Abstract

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

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

$c(\bot, \bot) = \bot$
$c(\bot, \top) = \top$
$c(\top, \bot) = \top$
$c(\top, \top) = \top$.

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

$p(\text{ff}, \text{ff}) = \text{ff}$
$p(\text{tt}, \bot) = \text{tt}$
$p(\bot, \text{ff}) = \bot$
$p(\bot, \text{tt}) = \text{tt}$
$p(\text{ff}, \text{tt}) = \bot$

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 $\bot$ even if it is horribly non-recursive. I came up with this proof in 1988 while having a shower.

References


