Combinators for Concurrent Computing

Raja Natarajan
Tata Institute of Fundamental Research
raja@tifr.res.in

Combinators arose in the analysis of foundations of mathematics. Combinatory logic was independently invented by Moses Schonfinkel and by Haskell Curry in an attempt to reduce logic to its simplest possible primitive basis, without taking implicit recourse to complex primitives. The subsequent impetus provided by computer science to research on combinators has led to deep insights in the theory of sequential programming, and has also had a great influence in the implementation of sequential programming languages.

In analogy with the study of combinators in sequential systems, we examine the notion of combinators in the setting of concurrent systems. In the field of concurrent computation, many new process calculi have emerged as suitable candidates for a basic framework which would help in modeling the essential aspects of interaction between concurrent computational processes. We explore and develop the discipline of combinators in the setting of process calculi for concurrent computation.

We design a system with six Basic Combinators and prove that it is powerful enough to embed the full asynchronous Pi-calculus, including replication [1]. Our Combinatory Calculus presents a significant departure from those propounded by Schonfinkel and Curry. Our theory for constructing Combinatory Versions of concurrent languages is based on a method, used by Quine and Bernays, for the general elimination of variables in formulations of first-order logic. Our combinators are designed to eliminate the requirement of names that are bound by an input prefix. They also eliminate the need for input prefix, output prefix, and the accompanying mechanism of substitution. We define a notion of bisimulation for the combinatory version and show that the combinatory version preserves the semantics of the original calculus. One of the distinctive features of the approach is that it can be used to rework several process algebras in order to derive equivalent combinatory versions.

We also extend the system of Combinators, to go beyond the realm of first-order quantification theory, and cover the entire lambda-calculus [2]. The higher-order system consists of five Combinators, powerful enough to represent lambda-abstractions over arbitrary terms. We provide algorithmic translations from lambda-calculus to its Combinatory Version, and vice-versa.

References

[1] N. Raja and R.K. Shyamasundar; Combinatory Formulations of Concurrent Languages, ACM Transactions on Programming Languages and Systems, Vol 19 (1997) pages 899–915.

[2] N. Raja and R.K. Shyamasundar: The Quine-Bernays Combinatory Calculus, International Journal for Foundations of Computer Science, Vol. 6, No. 4 (Dec 1995) pages 417–430.