Description of CONCUR : ESPRIT basic research action 3006

Description of CONCUR

ESPRIT Basic Research Action 3006

J.C.M. Baeten

CONCUR coordinator,
Dept. of Software Technology,
Centre for Mathematics and Computer Science,
P.O.Box 4079, 1009 AB Amsterdam, The Netherlands.
e-mail: josb@cwi.nl.

We give a description of the aims and objectives, baseline and rationale, and structure of the ESPRIT Basic Research Action 3006, CONCUR (Theories of Concurrency: Unification and Extension). CONCUR is a two year research project funded by the European Communities, involving 7 scientific institutions in 4 countries. We announce the conference CONCUR 90.

1. PARTICIPANTS

The project CONCUR (Theories of Concurrency: Unification and Extension) started September 1, 1989, and runs for two years. The project brings together researchers at the following 7 institutions. We list for each partner the acronym, full name and key personnel.

CWI	Dept. of Software Technology, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands. J.C.M. Baeten, J.W. Klop.
UEd	Laboratory for Foundations of Computer Science, University of Edinburgh, UK. R. Milner, D.J. Walker.
UOx	Programming Research Group, University of Oxford, UK. C.A.R. Hoare, M.B. Josephs.
USx	Dept. of Computer Science, University of Sussex, Brighton, UK. M. Hennessy.
INRIA	INRIA, Sophia Antipolis, France. G. Berry, E. Madelaine.
SICS	Swedish Institute of Computer Science, Kista, Sweden. J. Parrow
UAm	Programming Research Group, University of Amsterdam, The Netherlands. J.A. Bergstra, S. Mauw.

2. SUMMARY

Formal verification of software programs, protocols and chip designs, is becoming increasingly important, as systems become more complex, production more expensive, and testing is widely recognized as being insufficient to guard against malfunction.

Verification of concurrent or distributed systems has up till now been undertaken only on a very small scale, in a haphazard way, and with a multitude of techniques and formal theories. For industrial applicability, it is essential that some unity emerge from the competing theories of concurrency, and that the verification process be supported by reliable software tools.

Normal academic interchange will slowly bring unity into the disparate world of concurrency theories, but collaboration of a more intense kind is needed to accelerate the process. Among the many formal approaches which exist for concurrent communicating systems, the important algebraic approaches are represented in this project.

The principal aims of the project are to explore the relationships among these different approaches, and to develop a formalism applicable to a wide range of case studies. We will investigate the possibilities for integration, or at least incorporation of features of one system in another. We will elucidate the relationship and relevant roles for algebraic and model-based theories. We will compile a list of open problems of agreed interest. Methods pioneered in one group can be applied to problems originating in another, and results can be compared. In addition to collaborating at the theoretical level, we will collaborate through the development, use and comparison of software reasoning tools. This latter collaboration will serve both to further unity and to enhance the theoretical collaboration, as experience has shown that development and use of software tools acts as a catalyst for theoretical advance.

Tools find many applications in case studies. Case studies are important for the comparison of theories of concurrency, since the true strength of a theory will only appear through its use in tackling substantial case studies, systems of a considerable size. Such systems are complex, and thus require special syntactical constructions and mechanical assistance to relieve tedium and to check rigour in reasoning. We find that graphical aids are often exceedingly helpful in understanding them, and therefore, these should be developed. Large case studies will in all cases be relevant for the European software industry. Several of the partners already have ongoing tool building activities.

The proposed action provides an outstanding opportunity to coordinate the participants' well-established programmes of research into theories of concurrency.

3. OBJECTIVES.

The proposed action is in the area of Computer Science or Software Technology, more specifically: Formal methods in software engineering, distributed algorithms and protocols. Its binding goals are in two different directions:

a) development of accepted standards
Accepted standards will enable people working in future research or implementation, to import new features - developed at other sites - in their formal specification language. This way, a higher level of compatibility between the many different formalisms and approaches in algebraic concurrency semantics is achieved, and results which are established at the one site directly apply to any of the others as well.

b) development of a tool set
The production of a common tool set will enforce cooperation between the various partners. It will provide a coherent workbench with less software redundancy. An important feedback should come from software experiments and give new insights on theoretical concepts to experimentation and implementation, accelerating greatly their integration to existing theory. The resulting tool set will provide a strong basis for further software development in the area of verification of concurrent systems.

By means of these goals the action will provide the European IT industry with new technologies that can help to meet the competitive requirements of the 1990's. Since the action hopes to unify the different algebraic theories of concurrency into a wide spectrum formalism, applicable on a range of case studies, it will contribute to the development of an internationally accepted standard in this area. This unification will be greatly accelerated by the proposed action. Studies will be undertaken on extensions of the theories, in order to expand the range of applicability.

4. BASELINE AND RATIONALE

As indicated in the summary, the participants have well-established programmes of research into concurrency theories, and are the originators of the most significant theories in Europe in this field (see references). The present state of algebraic and logical theories of concurrency may be summarised briefly as follows. Many fundamental and promising ideas have arisen in the last eight years; to harmonise them requires attention from the originators to avoid as far as possible the danger that the present formulations, which are not perfected, become enshrined in professional methodology. The project is well-timed to attack this problem.

At this moment, the participating institutes are studying on various aspects of concurrency theory (such as semantics and extension of languages), on problems on the specification of systems, on case studies for process verification and on the development of research tools - aimed at theoretical study - in order to be able to go beyond the level of studying only small examples of theoretical interest. In order to avoid a serious divergence between the methods that are developed, the coordination of all work in one project has become necessary.

The current 'state of the art' can best be described by presenting a brief overview of the research being employed at the various sites:

CWI: At CWI, research concerns the many possible semantics that can be defined for concurrent systems. Among such semantics we find graph models, transition models, partial order semantics and event structures. In the setting of the Algebra of Communicating Processes (ACP) complete algebraic axiomatisations are sought for some of the corresponding equivalence notions. Furthermore, the language ACP is equipped with new features and operators, in order to further enlarge its range of applications. References: [Ba1], [BK1,2,3].

UEd: Much work at UEd takes as its starting point operational semantics of communicating systems. Associated proof techniques, such as that for bisimulation equivalence, are studied, and complete axiomatizations are sought. Complementary to the algebraic work, logics for concurrent systems, including modal and temporal logics, are subject of interest. The relationships between the algebraic and the logical theories are also studied. Much of this approach is reflected in the Concurrency Workbench. References: [Mi1,2,3].

UOx: At UOx, research into the theory of Communicating Sequential Processes (CSP) has investigated both denotational semantics, involving the construction of mathematical models of processes, and algebraic semantics, the laws that facilitate transformation between behaviourally-equivalent networks of processes. A model and an algebra evolve together, each shedding new light on the other: it is the model that defines exactly what is meant by the observable behaviour of a process; it is the algebra that provides us with a convenient notation for describing individual processes and combining them into networks. References: [Ho], [RH].

USx: A wide variety of behavioural equivalences for concurrent processes have been formalised and investigated at USx. These are usually defined by abstracting in some way from an operational semantics for processes and we have done much work on alternative characterisations for these equivalences. These include algebraic theories and denotational models. Currently, the main field of interest is that of processes with real-time or priority features and of developing algebraic theories which reflect a difference between concurrency and nondeterminism. Reference: [He1,2].

INRIA: The MEIJE project at INRIA has both a theoretical and a practical research activity. At the theoretical level, we study various aspects of parallelism and concurrency models. We have explored the relationship between classical process calculi and we have developed the synchronous language ESTEREL. ESTEREL is dedicated to the programming of real-time reactive systems and of command automata. ESTEREL is now getting an industrial development. At the level of program analysis we are developing verification software tools based on the transition systems model. The ECRINS system implements algorithms for symbolic manipulation of various process calculi or from concurrent languages. AUTOGRAPH is a system for the graphical manipulation of networks of automata. Using AUTO and AUTOGRAPH in combination permits the analysis of non-trivial concurrent programs, including communication protocols and ESTEREL reactive systems. References: [BG], [BS], [LMV], [MS].

UAm: At UAm research is concentrated on the algebra of communicating processes (ACP). ACP is an axiomatic approach towards the specification of concurrent systems. Systems specified in ACP can be proved equivalent by means of these axioms. Extensions of ACP and semantic models for ACP are studied. To be able to handle modular specifications, the modularization concepts of Module Algebra were added to ACP. A tool set for the specification, verification and implementation of concurrent systems is being constructed. This work is based on Process Specification Formalism (PSF), which can be considered as a concrete syntax for ACP, including the possibility to specify the data types involved with a specification. References: [BK1,2,3], [MV].

Many of the participating groups are involved in other European projects in order to guarantee a minimal level of communication with other fields in Computer Science. In particular, we encourage relations with the Basic Research Actions 3011 (CEDISYS, Compositional Distributed Systems), 3096 (Formal Methods and Tools for the Development of Distributed Real-Time Systems) and 3148 (DEMON, Design Methods based on Nets).

5. EXPLOITATION.

The impact of CONCUR on industrial applications and procedures will initially not be very large, but is potentially considerable in the medium to long term.

The action hopes to provide some uniform approach to the algebraic theory of synchronous and asynchronous concurrent systems. This covers the fields of communication protocols, distributed systems, operating systems, reactive systems, etc.

Such a strong mathematical basis is required for future developments of standard languages and systems. The action will initiate the production of a tool set, which should allow designers of new concurrent languages to experiment our verification techniques with their own semantic definitions. This can increase greatly the speed and quality of language definition and standardization.

Prototypes of these tools can be available at the end of the action. Since the action hopes to unify the different algebraic theories of concurrency into a wide spectrum formalism, applicable on a range of case studies, it will contribute to the development of an internationally accepted standard in this area. Standards are wanted in this area, as the development of the LOTOS formalism shows.

The tools proposed in the LOTOSPHERE project (ESPRIT II 2304) are specific to the LOTOS language. These tools must accept the whole LOTOS international standard and take into account all aspects of the language. Tools devoted to external syntax analysis and manipulation are of great importance; tools aimed at semantic analysis will take advantage of the specificity of LOTOS. The inital workplan of LOTOSPHERE does not include verification tools.

In contrast, the workbench proposed in the project CONCUR concerns more the fundamental aspects of the theory of concurrency, with a strong emphasis on verification. Rather than concentrating on a specific theory, we aim at building a tool set with a wide spectrum of applications, suitable for comparison of various approaches of concurrency and for testing of new algebraic process calculi.

Naturally, fundamental algorithms and methods developed inside the CONCUR action will provide a strong basis for further developments of verification tools for specific languages, including LOTOS.

6. THE WORKPLAN.

The nature of basic research is such that it is hard to predict in advance what research will be done during a certain period of time. Also, new ideas will come up during the course of the project. Therefore, we cannot pin ourselves down to a too rigid schedule, but must allow for regular changes and updates. The workplan is divided into three workpackages. Each workpackage is divided into tasks.

6.1. WORK PACKAGE 1. UNIFICATION AND COMPARISON.

Task 1.1 Comparing process algebras
Important algebraic approaches to concurrency as CCS, CSP, ACP, MEIJE are represented in the project. We acquaint everyone in the project with each of the theories, and compare the different theories, by a regular exchange of papers and reactions to papers, where comparisons are made, and advantages and disadvantages of each theory become clear.
References: [Br], [vG], [DH].

Task 1.2 Comparative concurrency semantics
In this project, a number of different approaches to concurrency is represented. We will undertake a mutual comparison of the different approaches, as a first step towards the goal of integration.

We want to come to a classification of process semantics, in a lattice partially ordered by the relation "makes more identifications on processes". Most semantical notions encountered in contemporary process theory can be classified along four different lines, corresponding with four different kinds of identifications. First there is the dichotomy of *linear time* versus *branching time*: to what extent should one identify processes differing only in the timing of the divergencies between their different courses of actions. Secondly there is the dichotomy of *interleaving semantics* versus *partial order semantics*: to what extent should one identify processes differing only in the causal dependencies of their actions (while agreeing on the possible orders of execution). Thirdly one encounters different treatments of *abstraction from internal actions* in a process: to what extent should one identify processes differing only in their internal or silent actions. And fourthly there are different approaches to *infinity*: to what extent should one identify processes differing only on their infinite behaviour.

Well-known semantics as bisimulation semantics, failure semantics, readiness semantics, testing equivalence, ready trace semantics, barbed semantics, failure trace semantics, refusal semantics, trace semantics, maximal trace semantics can be located in this lattice, the ideas involved in their construction can be unraveled and combined in new ways.
References: [BR], [BKO].

Task 1.3 Unification through tools
Several of the participants have experience in the development of automated tools for reasoning about concurrent systems (see work package 3 below). Through the medium of these implementations the different algebraic and logical combinators, and different notions of process equivalence, will be assessed in use. The collaborators in this project include the originators of most of the prominent theories in Europe. They each have good reasons for using their theories. The above paragraphs have indicated how a unified theory of concurrency may be developed partly through careful theoretical study; on the other hand, it may be that the deciding factor is how the theory works for a wider community working on weighty applications. This community must be provided with an automated system which implements the different theories, or an attempt at a unified theory, before it will undertake such an experiment.

This thread of work, emphasizing the automated tools, will be intimately intertwined with the existing theoretical program at UEd, USx, SICS and INRIA.

Task 1.4. Case studies
The first year of the action will provide an excellent opportunity to design suitable medium-sized case studies. During this year we shall exchange suggestions for such case studies, and shall reach agreement upon a selection of them at the general meeting in month 12 of the first year. Each formalism will then be applied to some of the chosen studies.

On the one hand, case studies are useful to test the different theories, and on the other hand, tools can be tested in the application to case studies.
Reference: [Ba2].

6.2. WORK PACKAGE 2. EXTENSIONS.

This work package is concerned with extending existing theories of concurrency and their applicability.

Task 2.1 Value passing
Most research carried out on process algebras assume that processes do not transmit data between each other; they merely synchronise. We intend to investigate the possibilities of extending process algebras in a number of ways so as to allow values to be passed between processes. At the moment we see three strands to this topic.
1. We will reexamine the "traditional" method of using pure processes, which only allow synchronisation, to model value-passing. Here a process which can input a value on a channel c is modelled by a similar process which can synchronise on an number of different actions, c?v, one for each value v in the data-type under consideration. We will investigate the possiblity of replacing this infinite choice with a process which can only perform one action - input on channel c, but whose residual is now a function parameterised on the data-type. This change in view may lead to more tractible theories. Reference: [He3].
2. We also want to extend process algebra to cover communication of port names between pocesses. This will make process algebra applicable to systems with a dynamic inter-connection topology. Some work has already been done on this topic but it lacks suitable theoretical support. References: [MPW1,2].
3. Also, we will study Algebraic Data Types in connection with the specification of concurrent programs. As a first step towards a general specification formalism for processes, we need to be able to talk about process names and atomic actions that are parametrized by elements of a (possibly infinite) data structure. Next, we want to be able to have alternative and parallel composition indexed over such a data structure. Reference: [MV].

Task 2.2 Time and probabilities
It is desirable to extend existing theories of concurrency to address issues of real-time systems design. Real-time systems must operate to within acceptable delay-tolerances and must function correctly with a high degree of certainty. We will identify and explain key concepts in the modelling of time and probability. Our hope is to find an algebra which will slot into the hierarchy of algebras already available for dealing with concurrency at higher levels of abstraction.
References: [RR], [BB2], [LS], [GJS].

Task 2.3 Linking Theory and Practice.
It is clearly important to bridge the gap between sound, mathematical theories of concurrency and practical design methods for use in an industrial environment. By providing a mathematical underpinning for an existing practical design method, such as JSD (Jackson System Development), defects of the method and solutions to them will become apparent, leading to improvements in practice and better understanding of theory. Another area where theories of concurrency have already yielded benefits in practice are in VLSI design, for example the trace theory of Martin Rem, and the self-timed circuits of Alain Martin. Algebraic methods, like those used successfully in the occam transformation system, may be extensible to these other areas too.

Furthermore, we would like to provide a unified foundation for the design of systems in mixed technology.
Reference: [JHJ].

Task 2.4 Other extensions
At Amsterdam, extension of process algebra with several other operators has been studied, e.g. a priority operator ([BBK]), a state operator ([BB1]), a process creation operator, a mode transfer operator, signals and observations. We will continue this work, and also will investigate the addition of these operators to other process algebras. The study of additional operators has two aspects. Firstly, one wishes to discover which operators are practically useful, and work in this direction was done in [Ho]. It can be continued with the help of implementations in experiment. Secondly, each new operator raises questions both for the algebraic calculi and for the associated logics.

6.3. WORK PACKAGE 3. AUTOMATED TOOLS.
We are interested in the development, the use and comparison of software reasoning tools. As described in the Summary and section 3 above, such tools will also further unity. The emphasis within the action will be on a theoretical foundation of such tools, and a requirements analysis, stating what kinds of tools are needed, and what they are intended to do.

Task 3.1 Distribution of tools
At the outset of the project, the tools mentioned below and associated documentation were distributed to all partners.

1. The CONCURRENCY WORKBENCH (developed at UEd, USx, SICS) is a collection of software tools for investigating the operational semantics of processes. One can define a process using the syntax of CCS and then the system will accept a number of commands for investigating its symbolic execution. It will also check whether or not two processes are semantically equivalent (under so-called bisimulation equivalence). One can also define properties of processes using a modal logic and the system will check if a given process has a given property. There is also a subsystem which enables one to design a system which will be equivalent to a specification given predefined constraints.

2. ECRINS (INRIA) is a process calculus workbench. The user can define his own process calculus by giving a syntactical and semantical description. The semantical description is a set of Plotkin operational rewrite rules, in the style of most existing process calculi. Once a calculus is defined, the user can parse terms of the calculus, study their behavior using an elaborate system of tactics and tacticals to guide term evaluations, test term equivalences (strong bisimulations parametrized by equational simplifiers), or compute specifications of terms, that is complete sets of rules which terms obey. Most classical calculi are available in ECRINS (CCS, SCCS, MEIJE, TCSP, ...). The system has been used to verify algebraic properties of process calculi and to compare calculi with each other.

3. AUTO (INRIA) is a system dedicated to manipulation of finite automata and networks of finite automata. The input langage is a subset of MEIJE, restricted to static networks of finite automata. The compiler of our ESTEREL real-time programming language is also able to produce input to AUTO from arbitrary ESTEREL programs. The user defines abstract actions as rational sets of sequences of concrete actions. The AUTO system efficiently performs reductions of automata w.r.t. abstract actions: the reduction of an automaton yields a simpler automaton that only contains abstract actions. The reduction technique permits to observe partial properties of a given automaton and gives several observation angles on local or global properties of the automaton. AUTO also checks for automata equivalences. The combination of reductions and equivalence checks is used to prove correctness of programs with respect to abstract specifications. AUTO is currently used to prove the behavioral correctness of communication protocols and of real-time systems.

4. AUTOGRAPH (INRIA) is a mouse-driven multi-window environment for graphical manipulation of networks of finite automata. It can be used as an interactive input device for AUTO. It can also be used as an output device: there are facilities to guide the drawing of automata computed by the AUTO and ESTEREL systems.

The tools will be used and investigated at all sites. Feedback will be given to tool authors, and different tools can be combined into an "Algebraist's Assistant".
References: [CPS], [LMV], [MS].

Package 3.2 Investigating and building tools

Task 3.2.1 The Concurrency Workbench.
Work will continue on the development of the Concurrency Workbench implementing more algorithms for analysis, building a graphics interface, and applying it to more examples.
i. We will provide interactive proof assistance, using a tableau formalism whose atomic formulae express that "Process P satisfies property F". Proofs derive their structure both from the processes and from the formulae. At first, the research will treat only finite-state systems in depth; there are strong indications that the right methods for infinite-state systems, for example systems in which arbitrary data objects are used in communication, can best be arrived at without initial distraction by questions of data type, logical theories of data etc.
ii. Examples of proof will be examined in which data theories and infinite state-space play an important part. On the basis of this work the next stage in the Concurrency Workbench will be designed and implemented.

iii. Algorithms for the analysis of concurrent processes will be extended. A prime example is to extend the algorithm for finding bisimulations in finite state spaces to processes with a restricted form of value passing where the values range over infinite data domains.
iv. A set of demos will be available after 24 mos. That is, apart from written reports about the achievements that have been made, executable Software Ware will be available according to the report descriptions.

The collaboration among UEd, USx and SICS on the development of the Workbench will continue, and we shall look for ways of combining insights with INRIA.

Task 3.2.2 INRIA tools

The development of INRIA tools will continue. We shall not start building new software systems, but we shall bring our existing systems to a strong and stable state. We shall extend our tools with new theoretical and pragmatic features, enforcing interfaces between our tools and between them and the Concurrency Workbench, and avoiding any unnecessary duplication of software.
i. Process calculi experimentation: The Ecrins system will be extended to deal with equivalences based on general abstraction mechanisms. This includes the extension of the abstract action reduction technique of Auto to the general case of possibly infinite transition systems defined by structural rules. This should allow us to relate abstraction and implementation notions and to inter-translate different process calculi with different action structures such as CCS, TCSP, SCCS, or Meije.
ii. Extension of the finite automaton approach. Auto should take its input not only from Meije but from arbitrary process calculi, provided that appropriate syntactic restrictions are given to restrict terms to generate finite automata. This should of course be done by using the Ecrins system.
iii. Development of debugging tools for Auto. When Auto detects the inequivalence of automata, it presently produces rather poor error messages. We plan to produce better messages such as paths showing the inequivalences or preferably temporal logic formulae that explain why automata are inequivalent.
iv. Enhancements to Autograph. We aim to provide a general graphical interface to deal uniformly with all Ecrins-definable calculi, including graphical mechanisms to draw user-defined operators with arguments. We also plan to develop graphical display and debugging tools for AUTO.
v. Development of the ESTEREL/AUTO interface. The interfaces from the ESTEREL compiler to AUTO and AUTOGRAPH will be enhanced. We shall experiment on verifying properties on the automata derived from ESTEREL programs.

Task 3.2.3 Comparison of tools

It is likely that the weaknesses of some tools (no axiomatic treatment of equivalence, no testing theory, no treatment of divergence and fairness, no temporal logics, ...) will match the strengths of other research groups and of their tools. A careful comparison will be made of the tools developed at different sites; the aim will be to combine software where appropriate, to exchange experience of developing such programs, and to arrange to tackle complementary problems.

Since everybody wants to avoid redundancy in large tools construction, bridges should be developed early as possible in between the various tools. However, bridges do not mean complete integration. We do not believe that a single approach can treat all problems. Instead, we think that it is useful to apply the various approches to a common set of examples to know which technique is best where.

7. INFORMATION DISSEMINATION.

All reports, deliverables and other achievements of the project shall be reported on in scientific publications in journals and proceedings. All deliverables of the project have public status. Tools that are developed further in this project, will remain to be available free of charge to all universities, and will also be made available to other institutions.

8. CONFERENCE CONCUR 90.

The project will organise two conferences: CONCUR 90 will be held in Amsterdam from 27 to 30 August 1990, CONCUR 91 will be held in Edinburgh. An open call for papers for CONCUR 90 has already been sent out, listing as main topics:

- formal methods in software engineering;
- distributed algorithms and protocols;
- formal specification languages and their semantics;
- verification methods;
- tools for the verification and design of concurrent systems.

Deadline for submission of a draft full paper of no more than 15 pages is March 1, 1990. The programme includes four invited lectures and four tutorials. The programme committee consists of J.A. Bergstra, G. Berry, E. Best, C.A.R. Hoare, J.W. Klop, K.G. Larsen, R. Milner, U. Montanari, E.-R. Olderog and J. Parrow.

Further inquiries can be directed to the programme committee chairman, J.W. Klop, or the organizing committee chairman, J.C.M. Baeten, both at CWI.

ACKNOWLEDGEMENTS
This document was compiled by the project coordinator from the Technical Annex. The author acknowledges the valuable input he received from all partners of CONCUR.

8. BIBLIOGRAPHY.

[Ba1] J.C.M. Baeten, *Procesalgebra,* Kluwer Deventer 1986 (in Dutch).
[Ba2] J.C.M. Baeten, ed., *Applications of process algebra,* Cambridge Tracts in Theor. Comp. Sci., Cambridge University Press, to appear in 1990.
[BB1] J.C.M. Baeten & J.A. Bergstra, *Global renaming operators in concrete process algebra,* I&C 78, 1988.
[BB2] J.C.M. Baeten & J.A. Bergstra, *Real time process algebra,* report P8916, Programming Research Group, University of Amsterdam 1989.
[BBK] J.C.M. Baeten, J.A. Bergstra & J.W. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra,* Fund. Inf. IX, 1986.
[BK1] J.A. Bergstra & J.W. Klop, *Process algebra for synchronous communication,* I&C 60, 1984.
[BK2] J.A. Bergstra & J.W. Klop, *Algebra of communicating processes with abstraction,* TCS 37, 1985.
[BK3] J.A. Bergstra & J.W. Klop, *Process algebra: specification and verification in bisimulation semantics,* in: Math. & Comp. Sci. II (M. Hazewinkel, J.K. Lenstra & L.G.L.T. Meertens, eds.), CWI Monograph 4, North-Holland, Amsterdam 1986.
[BKO] J.A. Bergstra, J.W. Klop & E.-R. Olderog, *Failures without chaos: a new process semantics for fair abstraction,* in: Proc. IFIP Conf. on Formal Descr. of Progr. Concepts - III, Ebberup 1986 (M. Wirsing, ed.), North-Holland, Amsterdam 1987.
[BG] G. Berry & G. Gonthier, *The synchronous programming language ESTEREL: design, semantics, implementation,* to appear in SCP, 1988.
[BS] G. Berry & R. Sethi, *From regular expressions to deterministic automata,* TCS 48, 1986.
[Br] S.D. Brookes, *On the relationship of CCS and CSP,* in: Proc. 10th ICALP, Barcelona (J.Diaz, ed.), LNCS 154, Springer Verlag 1983, pp. 83-96.
[BR] S.D. Brookes & W.C. Rounds, *Possible futures, acceptances, refusals and communicating processes,* in: Proc. 22nd Symp. on Found. of Comp. Sci., IEEE, New York 1981.
[CPS] R. Cleaveland, J. Parrow & B. Steffen, *The Concurrency Workbench: a semantics based tool for the verification of concurrent systems,* report ECS-LFCS-89-83, University of Edinburgh 1989.
[DH] R. De Nicola & M. Hennessy, *CCS without τ's,* Proc. TAPSOFT 87, LNCS 250, Springer 1987.
[GJS] A. Giacalone, C.-C. Jou & S.A. Smolka, *Algebraic reasoning for probabilistic concurrent systems,* SUNY at Stony Brook, to appear.
[vG] R.J. van Glabbeek, *Notes on the methodology of CCS and CSP,* report CS-R8624, Centre for Math. & Comp. Sci., Amsterdam 1986.
[GLZ] J.C. Godskesen, K.G. Larsen & M. Zeeberg, *TAV (Tools for Automatic Verification) users manual,* report R 89-19, University of Aalborg 1989.

[He1] M. Hennessy, *An algebraic theory of concurrency*, MIT Press, Cambridge Ma. 1988.
[He2] M. Hennessy, *Axiomatising finite concurrent processes*, SIAM Journal of Computing 1988.
[He3] M. Hennessy, *A proof system for communicating processes with value-passing*, Technical Report 3/89, University of Sussex 1989.
[Ho] C.A.R. Hoare, *Communicating sequential processes*, Prentice Hall 1985.
[JHJ] M.B. Josephs, C.A.R. Hoare & He Jifeng, *A theory of asynchronous processes*, Oxford University Programming Research Group 1989.
[LS] K.G. Larsen & A. Skou, *Bisimulation through probabilistic testing*, report R88-29, Aalborg University 1988.
[LMV] V. Lecompte, E. Madelaine & D. Vergamini, *Auto: a verification system for parallel and communicating processes*, INRIA Sophia Antipolis 1988.
[MS] E. Madelaine & R. De Simone, *ECRINS un laboratoire de preuve pour les calculs de processus*, R.R. INRIA 672, 1987.
[MV] S. Mauw & G.J. Veltink, *A process specification formalism*, report P8814, Progr. Res. Group, UAm 1988.
[Mi1] R.Milner, *A calculus of Communicating Systems*, Lecture Notes in Computer Science Vol 92, Springer-Verlag, 1980.
[Mi2] R.Milner, *Operational and Algebraic Semantics for Concurrent Processes*, LFCS Report No ECS-LFCS-88-46, 1988 (to appear in the Handbook of Theoretical Computer Science).
[Mi3] R. Milner, *Communication and concurrency*, Prentice Hall 1989.
[MPW1] R. Milner, J. Parrow & D. Walker, *A calculus of mobile processes, part I*, report ECS-LFCS-89-85, University of Edinburgh 1989.
[MPW2] R. Milner, J. Parrow & D. Walker, *A calculus of mobile processes, part II*, report ECS-LFCS-89-86, University of Edinburgh 1989.
[Pa1] Joachim Parrow. *Fairness Properties in Process Algebra*. PhD Thesis, Uppsala University, 1985.
[Pa2] Joachim Parrow. *Submodule Construction as Equation Solving in CCS*. In Nori (Ed) Foundations of Software Technology and Theoretical Computer Science, LNCS 287.
[Pa3] Joachim Parrow. *Verifying a CSMA/CD Protocol in CCS*. To appear in The 8:th International Workshop on Protocol Specification, Testing, and Verification, 1988.
[RR] G.M. Reed & A.W. Roscoe, *A timed model for communicating sequential processes*, TCS 58, 1988, pp. 249-261.
[RH] A.W. Roscoe & C.A.R. Hoare, *The laws of Occam programming*, PRG Technical Monograph 53.