# Special Issue Editorial Process Algebra*

Concurrency theory studies parallel and/or distributed systems, their design, specification and verification. Since the middle of the 1980s, the term Process Algebra has become the popular name to refer to the algebraic and axiomatic approach to concurrency theory, but of course, in retrospect, theories that can be considered as process algebra exist at least 10 years before that. In 20 years, process algebra has come of age. The articles in this special issue can tell the reader just that: process algebra is a branch of computer science like any other, with its own theory, tooling and applications, but also with many connections to other topics, in the first place within other parts of computer science, but also within logic and mathematics.

In concurrency theory, we are concerned with many aspects or properties of systems. To mention a few, we can be interested to know whether a system is free of deadlock or livelock, whether different subsystems are scheduled in a fair manner, whether timing constraints are met, whether or not a task will always terminate. When we consider a particular application area, we concentrate on the most important aspects of that area, as it is usually *undoable* to consider all aspects. Now it is possible to use a concurrency theory that is specifically geared to the set of aspects in question In this way, we get many different theories, each with its own favourite application area. It is preferable to have a general framework instead, that specialises into many different subtheories. Algebra offers such a general framework. Starting from a basic, general language, we can add a number of operators that capture relevant operators, and use a set of relevant manipulating rules in order to reason with the language.

Most well-known in the area of process algebra since the middle of the eighties are the theories CCS, CSP and ACP. CCS is the Calculus of Communicating Systems of [7], Theoretical CSP originates from [3] and the original reference to ACP is [2]. Of these three, (T)CSP is the most abstract (identifies more processes than the other two), and tends more in the direction of a specification language. The other two, CCS and ACP, are based on the same notion of equivalence (bisimulation), and are more operationally oriented, tend more in the direction of a programming language. Of the two, CCS has more links to logic and lambda-calculus, and ACP is more purely algebraical.

This issue concentrates on algebra, on calculations with processes. This means that very little attention is paid to semantics, to model-based reasoning. Semantic issues, in particular the use of transition systems, have been very important in process algebra research all along (see e.g. [4]), but here, we concentrate on the algebra.

The considerations above motivated the choice of articles for this issue. An additional motivation was to achieve unity of notation as much as possible. This introduces a bias towards ACP style process algebra (only the first article considers CCS). The reader should not take this to mean that other process algebras are less worth while. This is by no means the case. The main reason to concentrate on one particular process algebra is not to confront the reader with differing notation or subtly diverging meaning of very similar operators. Still, within this particular style of process algebra, we want to show different applications, different issues.

Finally, let us briefly review the contents. The first article, by Christensen, Hirschfeld and Moller, discusses decidability in process algebra. This is a line of research, that has really blossomed the last couple of years. The central observation is, that where language equivalence for context-free languages is undedicable, bisimulation equivalence for context-free languages is decidable. This article shows that decidability is achieved in CCS, if we disallow communication, and also if we disallow both restriction and relabelling. The results are proved by means of the tableau method for proof systems.

The second article, by Bergstra, Bethke and Ponse, introduces two iteration constructs in process algebra. Where systems are usually specified in process algebra as fixed points of recursive equations, they can be specified directly as terms over an algebra by the use of iteration operators like Kleene star. The iteration operators are given by a set of axioms, thus enabling algebraic reasoning. A number of facts about these operators are proven.

The third article, by Fokkink and Zantema, proves a fundamental result about the Kleene star operator of the previous article. It proves that the axiomatisation is complete for the standard model, i.e. that two terms can be proven equal using the axioms exactly when they are equivalent in the bisimulation model. The proof of this result requires a lot of theory from the area of term rewriting systems.

Thus, the first three articles are theoretically oriented. In contrast, the fourth article, by Mauw and Reniers, is geared towards an application. It uses process algebra in order to give a formal semantics to a graphical specification language, Basic Message Sequence Charts. This formal semantics is currently being standardised by the ITU-TS (the former CCITT).

The fifth article, by Korver, gives a formal definition of a simulator for process algebra. For many process

* This editorial should have appeared in Volume 37 number 4 of *The Computer Journal*. The Publishers wish to apologise for any inconvenience caused.

algebras, simulator tools exist, but this is the first time that a formal basis is provided for what they do, and what equivalence they induce on process terms.

Finally, the sixth article gives a full correctness proof for a communication protocol, a sliding window protocol with about 10,000 states. The proof is entirely algebraic, and is written in a process language in which we also can talk about data types in a formal way. It introduces novel concepts in order to modularise such lengthy proofs.

For those readers that want to know more about process algebra, the references below include four text books in the area ([BaW90, Hen88, Hoa85, Mil89]). I hope that all readers will find something to their liking in this issue.

*J. C. M. Baeten,*
*Eindhoven. The Netherlands*

## REFERENCES

[1] J. C. M. Baeten and W. P. Weijland, Process algebra, *Cambridge Tracts in Theor. Comp. Sci.,* 18, Cambridge University Press 1990.

[2] J. A. Bergstra and J. W. Klop, Process algebra for synchronous communication, *I and C,* 60, 1984, pp. 109–137.

[3] S. D. Brookes, C. A. R. Hoare and W. Roscoe, A theory of communicating sequential processes, *JACM,* 31, 1984, pp. 560–599.

[4] R. De Simone, Higher-level synchronising devices in Meije-SCCS, *TCS,* 37, 1985.

[5] M. Hennessy, *Algebraic Theory of Processes*, MIT Press, Cambridge MA, 1988.

[6] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall International, 1985.

[7] R. Milner, *A Calculus of Communicating Systems,* LNCS 92, Springer Verlag 1980.

[8] R. Milner, *Communication and Concurrency*, Prentice Hall International, 1989.