A **ProCoS II** Project Description: ESPRIT Basic Research project 7071

Jonathan Bowen *et al.*

Oxford University Computing Laboratory
Programming Research Group
11 Keble Road, OXFORD OX1 3QD

Tel: +44-865-273838 Fax: +44-865-273839
Email: procos-request@comlab.ox.ac.uk

Abstract

An overview of the current and planned activities of the ESPRIT Basic Research **ProCoS II** project (no. 7071) on "Provably Correct Systems" is presented. This is a follow-on project to **ProCoS** (no. 3104) previously announced in the Bulletin of the EATCS [3] and subsequently reported elsewhere (e.g., see [2, 4, 30]). A selected bibliography for both phases of the project is included.

1 Introduction

The ESPRIT Provably Correct Systems project is underway again. After discouraging delays and cuts, we have reformed a tightly focussed project with just four partners in 1992, dedicated to cover the fundamental technical aspects of a development process for critical embedded systems, from the original capture of requirements right down to the computers and special purpose hardware on which the programs run.

This breadth of scope is inspired by the brilliant work of Bob Boyer and J. Strother Moore at Computational Logic Inc. (CLInc) in Austin, Texas [16, 27]. The distinctive approach of the European effort is to emphasise:

1. A constructive approach to correctness, using proven transformations between specifications and designs and programs and compilers and hardware. Thus errors at this stage are avoided, so their absence never needs proving subsequently.

2. The use of a common abstract mathematical model to ensure global consistency across all the interfaces between design phases, notations, and technologies.

3. The inclusion of explicit parallelism and timing constraints throughout the development.

In these ways we hope to achieve an advance in technology in spite of reductions in research funding; furthermore, we hope to deliver theories and transformations which can be cheaply reused without specialist knowledge on an industrial scale for new projects and new products, as the need arises.

The partners in the project are as follows:

1. Oxford University Computing Laboratory, 11 Keble Road, Oxford, OX1 3QD, U.K. Responsible for coordination, political, financial and technical. Concentrates on a universal model to secure consistency of all the interfaces involved. Prof. Tony Hoare leads the site and the project, with the help of Jonathan Bowen, Stephen Brien, He Jifeng, Wayne Luk, Ian Page and Augusto Sampaio.

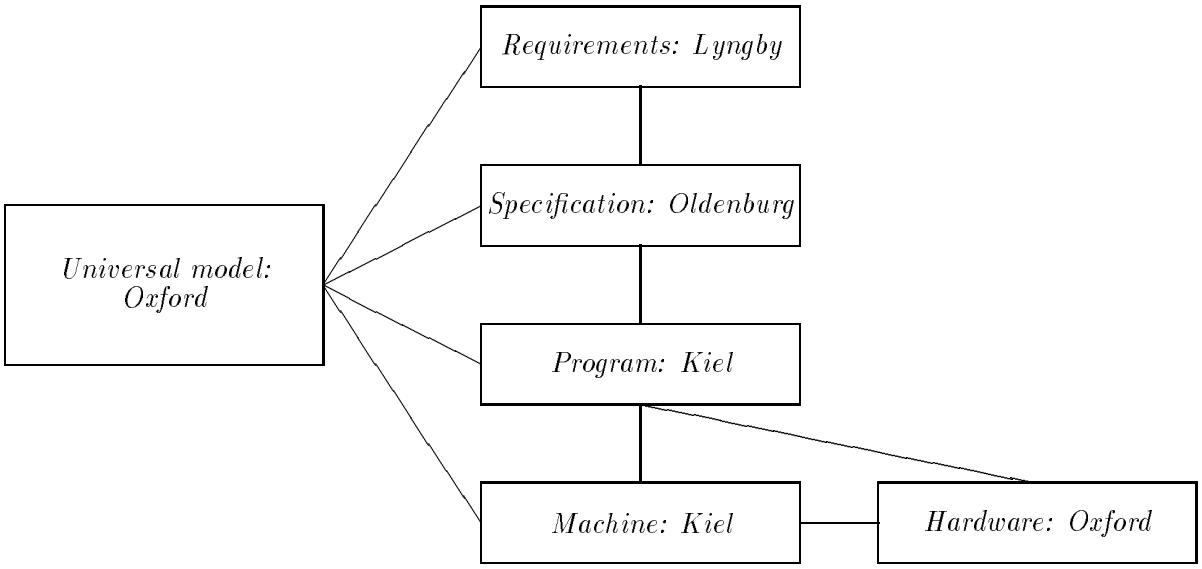
2. DTH, Department of Computer Science, Building 344, Technical University of Denmark, DK-2800 Lyngby, Denmark. Responsible for interface to reality, control engineering etc. Concentrates on capture and formalisation of total requirements, and the development of specifications for the computer-controlled components of the system. Anders P. Ravn is the site leader and is aided by Kirsten M. Hansen, Michael R. Hansen (currently visiting Oldenburg), Hans Henrik Løvengreen, Jens Nordahl, Hans Rischel, Jens U. Skakkebæk and E.V. Sørensen.

3. University of Oldenburg, FB10 - Informatik, Ammerländer Heerstraße 114–118, D-2900 Oldenburg, Germany. Responsible for production of correct programs. Concentrates on the design process from specification to the code of programs executed perhaps on multiple computers; plans mechanical checks and aids for this process. Prof. Ernst-Rüdiger Olderog is the site leader, with Stephan Rössig and Michael Schenke as team members.

4. Christian-Albrechts-Universität Kiel, Institut für Informatik und Praktische Mathematik, Preußerstraße 1–9, D-2300 Kiel 1, Germany. Responsible for production of correct machine code from timed high-level programs. Concentrates on the systematic development of a provably correct compiler. Prof. Hans Langmaack leads the site with Bettina Buth, Karl-Heinz Buth, Martin Fränzle, Burghard von Karger (currently visiting Oxford), Markus Müller-Olm and Ruben-Benjamin Reincke as members of the research group that do work for or related to the project.

Each site is responsible for progressing an agreed series of case studies, and for writing up a self-contained textbook for experts in the relevant area. In addition, there is a great deal of associated work not funded by the CEC. For example, Oxford hopes to explore the development of provably correct hardware compilers, targeting on Field Programmable Gate Arrays; and Lyngby are exploring theories to support reliability assessment.

Figure 1: Work part interdependencies and responsibilities



2 Work Plan

2.1 Technical Coordination – Oxford

Oxford is in charge of overall technical coordination and Figure 1 shows the main work parts of the project. The workplan for **ProCoS II** is structured around the research areas discussed in this section.

The project aims to develop a suite of techniques and mathematical theories which give a coherent approach to development of complex heterogeneous systems. Introduction of a common or universal model based on a Z calculus [28] would support the synthesis of a variety of design paradigms at the hardware, software and system levels. It starts with descriptions of all potential observable components of embedded safety-critical systems. A collection of system constructs will be defined on the subsets of observations, and their algebraic properties are then explored to give an algebraic semantics to a family of languages.

The universal model acts as a basis of the mathematical theory which connects various development activities; it has to support the coherent transition from requirement analysis down to hardware implementation. The universal model is also used to combine and coordinate those specific models adopted in different sites; each of which has to preserve the refinement ordering and the algebraic laws and must not introduce any extra observable components. As a result those models can be embedded into the universal model such that the satisfaction relation among two adjacent levels is consistent with the refinement order defined in the universal model.

Levels of interest to be modelled on this project, together with selected published

material produced so far during both phases of the project, include:

- Duration Calculus [18, 26, 33, 34, 37, 39, 41, 42]
- A design calculus [29, 31, 35]
- Compiling specifications [6, 10, 19, 21, 22, 23, 24]
- Hardware transformations [20, 32]

2.2 Requirements Engineering – Lyngby

Lyngby will contribute to the project within the following areas:

Specification and design of real-time systems. In cooperation with Oxford we plan to develop a system specification notation and calculus for real-time systems based on the Duration Calculus [34]. Its relationship to models and notations for dynamic systems and control systems will also be investigated [14, 33, 42], as well as its relation to program specifications as investigated in Oldenburg [9].

This line of work will also investigate design paradigms for real-time systems, and especially study the links to control theory. We also hope to find verification techniques that can utilise mechanised proof support or model checking along the lines of the timed automatons of Dill and Alur or timed transition systems of Sifakis [40].

Dependability. In collaboration with the ESPRIT PDCS project we intend to study dependability aspects of system specifications. We will strive to establish a formal interface between reliability assessment models and the models defined by the specifications [26, 38].

Case Studies. We plan to develop more ambitious case studies. These include a signal and switching system for a railway station on a trunk line [37]. An overview of a simple gas burner example, from requirements to hardware is given in [9].

2.3 Design – Oldenburg

The specific contribution of Oldenburg will be the formulation of a specification language SL^{time} for untimed and timed behaviour of communicating programs; and the development and correctness proof of compositional transformation rules for a design calculus. This will extend the work of Oldenburg in **ProCoS I** in the following directions:

Specification Language. The specification language SL of **ProCoS I** divides a specification into a trace part and a state part. This division is in principle closely related to ideas in UNITY [13], Back's action systems [1] and Lamport's TLA [25]. The state part corresponds to an iterative program or action system, and the trace part to a behavioural specification of that program.

The integration of time into the specification language encompasses the following aspects:

- The relations between the duration calculus and the specification language are explored. [17] describes the step from DC to switching circuits which are close to SL^{time}.
- In cooperation with Lyngby we shall evaluate case studies that indicate which timing properties are needed and how they can be expressed in different existing time calculi. One such study, a railway crossing, is presented in [36].
- Work at Oxford indicates that timing requirements can be classified into those stating lower bounds ("waits") and others stating upper bounds ("speed-ups"). We wish to investigate whether taking these two aspects separately leads to simpler calculi.

A first version of the timed specification language is proposed in [36].

Design Calculus. A proper integration of time and concurrency into a program design calculus is the most challenging task of Oldenburg. We plan to cope with this task as follows:

- A transformational approach to the design and verification of concurrent systems is being developed. [31] shows several ideas in this area along with a detailed case study.
- The design rules need to establish links between the discrete time calculus and the continuous real-time models and between global and local timing constraints.

Mechanical support. It appears that the transformational set-up is closely related to the structure of mechanical reasoning performed in meta-logical systems like LCF, HOL or LAMBDA. We shall explore this relationship and work on the mechanisation of the transformational approach to the design of Occam programs.

We are using the LAMBDA system of Abstract Hardware Ltd. to implement a transformational approach. Initially these activities will be performed within a national "BMFT" research project, called "KORSO" (for "Correct Software"). In KORSO we are implementing a semantics model in LAMBDA which allows the verification of transformation rules and their mechanical application for the design of Occam-like programs from specifications [5]. Later we plan to take over this work to **ProCoS II** for (subsets of) the specification language and transformation rules used there.

2.4 Compilation – Kiel

In **ProCoS I**, it was demonstrated, how a compiler for a very simple language could be specified and implemented in a provably correct manner. In **ProCoS II** we concentrate

on compiler development for more powerful languages, including real-time and parallelism. The work centered on program compilation is essentially done at Kiel University. It is divided into the following parts:

Programming language design. PL is an imperative language with parallel composition and communication to reflect concurrency in the controlled systems. Since treatment of real-time constraints for reactive systems is the overall goal of **ProCoS II**, PL provides the means to express assertions on the timed behaviour of programs. It features delay timing as present in Occam and allows to specify upper bounds for the time spent for the execution of internal actions. A proposal for the core of PL can be found in [15].

Machine language design. As target language for the compilation, a suitable machine language is selected. Our starting point is the Transputer instruction set. More abstract levels of machine language might be investigated to ease theoretical considerations. But the target language of the compiler will be the machine language of an existing processor in order to be able to execute generated code on real hardware. The semantics for the machine language will be formalised in an appropriate style to make correctness proofs for compiler specifications feasible. An important aspect in the design of the semantics will be the formalisation of time at the hardware level.

Compiler design. We will develop a compiler for the project language, to enable the full demonstration of the chosen case studies. The development of this example compiler will not necessarily be fully proven. Suitable tools will be used to obtain a prototype compiler as early as possible to support the full investigation of the case studies. As far as possible in the framework of the project, the prototype compiler will be based on proven correct specifications. On a more loose basis we will investigate global methods for compiler design which lead to a high degree of confidence in their correctness.

Verification support. Mechanical verification support is desirable for most correctness proofs arising in compiler development. The emphasis of **ProCoS II** is on timing. The main effort in proofs for compiler correctness concerns the correctness of the code-generator specification. Since this essentially depends on the semantic model, a proof support system is only suitable, if the model can be easily formalised in the system. We will investigate appropriate tools concerning their applicability. [11, 12] describe how transition systems that are used for structural operational semantics definitions can be embedded into term rewriting systems in general and the Larch-based LP proof system in particular.

2.5 Hardware – Oxford

To manage the complexity of large-scale computer systems being developed by industry, a series or hierarchy of specifications is produced, each containing progressively more detail. This is known as refinement. Refinement needs both guidelines on how to proceed from a

high level to a low level specification, and rules for verifying that this has been done in a consistent manner. For hardware design we identify the following major activities:

- Elaborate and coordinate the existing design paradigms.
- Develop a design calculus for sequential (synchronous and perhaps asynchronous) circuits in the framework of CSP and Duration Calculus.
- Facilitate algebraic transformation at various stages of hardware implementation, and study the potential use of term rewriting systems and other tools in hardware design.
- Rephrase the application of refinement calculus on transformations between different levels of hardware design (e.g., instruction level, register level, gate level, switching circuit level and analog level).
- Build the link with other available technologies and design calculi (e.g., Huffman and Zissos's Sequential Equations, Interface State Graph, Martin's Compilation and Ebergen's Regular Expression).

Activity in this area will be dependent on projects with other sources of funding. Initial results are reported in [8, 9, 20]. This will continue the work of the currently abutted UK collaborative safemos project at Oxford [7].

3 Working Group

An integral part of our research plan is the formation of a Working Group of potential collaborators and industrial partners. We have recently submitted a proposal to the CEC. We also intend to undertake the organisation of newsletters, distribution lists, seminars, technical meetings, workshops and an open conference at which the directions of research can be discussed and the results can be disseminated. We have planned the following dates:

- 1993 September 20–23, Technical Meeting, Oldenburg (D)
- 1994 January, Workshop, Copenhagen (DK)
- 1994 September, Open Conference, Kiel (D)
- 1995 January, Workshop (UK?)
- 1995 August, Technical Meeting (DK?)

The open conference may be held in conjunction with an existing related conference series. If anyone would like to attend on these occasions or to know more about them, please contact the **ProCoS II** project coordinator, Oxford University, to be added to our mailing list and to obtain further information.

References

[1] R.J.R. Back. Refinement calculus, part II: Parallel and reactive programs. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems – Models, Formalisms, Correctness*, volume 430 of *Lecture Notes in Computer Science*, pages 67–93. Springer-Verlag, 1990.

[2] D. Bjørner. Trusted computing systems. In *Proc. 14th International Conference on Software Engineering (ICSE), Melbourne, Australia.* North-Holland, May 1992.

[3] D. Bjørner et al. A ProCoS project description: ESPRIT BRA 3104. *Bulletin of the EATCS*, 39:60–73, 1989.

[4] D. Bjørner, H. Langmaack, and C.A.R. Hoare. ProCoS I final deliverable. ProCoS Technical Report [ID/DTH DB 13/1], Department of Computer Science, Technical University of Denmark, DK-2800 Lyngby, Denmark, January 1993.

[5] J. Bohn. Interaktive synthese kommunizierender systeme mit LAMBDA. Bericht, Carl-von-Ossietzky-Universität Oldenburg, Germany, February 1993.

[6] J.P. Bowen. From programs to object code using logic and logic programming. In R. Giegerich and S.L. Graham, editors, *Code Generation – Concepts, Tools, Techniques*, Workshops in Computing, pages 173–192. Springer-Verlag, 1992. Proc. International Workshop on Code Generation, Dagstuhl, Germany, 20–24 May 1991.

[7] J.P. Bowen, editor. *Towards Verified Systems.* Real-Time Safety Critical Systems Series. Elsevier, 1993. In preparation.

[8] J.P. Bowen, B. Buth, He Jifeng, M. Müller-Olm, E.-R. Olderog, and A.P. Ravn. Provably correct systems: Tutorial material, Formal Methods Europe 93. ProCoS Technical Report [ID/DTH APR 20/1], Department of Computer Science, Technical University of Denmark, DK-2800 Lyngby, Denmark, March 1993.

[9] J.P. Bowen, M. Fränzle, E.-R. Olderog, and A.P. Ravn. Developing correct systems. In *Proc. 5th Euromicro Workshop on Real-Time Systems, Oulu, Finland.* IEEE Computer Society Press, 22–24 June 1993. To appear.

[10] B. Buth, K.-H. Buth, M. Fränzle, B. von Karger, Y. Lakhneche, H. Langmaack, and M. Müller-Olm. Provably correct compiler development and implementation. In *Compiler Construction '92, Proc. 4th International Conference (CC'92), Paderborn, Germany*, volume 641 of *Lecture Notes in Computer Science*, pages 141–155. Springer-Verlag, October 1992.

[11] K.-H. Buth. Simulation of transition systems with term rewriting systems. Bericht 9212, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, Germany, 1992.

[12] K.-H. Buth. Using SOS definitions in term rewriting proofs. In Ursula Martin and Jeannette Wing, editors, *Proc. First International Workshop on Larch*, Workshops in Computing. Springer-Verlag, 1992. Full version available as Bericht 9214, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, Germany.

[13] K.M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, 1988.

[14] M. Engel et al. A formal approach to computer systems requirements documentation. In *Hybrid Systems*, Lecture Notes in Computer Science. Springer-Verlag, 1993. To appear.

[15] M. Fränzle. Proposal for a programming language core for ProCoS II. ProCoS Technical Report [Kiel MF 11/2], Christian-Albrechts-Universität Kiel, Germany, March 1993.

[16] D.I. Good and W.D. Young. Mathematical methods for digital system development. In S. Prehn and W.J. Toetenel, editors, *VDM'91, Formal Software Development Methods, Volume 2: Tutorials*, volume 552 of *Lecture Notes in Computer Science*, pages 406–430. Springer-Verlag, 1991.

[17] M.R. Hansen and E.-R. Olderog. Constructing circuits from decidable Duration Calculus. Bericht, Carl-von-Ossietzky-Universität Oldenburg, Germany, April 1993.

[18] M.R. Hansen and Zhou Chaochen. Semantics and completeness of Duration Calculus. In W-P. de Roever, editor, *Proc. REX'91, Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[19] He Jifeng and J.P. Bowen. Time interval semantics and implementation of a real-time programming language. In *Proc. 4th Euromicro Workshop on Real-Time Systems*, pages 110–115. IEEE Press, June 1992.

[20] He Jifeng, I. Page, and J.P. Bowen. Towards a provably correct hardware implementation of Occam. In G.J. Milne, editor, *Proc. IFIP WG10.2 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'93), Arles, France*, Lecture Notes in Computer Science. Springer-Verlag, 24–26 May 1993. To appear.

[21] C.A.R. Hoare. Refinement algebra proves correctness of compiling specifications. In C.C. Morgan and J.C.P. Woodcock, editors, *3rd Refinement Workshop*, Workshops in Computing, pages 33–48. Springer-Verlag, 1991.

[22] C.A.R. Hoare and He Jifeng. Refinement algebra proves correctness of a compiler. In M. Broy, editor, *Programming and Mathematical Method: International Summer School directed by F.L. Bauer, M. Broy, E.W. Dijkstra, C.A.R. Hoare*, volume 88 of *NATO ASI Series F: Computer and Systems Sciences*, pages 245–269. Springer-Verlag, 1992.

[23] C.A.R. Hoare, He Jifeng, J.P. Bowen, and P.K. Pandya. An algebraic approach to verifiable compiling specification and prototyping of the ProCoS level 0 programming language. In CEC DG XIII, editor, *ESPRIT'90 Conference Proceedings, Brussels*, pages 804–818, 1990.

[24] C.A.R. Hoare, He Jifeng, and A.C.A. Sampaio. Normal form approach to compiler design. *Acta Informatica*, to appear.

[25] L. Lamport. The temporal logic of actions. Technical Report 79, Digital Systems Research Center, 130 Lytton Avenue, Palo Alto, California 94301, USA, 25 December 1991.

[26] Z. Liu, A.P. Ravn, E.V. Sørensen, and Zhou Chaochen. A probabilistic Duration Calculus. In *Proc. 2nd Int. Workshop on Responsive Computing Systems, Tokyo, Japan*. KDD R & D Laboratories, October 1992.

[27] J.S. Moore et al. Special issue on system verification. *Journal of Automated Reasoning*, 5(4):409–530, December 1989.

[28] J.E. Nicholls, S.M. Brien, et al. Z base standard. ZIP Project Technical Report ZIP/PRG/92/121, SRC Document: 132, Version 1.0, Oxford University Computing Laboratory, 11 Keble Road, Oxford OX1 3QD, UK, 30 November 1992.

[29] E-R. Olderog. Towards a design calculus for communicating programs. In J.C.M. Baeten and J.F. Groote, editors, *Proc. CONCUR'91*, volume 527 of *Lecture Notes in Computer Science*, pages 61–72. Springer-Verlag, 1991.

[30] E-R. Olderog. Interfaces between languages for communicating systems. In W. Kuich, editor, *Automata, Languages and Programming: Proc. 19th International Colloquium (ICALP), Wien, Austria, July 1992*, volume 623 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[31] E.-R. Olderog and S. Rössig. A case study in transformational design of concurrent systems. In M.-C. Gaudel and J.-P. Jouannaud, editors, *Proc. TAPSOFT'93*, volume 668 of *Lecture Notes in Computer Science*, pages 90–104. Springer-Verlag, 1993.

[32] I. Page and W. Luk. Compiling Occam into field-programmable gate arrays. In W. Moore and W. Luk, editors, *FPGAs, Oxford Workshop on Field Programmable Logic and Applications*, pages 271–283, 15 Harcourt Way, Abingdon OX14 1NV, UK, 1991. Abingdon EE&CS Books.

[33] A.P. Ravn and H. Rischel. Requirements capture for embedded real-time systems. In *Proc. IMACS-IFAC Symposium on Modelling and Control of Technological Systems (MCTS'91)*, volume 2, pages 147–152, 1991.

[34] A.P. Ravn, H. Rischel, and K.M. Hansen. Specifying and verifying requirements of real-time systems. *IEEE Transactions on Software Engineering*, SE-19(1):41–55, January 1993.

[35] S. Rössig and M. Schenke. Towards a design calculus for communicating programs. In S. Prehn and W.J. Toetenel, editors, *VDM'91 Formal Software Development Methods*, volume 551 of *Lecture Notes in Computer Science*, pages 148–163. Springer-Verlag, 1991.

[36] M. Schenke. A timed specification language for concurrent reactive systems. ProCoS Technical Report Oldenburg MS 6, Carl-von-Ossietzky-Universität Oldenburg, Germany, March 1993.

[37] J.U. Skakkebæk, A.P. Ravn, H. Rischel, and Zhou Chaochen. Specification of embedded real-time systems. In *Proc. 4th Euromicro Workshop on Real-Time Systems*, pages 116–121. IEEE Press, June 1992.

[38] E.V. Sørensen, J. Nordahl, and N.H. Hansen. From CSP models to Markov models. *IEEE Transactions on Software Engineering*, to appear.

[39] Zhou ChaoChen, M.R. Hansen, A.P. Ravn, and H. Rischel. Duration specifications for shared processors. In J. Vytopil, editor, *Proc. Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 571 of *Lecture Notes in Computer Science*, pages 21–32. Springer-Verlag, 1991.

[40] Zhou Chaochen, M.R. Hansen, and P. Sestoft. Decidability and undecidability results for Duration Calculus. In *STACS '93: 10th Symposium on Theoretical Aspects of Computer Science, Würzburg, Germany, February 1993*, Lecture Notes in Computer Science. Springer-Verlag, 1993. To appear.

[41] Zhou ChaoChen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.

[42] Zhou Chaochen, A.P. Ravn, and M.R. Hansen. An extended Duration Calculus for hybrid systems. In *Hybrid Systems*, Lecture Notes in Computer Science. Springer-Verlag, 1993. To appear.