J's definition!