Programming with Proof Nets

Luke Simon

The lambda-calculus has long since been a prevalent foundational calculi for formalizing algorithms, and for good reason: it has many useful theoretical and practical properties. When the lambda-calculus was created, mathematicians were mostly interested in algorithms for computing functions, but in recent times, computer scientists have broadened interests to include algorithms for concurrent, interactive, and mobile problem domains – concepts which the lambda-calculus does not address.

The current trend in foundational calculi has been towards interactive, concurrent process algebras, such as the pi-calculus, which are commonly called "mobile-calculi". However, the pi-calculus lacks useful properties such as confluence, safety, and normalizability. While type extensions to the pi-calculus do exist, our approach is to, again, turn to metamathematics for answers. Just as the typed lambda-calculus is isomorphic to natural deduction proofs in minimal logic, a new mobile-calculus can be derived by making an analogy with this isomorphism. Linear Logic's natural deduction proofs, known as "proof nets", contain the intrinsic properties we are after: concurrency, interaction, confluence, safety, and normalizability. Furthermore, intersection types can be used to extend this calculus to be able to describe all processes that remain interactive and can be halted.

The properties that this formal system has can be exploited for designing a non-Turing complete next-generation programming language, in which multi-threaded programs do not contain concurrency related bugs such as race conditions, live-lock, and deadlock. In addition, programs written in such a language would also be guaranteed to never "lockup" as a correctly typed program remains "responsive". Such a language would be ideal for developing firmware for embedded "interactive" devices such as cell phones, kiosks, routers, etc.

Therefore, our intent is to derive a suitable mathematical framework, upon which practical applications can be developed such as new programming languages, varients of already existing popular programming languages, and automated verification systems.

References:

D. Sangiorgi and D. Walker, The pi-calculus: A theory of Mobile Processes. Cambridge University Press, 2001.

J.Y. Girard, Linear Logic., Theoretical Computer Science 50 (1987) 1-102.

R. J. Hindley, Types with intersection, an introduction. Formal Aspects of Computing, 4:470-486, 1992.