Concurrency

J. W. de Bakker
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

The Concurrency Project is devoted to the study of mathematical models of *parallelism.* Whereas a *sequential* computation runs on one processor, a parallel computation assumes a number of processors which interact in a variety of ways depending upon the particular situation. At the macroscopic level, one often speaks of *distributed systems*, and examples one may think of are networks of computers or distributed databases. At the microscopic level, parallel architectures are the prime example, where sometimes thousands of, mostly identical, processors cooperate in computations the organization of which is a major focus of current algorithmic research.

The emphasis in the project in our department (Department of Software Technology, CWI) is on *languages* for parallelism and, more specifically, on formal models which are exploited in the mathematical analysis of parallel programming concepts. The project participates in two collaborative efforts, viz. in ESPRIT project 415 and in the Dutch National Concurrency Project. Moreover, part of its foundational work is performed in a fruitful cooperation with the Free University of Amsterdam, the University of Kiel (FRG) and the State University of New York at Buffalo.

1. ESPRIT PROJECT 415

ESPRIT is the European Strategic Program for Research and Development in Information Technology. In project 415, six major European industries, with Philips as prime contractor, work together on 'Parallel Architectures and Languages for Advanced Information Processing: a VLSI Directed Approach'. Six subprojects investigate parallel architectures based on a variety of parallel programming styles such as object oriented, functional and dataflow, and logic programming. Moreover, the project has two working groups which provide the subprojects with a forum for integrating their activities in the areas of

Semantics and Proof Techniques, and of Architecture and Applications.

CWI is involved in project 415 as subcontractor of Philips. We firstly participate in the research concerned with the mathematical modelling of POOL. This is an acronym for Parallel Object Oriented Language, the language developed by Philips as conceptual starting point for the architectural design of its DOOM (Distributed Object Oriented Machine) system. So far, we have jointly designed an operational and denotational semantics for POOL [2,3], and we have made initial steps towards the design of a proof theory, i.e. a method for formally deriving properties of programs, for POOL [9]. Secondly, the Concurrency project participates in the activities of the Working Group on Semantics and Proof Techniques, and we coordinate the production of the yearly deliverable of the Working Group [10].

It is an important part of the policy of project 415 to spend substantial effort on educational and promotional activities, and the CWI has contributed to the organization of such events. The proceedings of the Advanced School on Current Trends in Concurrency (organized for ESPRIT 415 by the LPC, see Section 2 below) have appeared in [8]. Project 415 also sponsored the highly successful PARLE conference - Parallel Architectures and Languages Europe, June 1987. The PARLE proceedings were published in [6,7].

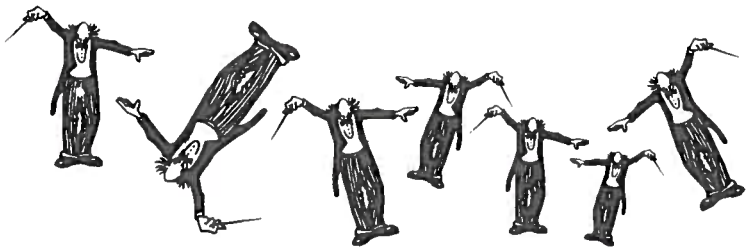
2. The Dutch National Concurrency Project

Under the joint direction of J.W. de Bakker, W.P. de Roever (Technological University Eindhoven) and G. Rozenberg (Leiden University), the Dutch National Concurrency Project (LPC abbreviating its Dutch name Landelijk Project Concurrency) has been operative since the beginning of 1984. The project is primarily sponsored by SION, the Research Foundation for Computer Science which is part of the Netherlands Organization for the Advancement of Pure Research (ZWO). The project concentrates on syntactic, semantic and proof theoretic aspects of Concurrency, and it has from its inception given great weight to effective cooperation with industrial research (in particular Philips and Shell laboratories). The section which concentrates on semantics of concurrency is located at CWI. J.N. Kok performs research on the semantics of nondeterministic dataflow, a computational model which underlies an important branch of contemporary research in parallel architectures. A recent achievement was the design of a 'fully abstract' mathematical model: briefly, this qualification amounts to the fact that the model has the right level of detail seen from the point of view of an observer of the behaviour of a dataflow net.

LPC has gradually become involved in a number of affiliated activities. A list of these activities includes: monthly 'concurrency days', devoted to tutorial and specialized presentations of research within the scope of LPC, visiting professors supported by the Dutch National Facility for Informatics, cooperation with ESPRIT projects 415 and 937 (Descartes: Debugging and Specification of ADA Real Time Systems), and formalized contacts with two other national programs: the French C^3 program (Cooperation, Concurrency and Communication) and the British *Alvey* Program, chapter Formal Aspects of Computer

Science. One example of an activity in the latter category is the C^3/LPC Concurrency Colloquium, held May 15, 16 at the CWI.

Our experience with LPC up to now has been most favorable: the cooperation has proven fruitful for our own concurrency research, both for scientific crossfertilization and as framework for a multitude of organizational activities.



The logo of the Dutch National Concurrency Project

3. Foundational research

Besides the activities which have been performed (mainly) under the auspices of the projects described above, we have also continued our ongoing research on the mathematical foundations of concurrency semantics, to a large extent embedded in joint subprojects with John Meyer (Free University of Amsterdam), Ernst-Rüdiger Olderog (University of Kiel) and Jeff Zucker (SUNY at Buffalo). We restrict ourselves to two examples: The *metric process theory* developed jointly with Zucker has proven a very useful tool for our denotational semantics for POOL, and comparative insights collected in [4,5] have found application in work in progress to assess various models for the new programming notion of *process creation,* abstracted from a key concept of the above mentioned language POOL (cf. [1]).

Finally, we mention our work on the semantics of the language *Occam,* a language gaining increased popularity due to the success of the *Transputer,* and our attempts to unify and interrelate a number of the semantic models for maybe parallel, maybe infinite, logic programming. Lastly, we want to point out that in the arrangements of instruments tuned for concurrency semantics, a key part is also due to the *process algebra* movement, ably conducted by Bergstra and Klop (CWI, Department of Software Technology).

4. A small case study

A sequential program S may, semantically, be seen as a function ϕ which transforms a given input state σ to an output state σ'. For example, the assignment $x := x+1$ will change a state where x equals zero to a state where x equals one. Notationally, we express this by writing $[\![S]\!] = \phi \in \Sigma \to \Sigma$: *the meaning* $[\![S]\!]$ of S is a function ϕ from domain Σ to codomain Σ. Sequential composition $S_1; S_2$ is modelled in a straightforward manner by *function composition.* In a context where *nondeterminism* is present, we enlarge the

codomain Σ to the *power set* $\mathcal{P}(\Sigma)$. It is then also meaningful to model the meaning $[\![S_1 \,[]\, S_2]\!]$ of the nondeterministic choice construct $S_1 \,[]\, S_2$ by the function $\lambda\sigma.([\![S_1]\!](\sigma) \cup [\![S_2]\!](\sigma))$, where the two operands of the set union '$\cup$' in this function are elements of $\mathcal{P}(\Sigma)$. (The notation $\lambda\sigma. \cdots \sigma \cdots$, where $\cdots \sigma \cdots$ is some expression involving σ, defines a function which maps the argument σ to the value $\cdots \sigma \cdots$.) There is an essential difficulty we encounter when we want to extend this approach to the language construct $S_1 \| S_2$ standing for the parallel execution of S_1 and S_2 - here and often elsewhere taken in the sense of the arbitrary interleaving of the elementary ('atomic') actions making up S_1 and S_2. There is no immediate mathematical operator modelling the programming notion of parallel execution, and in the semantic research one has resorted to a number of ways out of this problem. One method yielding only partial solace is to abstract from programs as *state transformations* and to use models which consist of structured entities - sets of sequences or tree-like structures - consisting of atomic or uninterpreted actions only.

For example, instead of working with building blocks such as the assignment $x := x + 1$, one now views this as nothing but an atomic action a. This gives rise to models which have a strong flavor of *formal language theory*: there is little surprise here, since sequences of elementary actions are eminently suitable to undergo the operation of *merge*, contrary to what is the case for (state transforming) functions. A second method, which we have in fact been advocating for some time, following Scott's *general domain theory* and Plotkin's introduction of *'resumptions'*, is to introduce a process domain P as solution of the 'equation'

$$P \cong \{p_0\} \cup (\Sigma \to \mathcal{P}_{\text{closed}}(\Sigma \times P))$$

(the work involved in solving the equation takes place in a metric setting, $\cong$ denotes isometry and p_0 the nul process). Now assume $p \in P$. Then either p equals p_0, or p is a function, and it is meaningful to write $<\sigma', p'> \in p(\sigma)$: this expresses that process p, for input σ, yields output σ' *together with* the resumption p'. Processes have now a structure which combines features of sequences ($\Sigma \times P$), sets ($\mathcal{P}_{\text{closed}}(\cdots)$) and functions ($\Sigma \to \cdots$), and the semantic operators '$\circ$', '$\cup$' and '$\|$' can all be defined satisfactorily. This allows typical denotational equations such as

$$[\![S_1 \| S_2]\!] = [\![S_1]\!] \| [\![S_2]\!].$$

Many more details of this style of semantic definitions may be found in [3].

References

1. P. America, J.W. de Bakker (1987). *Designing Equivalent Semantic Models for Process Creation*, Technical Report CS-R8732, CWI, Amsterdam.
2. P. America, J.W. de Bakker, J.N. Kok, J.J.M.M. Rutten (1986). Operational semantics of a parallel object oriented language. *Conference Record of the 13th ACM Symposium on Principles of Programming Languages* (St.

Petersburg, Florida), 194-208.

3. P. AMERICA, J.W. DE BAKKER, J.N. KOK, J.J.M.M. RUTTEN (1986). *A Denotational Semantics for a Parallel Object Oriented Language,* Technical Report CS-R8626, CWI, Amsterdam.
4. J.W. DE BAKKER, J.N. KOK, J.-J.CH. MEYER, E.-R. OLDEROG, J.I. ZUCKER (1986). Contrasting themes in the semantics of imperative concurrency. J.W. DE BAKKER, W.P. DE ROEVER, G.ROZENBERG (eds.). *Current Trends in Concurrency, Lecture Notes in Computer Science 224,* 51-121 Springer.
5. J.W. DE BAKKER, J.-J.CH. MEYER, E.-R. OLDEROG, J.I. ZUCKER (1986). *Transition Systems, Metric Spaces and Ready Sets in the Semantics of Uniform Concurrency,* Technical Report CS-R8601, CWI (to appear in *Journal of Computer and System Sciences).*
6. J.W. DE BAKKER, A.J. NIJMAN, P.C. TRELEAVEN (eds.) (1987). *Proceedings PARLE - Parallel Architectures and Languages Europe, Vol. I: Parallel Architectures, Lecture Notes in Computer Science 258,* Springer.
7. J.W. DE BAKKER, A.J. NIJMAN, P.C. TRELEAVEN (eds.) (1987). *Proceedings PARLE - Parallel Architectures and Languages Europe, Vol. II: Parallel Languages, Lecture Notes in Computer Science 259,* Springer.
8. J.W. DE BAKKER, W.P. DE ROEVER, G. ROZENBERG (eds.) (1986). Current Trends in Concurrency, Overviews and Tutorials *(Proceedings ESPRIT/LPC Advanced School), Lecture Notes in Computer Science 224,* Springer.
9. F.S. DE BOER (1986). A proof rule for process creation. M. WIRSING (ed.). *Proc. Formal Description of Programming Concepts III,* preliminary version, Ebberup.
10. P. AMERICA, K.R. APT, J.W. DE BAKKER et al. (1986). Deliverable year November 1, 1985-November 1, 1986 Working Group on Semantics and Proof Techniques ESPRIT project 415 (available from mr. F. Stoots, Philips Research, P.O. Box 80000, 5600 JA Eindhoven).