

# A “Book” proof that parallel convergence tester cannot implement parallel or

Prakash Panangaden

September 15, 2009

## Abstract

I give a short and elementary proof that parallel convergence tester cannot implement parallel or.

Parallel convergence tester is a two-argument function  $c : \mathbb{O} \rightarrow \mathbb{O} \times \mathbb{O}$  with the following graph:

$$\begin{array}{ll} c(\perp, \perp) = \perp & c(\perp, \top) = \top \\ c(\top, \perp) = \top & c(\top, \top) = \top. \end{array}$$

Parallel or is the well-known function  $p : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B}$  defined on the domain of booleans  $\mathbb{B}$  with the following graph:

$$\begin{array}{ll} p(ff, ff) = ff & p(\perp, tt) = tt \\ p(tt, \perp) = tt & p(ff, \perp) = \perp \\ p(\perp, ff) = \perp & \end{array}$$

with all other values being determined by monotonicity. These functions arise in the discussion of full abstraction of PCF [Plo77] and the lazy  $\lambda$ -calculus [AO93]. It is now well-known that the lattice of degrees of parallelism is very rich and infinite in two directions [Buc97, PP01]: the fact that one cannot implement  $p$  with  $c$  is a tiny part of these results.

There is, however, a very simple proof that PCF with  $c$  cannot implement  $p$  assuming that PCF by itself cannot implement  $p$ . Suppose that such an implementation exists so that there is some pure PCF context  $C[\cdot]$  with  $C[c] = p$ . The functional  $\lambda x.C[x]$  is monotone. Therefore the pure PCF

term  $C[\lambda u. \top]$  is extensionally above  $p$ , but  $p$  is maximal so the pure PCF term  $C[\lambda u. \top] = p$ , a contradiction.

In fact this argument applies to any function with return type  $\mathbb{O}$  even if it is horribly non-recursive.

## References

- [AO93] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus: I, ii. *Information and Computation*, 105:159–267, 1993.
- [Buc97] Antonio Bucciarelli. Degrees of parallelism in the continuous type hierarchy. *Theor. Comput. Sci.*, 177(1):59–71, 1997.
- [Plo77] G. D. Plotkin. Lcf considered a programming language. *Theoretical Computer Science*, 5(3):223–256, 1977.
- [PP01] Riccardo Pucella and Prakash Panangaden. On the expressive power of first-order boolean functions in pcf. *Theoretical Computer Science*, 266:543–567, 2001.