Types for concurrency

Aaron Turon

April 26, 2010

Typing and subtyping for mobile processes

@inproceedings{
    author = {Pierce, Benjamin and Sangiorgi, Davide},
    title = {Typing and subtyping for mobile processes},
    booktitle = {LICS},
    year = {1993},
    url = {http://www.cis.upenn.edu/~bcpierce/papers/pi.ps},
}

Pierce and Sangiorgi formalize and generalize the sort system Milner originally proposed for the $\pi$-calculus. Their type system gives types to channels, including both usage information (send, receive, both) and structural information (how many channels, of which types, will be sent or received). Usage information gives rise to a simple form of subtyping. Recursive channel types are allowed, so ultimately the type system must decide recursive subtyping, and the authors give an algorithm for doing so. Finally, the authors explore the ramifications of their type system on process equivalence.

Significance: This paper firmly establishes a basic framework for building type systems for the $\pi$-calculus: channels, not processes, are typed, usage information on those channels is included in the types, and recursion at the type level is fundamental. The majority of type systems subsequently appearing in the literature include all of these features.

Linearity and the pi-calculus

@inproceedings{
    author = {Kobayashi, Naoki and Pierce, Benjamin C. and Turner, David N.},
    title = {Linearity and the pi-calculus},
    booktitle = {POPL},
    year = {1996},
    isbn = {0-89791-769-3},
    pages = {358--371},
    location = {St. Petersburg Beach, Florida, United States},
    url = {http://portal.acm.org/citation.cfm?id=237804&dl=#},
    publisher = {ACM},
    address = {New York, NY, USA},
}
This paper incorporates a simple linear typing discipline with the channel polarities established in the previous paper. Linearly-typed channels must be used exactly once. This extension to the type system is quite simple, but it has pleasant consequences: communications along linear channels cannot be interfered with. The authors use the lack of interference to justify optimizations in the Pict programming languages.

**Significance:** The contributions of the paper lie not so much in the definition of the type system—which is straightforward—but in the notion of typed bisimilarity it gives rise to. While typed behavioral equivalences can be usefully employed by compilers, it is less clear whether the type system is useful at the surface level to programmers.

**Language primitives and type discipline for structured communication-based programming**

```latex
@inproceedings{  
author = {Honda, Kohei and Vasconcelos, Vasco and Kubo, Makoto},  
title = {Language primitives and type discipline for structured communication-based programming},  
booktitle = {Programming languages and systems},  
year = {1998},  
url = {http://www.springerlink.com/content/fakx696bw3vx5kgr/},
}
```

**Types and subtypes for client-server interactions**

```latex
@inproceedings{  
author = {Gay, Simon and Hole, Malcolm},  
title = {Types and subtypes for client-server interactions},  
booktitle = {Programming languages and systems},  
year = {1999},  
url = {http://www.springerlink.com/content/kbtj6c4kmrgkfd0/},
}
```

These two papers together introduce *session types*, a way of typing long-running protocols over a single channel. Sessions are inherently symmetric: they take place between a client and a server, each having the dual perspective to the other. A session type tracks internal and external choices, simple sends and receives, and protocol recursion.

The second paper adds a notion of subtyping for session types, as a generalization of Pierce and Sangiorgi’s subtyping. It also reformulates the type system, casting it in terms of standard $\pi$-calculus rather than the extended language used in the first paper.

**Significance:** These two papers represent a very practical use of types in the $\pi$-calculus—in particular, one that matches the kinds of programs actually written. Adherence to a protocol is a basic form of correctness for client/server interactions, and session types capture a fairly wide range of protocols. This work has been quite influential, and research in the area is ongoing.