1

Processes, Types, and Observations

Benjamin C. Pierce
Computer Laboratory, University of Cambridge
http://www.cl.cam.ac.uk/users/bcp1000

The economy and flexibility of the process calculi such as the pi-calculus make them attractive both as an object of theoretical study and as a basis for language design and implementation, in particular for concurrent object-oriented languages. However, such generality has a cost: these systems are rather low level, with correspondingly low-level reasoning techniques.

Static type systems for process calculi can be used to recover useful information about program structure, giving rise to more powerful (and realistic) principles of process equivalence. Three examples are discussed:

- Refining channel types to separate the capabilities of sending and receiving on a given channel [2] allows the type system to track the directionality of information flow in a process system and gives rise to an elegant notion of *subtyping*. Moreover, if the "observer" of a pair of processes in the definition of bisimulation is restricted to making only well-typed tests, we obtain a coarser equivalence than usual. We can use this equivalence to justify, for example, the validity of a simple translation of the lambda-calculus into the pi-calculus, which fails if we use an untyped definition of equivalence.
- *Linear* channel types [1] can be used to guarantee that certain communications never interfere with other activity in a system of processes, giving rise to a useful proof technique for bisimulation based on "partial confluence." Moreover, the typed variant of bisimulation now disallows observers that make "nonlinear tests" over linear channels, justifying useful program transformations such as tail-call optimizations.
- Process calculi also be enriched with *polymorphic* type systems supporting information hiding through type abstraction [3, 4]. Work is in progress on corresponding notions of process equivalence.

References

[1] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the pi-calculus. *Principles of Programming Languages*, 1996.

[2] Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. In *Logic in Computer Science*, 1993. Full version to appear in *Mathematical Structures in Computer Science*, 1996.

[3] Benjamin C. Pierce. Programming in the pi-calculus: An experiment in programming language design. Tutorial notes on the Pict language. Available electronically, 1995.

[4] David N. Turner. *The Polymorphic Pi-calulus: Theory and Implementation.* PhD thesis, University of Edinburgh, 1996.

E. Najm et al. (eds.), *Formal Methods for Open Object-based Distributed Systems*