CONFER-2

CONcurrency and Functions:

Evaluation and Reduction

Working Group

Project Number: 21836

Periodic Progress Report
December, 1997

Contents

Chapter 1

Overview

This report contains the First Periodic Progress Report for ESPRIT WG Nr. 21836 (CONFER-2).

The report contains 4 main parts: management at consortium level and at each site, deliverables (programs of each CONFER-2 workshops 1 and 2, software deliverables), a progress report describing the technical work achieved during the first year, and appendices listing CONFER-2 publications.

Further information may be requested from the coordinator:

Jean-Jacques Lévy
INRIA, Rocquencourt
bat.8, Domaine de Voluceau
78153–Le Chesnay, Cedex
France
tel: +33-1-39-63-56-89
fax: +33-1-39-63-53-30
e-mail: `Jean-Jacques.Levy@inria.fr`

This document has been compiled from input from all of the partners in the CONFER project: Andrea Asperti, Peter Sewell, Jan Willem Klop, Samson Abramsky and Paul-André Mellies, Pierre-Louis Curien, Jean-Francois Monin, Bent Thomsen and Lone Leth, Davide Sangiorgi, Bjorn Lisper, Gianluigi Ferrari, Matthew Hennessy, and David Walker. Lone Leth and Bent Thomsen from ICL greatly helped in the writing and the assembly of the document.

Chapter 2

Executive summary

The Confer-2 Working Group follows Basic Research Action CONFER. It is a framework to keep together the CONFER community, and to provide 2 annual workshops.

The overall objective of Confer-2 action is similar to the one of CONFER and to create both the theoretical foundations and the technology for combining the expressive power of the functional and the concurrent computational models. The action is organised around four main areas:

- Foundational Models and Abstract Machines
- Calculi
- Logics for Concurrency and the λ-calculus
- Programming Languages

(see Periodic Progress Reports of CONFER, for more explanations). In this first year of Confer-2, we may notice the following important and successful facts.

In the area of Foundational Models and Abstract Machines, progress has been done in three areas, there has been a major result by Asperti and Mairson on the non elementary-recursive cost of optimal reductions in the lambda calculus. The work on action calculi has been continued which now provide a uniform setting for dealing with rewrite rules on terms with bound variables. Work on tile models, axiomatic rewrite rules, the sequentiality theorem for the lambda calculus and syntactic transformation of recursive program schemes with non determinism is also part of this section.

In the area of Calculi work has been done at many levels. First, the new *blue calculus* of Boudol gives a nice language with both the functional and concurrent paradigms. The update-calculus of Parrow and Victor simplifies and extends the π-calculus, making it suitable also for models with shared states and concurrent constraints. Many results have been obtained to develop new theories for the equivalences of processes, to model concurrent objects or secure communication, or to prove transformations of programs with concurrency.

In the area of Logics for Concurrency and the λ-calculus, work has proceeded on the development of games semantics for the sequential lambda calculus, linear logic or concurrent processes. Denotational semantics has also been used to give semantics of *core Facile*.

In the area of Programming Languages several efforts have been completed: a stabilised public release of the PICT language, a first public release of the Join Calculus language, a distributed garbage collector collecting cycles, the design of Linda with localities, an efficient automata-based verification environment for the π-calculus. Several experiments are also led to investigate the real uses of mobility and agents in programming examples.

As for BRA CONFER, the progress reports in chapter 5 reflect that crossfertilisation of ideas among fellow researchers in the working group is very lively. As in previous years it is evident that related work done at other sites in the consortium is referenced often and also very often used as the basis for development of new results. It is no surprise that a number of these results have been achieved in collaboration between sites, enabled through site visits for shorter or longer periods. Leading researchers outside the consortium have also been involved in fruitful collaborations.

During Year 1 of Confer-2 two workshops have been held.

The working group has produced a large number (about 70) of reports and several of these have been published at conferences, international workshops and journals. Many of theses works are accessible at URL http://www.cs.auc.dk/mobility/, where a Web page on Mobility is maintained by Uwe Nestmann, who moved in September 1997 from INRIA Rocquencourt to Aalborg.

Several Ph.D.'s are in preparation in the action, and several have been defended.

Chapter 3

Management

3.1 Consortium level

The main management of the project is done at INRIA, Rocquencourt, and at ICL.

The following activities at the consortium level have taken place during the first year of the CONFER-2 working group:

Two workshops have been held. The first took place in March 1997 at University of Bologna with 20 presentations and 30 participants, and was organised by Andrea Asperti. The second workshop took place in October 1997 at CWI in Amsterdam with 14 presentations and 30 participants. It was organised by Inge Bethke and J.-W. Klop.

As specified in §2.6 of the Working Group Programme, the vast majority of the activity of CONFER-2 is devoted to the organisation of 2 workshops per year. The cost statements forms will strengthen this distribution of the activity, although some of the funds are still reserved for travelling expenses between sites.

As a general remark on the technical orientation of the Working Group, it should be noticed that the subject of area 3 in the Progress section has a bit changed with respect to the initial programme. The title remains unchanged, but the area covers more semantics foundations of the various syntactic calculi considered in the two first parts of the programme: foundational models and calculi. The third area is now more oriented towards game semantics and denotational semantics. However, the overall project keeps its very syntactic initial flavour. The aim of the Working Group is also still to consider the fusion of the functional and concurrent approach.

3.2 Università di Bologna

3.2.1 Research

The research activity in Bologna, organized along the main task of the Working Group, has been the following:

Calculi In this area, we have been involved both in the static semantics (polymorphic type theory) and the dynamic one (analysis of bisimulation) of concurrent calculi.

Polymorphic Type Theory We have studied the adaptation of the Damas-Milner typing discipline to the join-calculus [FLMR97]. Since interactions in the join-calculus are explicitly defined, it turns out that polymorphic types may be effectively inferred as in the lambda-calculus with the let-operator. Our main result is a new generalization criterion that extends the polymorphism of ML to join-calculus definitions. We prove the correctness of our typing rules with regard to a chemical semantics and we also relate typed extensions of the join-calculus to functional languages.

Analysis of Bisimulation We have mainly investigated the theory of bisimulation in the join-calculus [BFL97]. To this purpose, we introduce a refined operational model that makes interactions with the environment explicit. In this framework we show that the so-called "*locality property*" strongly affects the treatment of bisimulation, and provides a semantics simpler than other proposals in the literature. We also propose several formulations of bisimulation and we establish that all formulations yield the same equivalence, which is finer than barbed congruence. Bisimulation and barbed congruence coincide in the presence of name-testing.

Logics for Concurrency λ-calculus Our research in this field has been mainly focused on the theory of optimal sharing in β-reduction. In particular, we proved that the notion of optimal "parallel" reduction, as formalized by Lèvy, is not *elementary recursive* [AM98]. The result should not be understood in a negative sense: it just says that the notion of optimal sharing *cannot have* a simple implementation since, essentially, *all the computational work* can be done via sharing. Actually, the result is straightforward consequence of the fact that every term of the simply typed lambda calculus can be essentially reduced in a linear number of "parallel" (i.e. sharable) β-reduction.

Programming Languages We proceeded in two directions, here:

BOHM We are still working on our implementation of Lamping's graph reduction algorithm for optimal reduction of functional languages (the Bologna Optimal Higher-order Machine [AG98]). We have almost completed a new "light" version, essentially inspired by Girard's Light Linear Logic. The *light* version provides a much sharper syntactical control over the shape of sharing graphs than that provided by the rough encoding of intuitionistic implication into linear logic, with a significant impact on the efficiency of the reduction.

Join-calculus We have a first prototype of a C implementation of the distributed join-calculus. This has been mostly conceived as an implementation exercise, and some features are still missing (especially in the type system: no type inference, no polymorphism, no recursive types, . . .). According to our (very preliminary) tests, the performance looks comparable with the implementation developed at INRIA (but several optimizations are still missing). We expect to make a first release for the beginning of 1998.

3.2.2 Workshops, Travels and Visits

Workshops Bologna organized the first Workshop of the Working Group, from march 24 to march 26. We had more than 30 participants, and 20 talks. For more informations on the Workshop in Bologna, look at "http://www.cs.unib.it/ asperti/CONFER/bologna.html".
Please remark that we got the first money for the WG about two months *after* the Workshop.

Travels Cosimo Laneve participated at the Second Confer Workshop held in Amsterdam from 20 to 22 October 1997.

Visits Cédric Fournet visited the University of Bologna from July 29 to 2 August 1997. He worked with Cosimo Laneve on the issue of "Bisimulations in the Join-Calculus".

3.2.3 Publications

[AM98] A. Asperti, H.Mairson *Optimal β-reduction is not elementary recursive.* Proc. of the twenty-fifth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'98).

[AG98] A. Asperti, S. Guerrini *The Optimal Implementation of Functional Programming Languages.* To appear in the *"Cambridge Tracts in Theoretical Computer Science"* Series, Cambridge University Press, 1998.

[BFL97] M. Boreale, C. Fournet, C. Laneve: *Bisimulations in the Join-Calculus.* October 1997. Submitted.

[FLMR97] C. Fournet, C. Laneve, L. Maranget, D. Rèmy: *Implicit Typing à la ML in the Join-Calculus.* In Proceeding CONCUR '97, LNCS 1243.

3.3 University of Cambridge

3.3.1 Research directions

The research at Cambridge that is most relevant to CONFER 2 has been focused on the dynamics of action calculi, on the structure of higher-order and reflexive action calculi, and on the foundations of concurrent and distributed programming languages.

Action Calculi

- **Labelled transition systems and bisimulation congruences** The semantics of a process calculus, several of which have been introduced during CONFER, can often be most intuitively given as a reduction relation, defined by a structural congruence and reduction axioms. The action calculus framework embraces a rather wide class of such definitions. A central problem is then that of deriving, from such a definition of reduction, a labelled transition semantics for which bisimulation is a satisfactory congruence. Considerable progress has been made, e.g. by Jensen [Jen98] for a form of graph rewriting approximating to a graphical presentation of action calculi, and by Sewell for special cases of term rewriting. Work on more general cases is being undertaken by Gardner, Jensen, Leifer, Milner and Sewell.

- **Expressiveness** A uniform framework for the operational semantics of calculi allows comparisons of expressiveness to be made. Using natural homomorphisms between control structures (the models of action calculi), it is possible to characterise important properties of calculi such as *mobility of channels*. Leifer has shown that the π-calculus is mobile in this sense, whilst many other calculi such as CCS, the λ-calculus, and Petri nets, are not. This work, extending the work of Mifsud at Edinburgh, gives a precise meaning to the intuitive sense in which the π-calculus is more expressive than, say, CCS.

- Gardner has introduced a type-theoretic description of action calculi [GH97] which develops her previous work on a name-free account of action calculi and gives a logical interpretation based on the general binding operators studied by Aczel. With her Ph.D student Hasegawa, at Edinburgh, she has shown that the type-theoretic account of higher-order action calculi corresponds to Moggi's commutative computational λ-calculus. With Gordon Plotkin and Andrew Barber, also at Edinburgh, they have introduced a further extension of action calculi, called the *linear* action calculi [BGHP97]. Linear action calculi extend the higher-order case, and the associated type theory corresponds to the linear type theories studied by Barber and Nick Benton of Cambridge. The conservative extension results follow from looking at the corresponding categorical models.

- **Examples** Several example action calculi have been considered, including a functional programming language with mutable store and exceptions, the ambient calculus of Abadi and Cardelli, and distributed process calculi.

- **Higher-order reflexive action calculi** The first of these examples required two previously separate enrichments of AC, namely, the higher-order and the reflexive extensions (the latter in a simplified form). The proof that an AC combining the two is well-defined reduces to a proof of strong normalisation for the higher-order beta-reduction of the molecular forms corresponding to this enriched AC. Leifer completed this proof and, in the process of doing so, developed several new techniques for handling molecular forms, including the formalisation of manipulations involving reflexive substitutions.

Concurrent and distributed programming languages Our work on programming languages builds on the PICT language, based on the π-calculus, of Pierce and Turner. PICT now has a stable public release; its design is described in [PT97]. Pierce, who has moved from Cambridge to Indiana, has continued to work on type inference and (with Sangiorgi of INRIA-Sophia-Antipolis) on bisimulation in the presence of polymorphism. We are collaborating on calculi for distribution.

- **Observational semantics for concurrent programming languages** Sewell has studied the semantics of PICT in [Sew97c], and has shown how the behaviour of implementations of the language relates precisely to the labelled transitions of the π-calculus. His results indicate that there is an essentially unique behavioural equivalence appropriate for PICT, from the large space of choices in the literature.
- **Global/Local typing** In the design of distributed programming languages there is a tension between the implementation cost and the expressiveness of the communication mechanisms provided. In [Sew97a] a static type system for a distributed π-calculus has been introduced, in which the input and output capabilities of channels may be either global or local. This allows compile-time optimization where possible but retains the expressiveness of channel communication. Subtyping allows all communications to be invoked uniformly. The distributed π-calculus used integrates location and migration primitives from the Distributed Join Calculus with asynchronous π communication, with a simple reduction semantics that can be presented as an action calculus. Abstract machines for similar calculi have been considered by Unyapoth.
- **Location Independence** In [SWP97] we have begun to study in more detail the design, semantic definition and implementation of communication primitives by which mobile agents can interact. We have proposed a simple calculus of agents that can migrate between sites and can interact by both location dependent and location independent message passing. It has a precise reduction semantics. Implementations of the location independent primitives can be expressed as encodings, or compilations, of the whole calculus into the fragment with only location dependent communication. This allows the algorithms to be stated and understood precisely, a clean implementation strategy for prototype languages (such an encoding can be realized as one phase of a compiler), and reasoning. It

also provides a setting in which multiple communication mechanisms can coexist gracefully.

A prototype implementation, building on that of PICT, is ongoing, as is work on reasoning, in particular on correctness and robustness results.

Work has also been completed on the algebraic theory of processes [Sew97b] and on closed action calculi [Gar98].

3.3.2 Personnel and exchanges

Researchers and PhD students associated with CONFER at Cambridge include Adriana Compagnoni, Philippa Gardner, Ole Jensen, James Leifer, Robin Milner, Paola Quaglia, Peter Sewell, Asis Unyapoth and Paweł Wojciechowski.

During 96–97 Gardner and Sewell have given seminars at Sussex, and Sewell has visited Edinburgh. In November 97 Sewell spent a week with the INRIA-Rocquencourt group in discussions of distributed calculi and of labelled transition systems and bisimulation for arbitrary rewriting systems.

3.3.3 Publications

[BGHP97] Andrew Barber, Philippa Gardner, Masahito Hasegawa, and Gordon Plotkin. From action calculi to linear logic,. In *Proceedings of CSL 97, Aarhus*, 1997.

[Gar98] Philippa Gardner. Closed action calculi. *Theoretical Computer Science*, 1998. To appear.

[GH97] Philippa Gardner and Masahito Hasegawa. Types and models in higher-order action calculi. In *Proceedings of TACS 97, Sendai, Japan*, 1997.

[Jen98] Ole Høgh Jensen. PhD thesis, University of Cambridge. Forthcoming, 1998.

[PT97] Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. Technical Report CSCI 476, Computer Science Department, Indiana University, 1997. To appear in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, MIT Press.

[Sew97a] Peter Sewell. Global/local subtyping for a distributed π-calculus. Technical Report 435, Computer Laboratory, University of Cambridge, August 1997. Available from `http://www.cl.cam.ac.uk/users/pes20/`.

[Sew97b] Peter Sewell. Nonaxiomatisability of equivalences over finite state processes. *Annals of Pure and Applied Logic*, 1997. To appear.

[Sew97c] Peter Sewell. On implementations and semantics of a concurrent programming language. In *Proceedings of CONCUR '97. LNCS 1243*, pages 391–405. Springer-Verlag, 1997.

[SWP97] Peter Sewell, Paweł T. Wojciechowski, and Benjamin C. Pierce. Location independence for mobile agents. Submitted for publication, 1997.

3.4 CWI

3.4.1 Research Directions

J.W. Klop and I. Bethke have continued work on a generalization of Berry's Sequentiality Theorem for lambda calculus. Using the setting of infinitary lambda calculus where Bohm trees are infinite normal forms, the technique of origin tracking was employed to obtain the sequentiality property for lambda calculus. The infinitary lambda calculus differentiates in three versions according to the way syntactic depth is counted: the three versions yield ordinary lambda calculus with Bohm trees, the lazy lambda calculus with Levy-longo trees, and a version introduced by Berarducci. For all three versions Berry's sequentiality theorem is uniformly proved by the same arguments. Also, a strengthening of Berry's theorem was obtained that can be used to prove more undefinability results.

3.4.2 Participation to workshops

- I. Bethke and J.W. Klop have attended the Confer workshop in Bologna from March 24-26, 1997. J.W. Klop gave a talk: Infinitary lambda calculus and Berry's sequentiality theorem.
- I. Bethke and J.W. Klop attended the second Confer workshop in Amsterdam, from October 20-22, 1997. I. Bethke gave a talk: Sequentiality in lambda calculus.

3.5 University of Edinburgh

3.5.1 Research directions

Abramsky has developed a concurrent version of game semantics, using the Kahn-Plotkin theory of concrete domains and sequential functions. This has been used to give a game semantics for Classical Linear Logic. It has also been used to give a model for logic with branching quantifiers and information hiding, as proposed by Hintikka. This work was presented in invited talks given at the Workshop on "Problems and Advance in the Semantics of Linear Logic" held in Utrecht, November 28th and 29th 1997, and at the Amsterdam Colloquium, December 17-20 1997. Honda and Yoshida developed a version of game semantics for call-by-value PCF, and proved a full abstraction result. Their description of their model used some tools from the pi-calculus. Abramsky and McCusker gave a more abstract and general construction of call-by-value models from call-by-name ones, and showed that when applied to the call-by-name games model of PCF it gave rise to the call-by-value one of Honda and Yoshida; when applied to the model for the richer functional language FPC studied by McCusker in his thesis, it gave a fully abstract model for call-by-value FPC; and when applied to the model previously given by them for Idealized Algol, it gave a full abstract model for a fragment of ML with ground reference types.

Honda and Yoshida's paper appeared in ICALP '97, while Abramsky and McCusker's paper was given in CSL '97, and will appear in the proceedings of that conference. Subsequently, Abramsky, Honda and McCusker have obtained a fully abstract games model for general reference types; a paper containing this result has been submitted to LiCS '98.

A factorisation theorem in the sense of Freyd & Kelly has been obtained by Melliès in the framework of axiomatic rewriting systems. The theorem means that the efficient part of a computation can always be separated from its useless part modulo redex permutations, and that this separation is categorically functorial. This work was presented at the Confer workshop in Bologna and at the CTCS '97 conference in Santa Margherita.

Subsequently, Melliès has obtained a stability result in the sense of Berry expressing the existence of a cone $(e_i : M \to V_i)_{i \in I}$ of minimal computations from any term M to the set of its values V_i. The result whose proof relies heavily on the factorisation theorem is verified by any axiomatic rewriting system. In the lambda-calculus, it extends the result by Wadsworth that the head-reduction is the *minimal* reduction path from a lambda-term to its head-normal form. In the call-by-value lambda-calculus, the cone $(e_i : M \to V_i)_{i \in I}$ contains at most a singleton which describes the strategy of Landin SECD machine. In the lambda-sigma calculus, the existence of a cone and its finiteness implies the completeness of various weak strategies and the corresponding abstract machines. A paper containing these results has been submitted to LiCS '98.

3.5.2 Workshops, Travels and Visits

Travels Paul-André Melliès participated at the Confer Workshops held in Bologna and

Amsterdam.

3.5.3 Publications

[AM97] S. Abramsky, G. McCusker, *Call-by-Value games (extended abstract)* To appear in Proc. of CSL'97.

[HY97] K. Honda, N. Yoshida, *Game-theoretic analysis of Call-by-Value Computation (extended abstract).* To appear in Proc. of ICALP'97.

[M97] P.-A. Melliès, *A factorisation theorem in Rewriting Theory.* Proc. of the Category Theory and Computer Science symposium (CTCS'97).

3.6 ENS

3.6.1 Research Directions

At ENS, work has progressed in game semantics and in the investigation of Boudol's calculus of resources.

Game semantics. A key interest of game semantics is that it provides a framework within which one may conciliate dynamic and static aspects of meaning. It is expected that such semantics can help in complexity theory on the one hand and in extending denotational semantics to model concurrent and distributed languages (so-called process-calculi).

1. In [DHR96], V. Danos, H. Herbelin and L. Regnier explain the operational side of the two game models of Hyland-Ong (HO) and Abramsky-Jagadeesan-Malacaria (AJM) in terms of abstract machines. Building on this explanation, an embedding of the AJM model in the HO model is given that leads to aneasy proof of completeness for the former.

2. In [BDER97], P. Baillot, V. Danos, T. Ehrhard and L. Regnier adapt the AJM construction to give the first known games model which encompasses all of classical linear logic. Composition of strategies is recast as a communication process. The exponential part of the model is based on the idea of sticking to strategies which contain all possible encodings of copy-tags. This works is pursued in [BDER97b], which gives an explicit transformation that goes from that linear logic games model to the traditional relational model. This concrete extensional collapse helps in understanding what games semantics is good to.

3. In [CuBoeh] P.-L. Curien presents a formalism of trees with pointers, called abstract Böhm trees, that provide a suitable abstract framework in which various cut-free proofs or normal terms of several λ-calculus based languages (including PCF and Parigot's $\lambda\mu$-calculus) can be faithfully encoded. A simple abstract machine allows us to compute over abstract Böhm trees. This machine is closely related to Coquand's interaction sequences and debates. The execution over finite abstract Böhm trees always terminates. We introduce an abstract notion of type that fits the purpose of guaranteeing that the execution cannot go into deadlock, i.e., that it always reaches a satisfactory final state. The operational aspects of (untyped) Böhm trees are investigated in more depth in [CuHer96], a joint paper by P.-L. Curien and H. Herbelin,

Calculus with resources (λ_r). C. Lavatelli has been working on the definition of an algebraic semantics of λ_r based on approximants, in the flat scenario (where both abstractions and deadlock are observable), which turns out to be sound. Work in progress concerns completeness. The aim is to use these results as intermediate steps in the characterization of the flat semantics through an intersection type system. Another work in progress, by C. Lavatelli and P.-L. Curien, concerns the use of techniques of logical relations to compare the inequational theories of two models of λ_r that have arisen naturally when trying to give a semantics to λ_r.

3.6.2 Publications

[BDER97] P. Baillot, V. Danos, and T. E. andL. Regnier. Believe it or not, AJM's games model is a model of classical linear logic. In *Proc.* 12th *LICS*, pages 68–75. IEEE, 1997.

[BDER97b] P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Timeless games. In *Proc. CSL '97*, volume to appear. Springer Verlag, 1997.

[CuBoeh] P.-L. Curien. Abstract böhm trees. *Mathematical Structures in Computer Science*, 1998. to appear.

[CuHer96] P.-L. Curien and H. Herbelin. Computing with abstract böhm trees. In *Proc. of Third Fuji International Symposium on Functional and Logic Programming, World Scientific (Singapour)*, 1998.

[DHR96] V. Danos, H. Herbelin, and L. Regnier. Game semantics and abstract machines. In *Proc.* 11th *LICS*, pages 394–405. IEEE, 1996.

3.7 France Telecom – CNET

CNET is involved in the design of many algorithms to be used in a distributed environment. During 1997, CNET participated to two meetings (Bologna and Amsterdam) and two experiments related to CONFER-2 have been made.

The first one consists of adapting a theorem prover (Coq) for specifying and proving safety and liveness properties of protocols and distributed algorithms in a modular way. This works implements results obtained in 1996 by P. Crégut and B. Heyd on the theory of Unity. In order to apply Coq-Unity to real-world protocol, a translation from SDL is used. This method is currently applied on an ATM protocol.

Our second experiment is with join-calculus language (P. Brisset). The release of an implementation of the *join-calculus* by INRIA in 1997 allowed us to begin evaluating its usefulness for real-world applications.

We took advantage of the features of the *join-calculus* to design a very primitive distributed search engine that indexes text files residing on a network of workstations. Each workstation runs an autonomous agent which first builds an indexed database of the local files, then answers keyword-based queries while continuously updating the database. The only central element is a name-server whose function is mainly to transmit queries and answers between these agents and client processes.

Compared to popular WWW search engines like AltaVista, which fetch files from servers through the Internet and process them in a high-performance, centralized database server, this distributed architecture yields lower network usage (assuming low query rates) and allows the database to be kept consistent with the original files at a much lower cost.

Future work on this application might include:

- designing a generic mechanism that would allow agents to be upgraded without interrupting the system;
- testing a hybrid architecture in which local databases are transmitted incrementally to a central server; this would allow high query rates while still yielding good consistency with low usage of network resources;
- implementation of fault-tolerance, which is expected to be easy with upcoming releases of the *join-calculus*.

3.8 ICL

3.8.1 Research Directions

The role of ICL in CONFER-2 is, through our work on designing programming languages and advanced end-user applications, to be serving as a "laboratory" in which theoretical results are transferred into industrial usage.

Since the start of CONFER-2 a new research programme, called the ICL GTD[1] framework in Agents and Mobility, has been set up. The programme encourages I-CL businesses to take part in the research process from an early stage to ensure they are prepared for knowledge and technology transfer as results emerge. The framework includes setting up a virtual laboratory in collaboration with Fujitsu Laboratories, coordinating sponsored medium/longer term research at universities and external research laboratories (ANSA at APM, Cambridge and IC-Parc at Imperial College), a number of research and development projects with ICL businesses (TeamWare, Financial Services, ProcessWise and DAIS) and coordinating and participating in ICL efforts in defining market and technical strategies, architectural and methodological developments across the company. The participation in CONFER-2 is an integral part of this framework.

Of particular interest to CONFER-2 are activities on understanding the relationship between mobile agents, higher order processes, Milner's action structures, distributed concurrent functional programming languages, such as Facile and Business Process Modeling, in particular the POSD formalism developed by ICL ProcessWise. This is work in progress, but preliminary results may be viewed on the web (http://www.prattens.demon.co.uk/webposd/posdw.htm).

Furthermore, work on understanding the foundation for mobile agent programming and mobile/distributed computing support in general is forming a core of activities in developing ICL's technical strategy in this area.

In addition to the technical work relevant to CONFER-2 Lone Leth Thomsen and Bent Thomsen have assisted Jean-Jacques Levy in the management of the working group.

3.8.2 Exchange of Personnel

Lone Leth Thomsen and Bent Thomsen participated in the 2nd CONFER-2 workshop at CWI 20.10.97-22.10.97.

Lone Leth Thomsen and Bent Thomsen visited the CONFER-2 group at Edinburgh University 8.11.96.

In addition two meetings with the Theory and OPERA groups at the CS Lab in Cambridge and a meeting with the Theory group at Sussex University have taken place in the first year of CONFER-2. Although these meetings are not considered official CONFER-2 meetings issues related to CONFER-2 were discussed.

[1]Group Technical Directorate

3.8.3 Publications

In the first year of CONFER-2 the group at ICL involved in the working group has produced two relevant publications:

- Leth Thomsen, L., and Thomsen, B.: "Mobile Agents – THE new paradigm in computing", ICL Systems Journal, Volume 12, Issue 1, pp. 14–40, May 1997.
- Thomsen, B.: "Programming Languages, Analysis Tools and Concurrency Theory", ACM Computing Surveys *28A*(4), December 1996. (http://www.acm.org/-pubs/articles/journals/surveys/1996-28-4es/a57-thomsen/a57-thomsen.html)

3.9 INRIA-Rocquencourt

3.9.1 Research directions

INRIA Rocquencourt has worked in the following directions.

- Distributed Join-Calculus.

  Version 1.04 has been released in June by Fournet and Maranget. This prototype includes a distributed implementation of the join-calculus, which is about 20000 lines of Caml-light code. The system comprises a byte-code compiler, a run-time interpreter, an on-line documentation, a tutorial and many examples. As demo programs, we have a distributed calculation of the Mandelbrot set and several games. An article on the type-inference algorithm and a demo were presented at CONCUR'97 [FLMR97] and at TACS'97 [Le97]. The system is available at http://www.pauillac.inria/join/.

- Equivalences in the Join-Calculus

  A new theory of bisimulation has been developed by Boreale (Roma), Fournet (INRIA) and Laneve (Bologna). They consider open terms of the join calculus, and define two bisimulation, weak and asynchronous, which are both refinments of the observational equivalence. This gives a compositional sense to the interaction of the terms of the join-calculus with contexts, and is a start point to logical systems for reasoning on join-terms. An article has been submitted to a conference, the work has been presented at the Confer Workshop in Amsterdam [BFL98].

  More generally, many equivalences have been proposed for process algebra for the study of convergence, security or fairness. Gonthier started a comparison of equivalences used both in the pi and join calculus, mostly based on reduction semantics, observation of barbs and preservation by contexts. Many variants appear according to the way these elementary constructs are combined. But fortunately many lead to a same result. One shows that fair-testing is connected to coupled bisimulation, or that omega-limit bisimulation is also related to single-barb gfp bisimulation. This provides new proof techniques and a better understanding of process calculi. An article is in preparation, the work has been presented at the Confer Workshop in Amsterdam [Gon98].

- Concurrency and Security

  Secure implementation of communication in the join calculus has been developed by Abadi (DEC/SRC), Fournet and Gonthier. It is easy to write secure protocols when lexical scope is guaranteed, but one needs cryptographic means to enforce a secure lexical scope discipline. By compositionality, two protocols can be considered: one based on the redefinition of secure emission/reception primitives, another one based on firewalls for some locations. The correctness proofs are based on bisimulation techniques.

A presentation by Fournet has been made at the DARPA workshop on *Foundations for Secure Mobile Code*, March 26-28 mars in Monterey [Fou97], and at the Confer meeting in Amsterdam. An article is in preparation [AFG98].

- Distributed Garbage Collection

  F. le Fessant designed and implemented a scalable distributed GC algorithm which can collect cycles. It is a feasible implementation of an old idea by Hughes. An article has been sent to a conference [leF97].

- Expressivity and types

  The π-calculus with synchronous output and mixed-guarded choices is strictly more expressive than the π-calculus with asynchronous output and no choice. Palamidessi recently proved that there is no fully compositional encoding from the former into the latter that preserves divergence-freedom and symmetries. However there are 'good' encodings for languages with (1) input-guarded choice, (2) both input- and output-guarded choice, and (3) mixed-guarded choice. One can investigate them with respect to compositionality and divergence-freedom. The first and second encoding satisfy all criteria, but various 'good' candidates for the third encoding — inspired by an existing distributed implementation — invalidate one or the other criterion. This study suggests that the combination of strong compositionality and divergence-freedom is too strong for more practical purposes [Ne97a].

  We investigated confluence properties for concurrent systems of message-passing processes, because such properties have proved to be useful for a variety of applications, ranging from reasoning about concurrent objects to mobile and high-speed telecommunication protocols. Confluence means that for every two computations starting from a common system state, it is possible to continue the computations, so to reach a common state again. In order to prove confluence for a given system, we are required to demonstrate that for all states reachable by computation from the starting state, the 'flowing together' of possible computations is possible. However, it is possible to prove confluence properties for concurrent systems without having to generate all reachable states by use of a type system that supports a static analysis of possible sources of non-confluence. In message-passing systems, confluence is invalidated whenever two processes compete for communication with another process. We may statically check the occurrence of such situations by reducing them to the concurrent access on a shared communication port. For the technical development, we focus on the setting of a polarized pi-calculus, where we formalize the notion of port-uniqueness by means of overlapping-free context-redex decompositions. We then present a type system for port-uniqueness that, taking advantage of a subject reduction property, yields a sufficient criterion for guaranteeing confluence. This work is published in the Proceedings of FMICS'97 [NS97].

3.9.2 Persons and exchanges

INRIA Rocquencourt participants are Jean-Jacques Lévy, Fabrice le Fessant, Cédric Fournet, Georges Gonthier, Luc Maranget, Uwe Nestmann and Didier Rémy.

Uwe Nestmann moved to Aalborg in November, Carolina Lavatelli moved from ENS to Rocquencourt in December. Cédric Fournet (PhD student) visited Bologna to acheive a common paper with Michele Boreale and Cosimo Laneve.

From Cambridge, we had visits 2 one-week visits with Andrew Gordon, May 26-29, and Peter Sewell, November 24-28. Also Cosimo Laneve (Bologna), Julian Rathke (Sussex), and Nobuko Yoshida (Edinburgh) visited our site.

3.9.3 Publications

[AFG98] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. To be submitted for a conference, November 1997 .

[BFL98] M. Boreale, C. Fournet, and C. Laneve. Bisimulations in the join-calculus. Submitted for a conference, September 1997.

[leF97] F. le Fessant, I. Piumarta, and M. Shapiro, An implementation for complete asynchronous distributed garbage collection. Submitted for publication, November 1997, available electronically (`http://www-sor.inria.fr/docs/`).

[Fou97] C. Fournet. Security within a calculus of mobile agents? April 1997, http://www.cs.nps.navy.mil/research/languages/statements/fournet.ps.

[FLMR97] C. Fournet, C. Laneve, L. Maranget, and D. Rémy. Implicit typing à la ML for the join-calculus. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR '97: Concurrency Theory (8th International Conference, Warsaw, Poland, July 1997)*, volume 1243 of *LNCS*, pages 196–212. Springer, 1997.

[FouMa97] C. Fournet and L. Maranget. The join-calculus language. Available electronically (`http://pauillac.inria.fr/join`), June 1997.

[Gon98] G. Gonthier. On asynchronous barbs, simulations, and fair testing. Submitted to a conference, Octobre 1997.

[Le97] J.-J. Lévy. Some results in the join-calculus. In M. Abadi and T. Ito, editors, *TACS '97: Theoretical Aspects of Computer Software*, volume 1281 of *LNCS*, pages 233–249. Springer, 1997.

[Ne97a] U. Nestmann. What is a 'good' encoding of guarded choice? In C. Palamidessi and J. Parrow, editors, *EXPRESS '97: Expressiveness in Concurrency (Santa Margherita Ligure, Italy, September 8–12, 1997)*, volume 7 of *ENTCS*. Elesevier Science Publishers, 1997.

[NePie97] U. Nestmann and B. C. Pierce. Decoding choice encodings. In U. Montanari and V. Sassone, editors, *CONCUR '96: Concurrency Theory (7th International Conference, Pisa, Italy, August 1996)*, volume 1119 of *LNCS*, pages 179–194. Springer, 1996. Revised full version as report ERCIM-10/97-R051, European Research Consortium for Informatics and Mathematics, 1997.

[NS97] U. Nestmann and M. Steffen. Typing confluence. In S. Gnesi and D. Latella, editors, *Second International ERCIM Workshop on Formal Methods in Industrial Critical Systems (Cesena, Italy, July 4–5, 1997)*, pages 77–101. Consiglio Nazionale Ricerche di Pisa, 1997. Also available as report ERCIM-10/97-R052, European Research Consortium for Informatics and Mathematics, 1997.

3.10 INRIA-Sophia

3.10.1 Research directions

The notion of *type*, intended in a very general sense as a “partial specification of the way system components should be used”, is present, at various degrees, in most of our works this year.

In [Am97] R. Amadio presents a model of mobility based on the pi-calculus. He relies on an elementary typing, which ensures that in a system, at every moment, to a given name there corresponds a unique resource (whose state can change from one call to another, however). In this way, one can consider a site – or a locality – as a name associated to a resource whose state is either ‘alive’ or ‘dead’. R. Amadio develops on this basis a formalism to speak about migration and failures, and he provides reasoning methods which use a notion of bisimulation.

In [San97a], D. Sangiorgi formulates a type system for mobile processes that guarantees, in a distributed process, that at any moment a given name gives the access to a unique stateless resource. This discipline, called uniform receptiveness, is employed, for instance, when modeling functions, objects, higher-order communications, remote-procedure calls. The impact of the discipline on behavioural equivalences and process reasoning is studied. Some theory and proof techniques for uniform receptiveness is developed, and their usefulness on some non-trivial examples is illustrated.

The work [San97J] is an application of the proof techniques developed in [San97a]. Cliff Jones has proposed transformations between concrete programs and general transformation rules that increase concurrency in a system of objects, and has raised the challenge of how to prove their validity. A proof of correctness of the hardest of Jones’s concrete transformations is presented.

The *blue calculus*, introduced by G. Boudol in a POPL’97 paper [Bou97a], proposes a synthesis of the lambda-calculus and the pi-calculus. This calculus contains a new type system which incorporates both the notion of functional type and that of sort for communication channels. This type system has been further studied by S. Dal-Zilio in [Dal97], where it is shown to extend naturally to an implicit polymorphic type system à la ML.

However, these types only express very weak properties of the behaviour of processes. For instance, one cannot guarantee the absence of *deadlock*; here we mean by deadlock that a process calls a resource that will never become available. To overcome this lack of expressiveness, G. Boudol has proposed in [Bou97b] a more refined notion of type, which incorporates a part of Hennessy-Milner logic: a resource declaration is given a modal type indicating that a value of a certain type is available on a certain name. This new type system is shown to enjoy the subject reduction property. This research will be hopefully extended with a proof that typability implies the absence of deadlock.

In [PiSa97], parametric polymorphism in message-based concurrent programs is investigated, focusing on behavioral equivalences in a typed process calculus analogous to the polymorphic lambda-calculus of Girard and Reynolds. Polymorphism constrains the power of observers by preventing them from directly manipulating data values whose types are abstract, leading to notions of equivalence much coarser than the standard untyped ones. The nature of these constraints is studied through simple examples of concurrent abstract data types and develop basic theoretical machinery for establishing bisimilarity of polymorphic processes. Some surprising interactions between polymorphism and aliasing are also observe, drawing examples from both the polymorphic pi-calculus and ML.

Another work on mobility [AmPra97] concerns the modelling of *mobile hosts* on a network in a simple name-passing process calculus. The protocol considered is a highly simplified version of proposals for *mobility support* in the version 6 of the Internet Protocols (IP). We concentrate on the issue of ensuring that messages to and from mobile agents are delivered without loss of connectivity. We provide three models of increasingly complex nature of a network of routers and computing agents that are interconnected via the routers: the first is without mobile agents and is treated as a specification for the next two; the second supports mobile agents, and the third additionally allows correspondent agents to *cache* the current location of a mobile agent. Following a detailed analysis of the three models to extract invariant properties, we show that the three models are related by a suitable notion of equivalence based on *bisimulation*.

In [AmCou97] we consider the issue of representing “infinite” and “total” objects like infinite streams or non-terminating processes in type theory. We present a realizability interpretation of co-inductive types based on *partial equivalence relations* (per’s). We extract from the per’s interpretation sound rules to type recursive definitions. These recursive definitions are needed to introduce ‘infinite’ objects of co-inductive type. We show that the proposed type system enjoys the basic syntactic properties of subject reduction and strong normalization with respect to a confluent rewriting system first studied by Gimenez. We also compare the proposed type system with those studied by Coquand and Gimenez. In particular, we provide a semantic reconstruction of Gimenez’s system which suggests a rule to type nested recursive definitions.

The final version of the book [AmCu97] has been delivered. The book addresses the mathematical aspects of the semantics of programming languages. The mathematical objects manipulated are order-theoretic structures, and the languages we consider are typed lambda-calculi. The book considers various extensions of typed lambda-calculi to include non-determinism, communication, and concurrency, and it summarizes some of the work carried on in the Confer project.

3.10.2 Participation to CONFER Workshops

R. Amadio, G. Boudol, I. Castellani, D. Sangiorgi and S. Dal-Zilio attended the first CONFER II workshop in Bologna.

G. Boudol, I. Castellani and M. Merro attended the second CONFER II workshop in Amsterdam.

(We have not used CONFER money for other visits or activities.)

3.10.3 Publications

[Am97] R. Amadio. An asynchronous model of locality, failure, and process mobility. In *Proc. Coordination 97, Springer Lect. Notes in Comp. Sci. 1282*, 1997.

[AmCou97] R. Amadio and S. Coupet-Grimal. Analysis of a guard condition in type theory (preliminary report). Technical Report TR 1997.245. Also RR-3300 INRIA, Université de Provence (LIM), 1997.

[AmCu97] R. Amadio and P.-L. Curien. *Domains and Lambda-Calculi.* Cambridge University Press. To appear.

[AmPra97] R. Amadio and S. Prasad. Modelling IP mobility. Technical Report TR 1997.244. Also RR-3301 INRIA, Université de Provence (LIM), 1997.

[Bou97a] G. Boudol. The pi-calculus in direct style. In *Proceedings POPL'97*, 1997.

[Bou97b] G. Boudol. Typing the use of resources in a concurrent calculus. In *Proceedings ASIAN'97, LNCS 1345*, 1997.

[Dal97] S. Dal-Zilio. Implicit polymorphic type system for the blue calculus. Technical Report RR-3244, INRIA, 1997.

[PiSa97] B. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. In *24th POPL.* ACM Press, 1997.

[San97a] D. Sangiorgi. The name discipline of receptiveness. volume 1256 of *Lecture Notes in Computer Science.* Springer Verlag, 1997. Full paper available electronically as `ftp://zenon.inria.fr/meije/theorie-par/-davides/Receptiveness.ps.Z`.

[San97J] D. Sangiorgi. Typed π-calculus at work: a proof of Jones's parallelisation transformation on concurrent objects. Presented at the Fourth Workshop on Foundations of Object-Oriented Languages (FOOL 4). To appear in *Theory and Practice of Object-oriented Systems*, 1997.

3.11 KTH

3.11.1 Research directions

We have developed the update-calculus where processes can perform *update* actions with side effects, and a scoping operator controls the extent of the effects [PV97]. In this way it incorporates fundamental concepts from imperative languages, concurrent constraints and from functional formalisms. Structurally it is similar to but simpler than the pi-calculus; it has only one binding operator and a symmetry between inpout and output.

Furthermore, we have completed earlier work within Confer-1 regarding unfold/fold program transformations in languages with nondeterminism [Lisp98]. These results hold under the classical condition of "restricted folding-unfolding" [Cou90], which is quite restrictive with respect to the formation of new recursive definitions. We have made some initial investigations towards extending the results to the particular formation of new recursive definitions that typically occur during program specialization.

3.11.2 Persons and exchanges

During 1997, the following persons have been active within Confer-2: Björn Lisper, Joachim Parrow, Björn Victor (Uppsala University), and José Luis Vivas. Lisper, Parrow and Vivas attended the Confer workshop in Bologna. Victor attended the workshop in Amsterdam.

3.11.3 Publications

[Cou90] B. Courcelle. Recursive applicative program schemes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 9, pages 459–492. Elsevier Science Publishers B. V., 1990.

[Lisp98] B. Lisper. Computing in unpredictable environments: Semantics, reduction strategies, and program transformations. *Theoretical Comput. Sci.*, 190(1):61–85, Jan. 1998.

[PV97] J. Parrow and B. Victor. The update calculus (full version). Technical Report DoCS 97/93, Department of Computer Systems, Uppsala University, Sweden, Sept. 1997. Extended abstract to appear in the proceedings of AMAST'97.

3.12 Università di Pisa

3.12.1 Outline

The major line of research of the group at the Dipartimento di Informatica, Università di Pisa, has been in the areas of *Calculi*, *Foundational Models and Abstract Machines* and *Programming Languages.*

- *Calculi*

  Research has continued on developing efficient finite-state techniques for verifying properties of history dependent process calculi (e.g. calculi for mobile processes). A prototype version of a semantic-based verification environment for the π-calculus, based on Montanari and Pistore HD-Automata has been developed. The environment supports verification of both behavioral properties (bisimulation checkers) and logical formulae (model checkers).

- *Foundational Models and Abstract Machines*

  The tile model relies on certain rewrite rules with side effects, called *tiles*, reminiscent of both SOS rules and rewriting logic rules. The tile model provides a logic of concurrent systems with conditional state changes and synchronization. The tile model has been shown to be especially useful for defining compositional models of mobile processes and causal and located concurrent systems. The tile model has been used for modelling coordination formalisms equipped with flexible synchronization primitives. The main advantage of the tile approach is the full compositionality of the underlying logic, which is able to handle computations of (nonclosed) system components as they were new rewrite rules specifying complex coordinators.

- *Programming Languages*

  We introduces a kernel programming language, called `LLinda`, which supports a programming paradigm where processes as well as data can migrate from one computing environment to another. The language is equipped with a sophisticated type system which statically checks access rights violations of mobile agents.

3.12.2 Perspective

We are interested to further develop theoretical models for mobile process calculi and the verification techniques based on these models. Bisimulations for the π-calculus are instances of a larger class of bisimulation semantics for processes calculi, called *history dependent.* This concept can be found in various semantics for process calculi, e.g. locality or causality. One aspect that we intend to study is the development of verification methods for checking history dependent bisimulation semantics.

We intend to exploit the capabilities of the tile model to formally describe arbitrary coordination patterns (e.g. workflows) between tasks to be executed by sequential processes or by other coordinators. One aspect that we intend to study is the development of a tile-based model for the specification of Software Architectures.

We intend to further investigate notions of types which ensure only authorized accesses to information. We aim at demonstrating that typing information can be systematically used to guarantee that well–typed processes always enjoy classes of security properties. One aspect that we intend to study is the modelling of dynamic transmission of access rights.

3.12.3 Persons

The group at Pisa involved in the project consists of Ugo Montanari, Gianluigi Ferrari, Marco Pistore, Roberto Bruni and Gioia Ristori.

Two PhD-students, Marco Pistore and Roberto Bruni, both under the supervision of Ugo Montanari, are working on topics which are closely related to the aims of the CONFER 2 Working Group.

3.12.4 Publications

1. De Nicola, R., Ferrari, G., Pugliese, R., Locality Based Linda: Programming with Explicit Localities, In Proc. TAPSOFT'97, LNCS 1214, 1997.

2. De Nicola, R., Ferrari, G., Pugliese, R., Coordinating Mobile Agents via Blackboards and Access Rights, In Proc. COORDINATION'97, LNCS 1282, 1997.

3. Ferrari, G., Ferro, G., Gnesi, S., Montanari, U., Pistore, M., Ristori, G., An automata based verification environment for mobile processes, In Proc. TACAS'97, LNCS 1217, 1997.

4. Ferrari, G., Montanari, U. A Tile-Based Coordination View of Asynchronous π-calculus, In Proc. MFCS'97, LNCS 1295, 1997.

5. Ferrari, G., Montanari, U. Tiles for Concurrent and Located Calculi. In Proc. EXPRESS'97, ENTCS 7, 1997.

6. Gadducci, F. and Montanari, U., The Tile Model. In: Gordon Plotkin, Colin Stirling, and Mads Tofte, Eds., Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, to appear.

7. Montanari, U. and Pistore, M., Minimal Transition Systems for History-Preserving Bisimulation. In Proc. STACS'97, LNCS 1200, 1997.

8. Bruni, R., Montanari, U., Zero-Safe Nets, or Transition Synchronization Made Simple In Proc. EXPRESS'97, ENTCS 7, 1997.

3.12.5 Description of Technical Contributions

1. The issue of defining a process calculus which supports programming with explicit localities is investigated. We introduce a language which embeds the asynchronous Linda communication paradigm extended with explicit localities in a process calculus. We consider multiple tuple spaces that are distributed over a collections of sites and use localities to distribute/retrieve tuples and processes over/from these sites. The operational semantics of the language turns out to be useful for discussing the language design, e.g. the effects of scoping disciplines over mobile agents which maintain their connections to the located tuple spaces while moving along sites. The flexibility of the language is illustrated by a few examples.

2. LLinda (Locality based Linda) is a variant of Linda which supports a programming paradigm where agents can migrate from one computing environment to another. In this paper, we define a type system for LLinda that permits statically checking access rights violations of mobile agents. Types are used to describe processes intentions (read, write, execute, ...) relatively to the different localities they are willing to interact with or they want to migrate to. The type system is used to determine the operations that processes want to perform at each locality, to check whether they comply with the declared intentions and whether they have the necessary rights to perform the intended operations at the specific localities.

3. An automata-based verification environment for the π-calculus is developed. The environment takes a direct advantage of a general theory which allows to associate ordinary finite state automata to a wide class of π-calculus agents, so that equivalent automata are associated to equivalent agents. A key feature of the approach is the re-use of efficient algorithms and verification techniques (equivalence checkers and model checkers) which have been developed and implemented for ordinary automata. The verification environment has been implemented on top of an existing verification environment for process calculi: the JACK environment.

4. Tiles are rewrite rules with side effects, reminiscent of both Plotkin SOS and Meseguer rewriting logic rules. They are well suited for modeling coordination languages, since they can be composed both statically and dynamically via possibly complex synchronization and workflow mechanisms. In this paper, we give a tile-based bisimilarity semantics for the asynchronous π-calculus of Honda and Tokoro and prove it equivalent to the ordinary semantics. Two kinds of tiles are provided: *activity tiles* and *coordination tiles*. Activity tiles specify the basic interactions sequential processes are able to perform, without considering the operational environment where they live. Instead, coordination tiles control the global evolution of programs.

5. When concurrency is a primitive notion, models of process calculi usually include commuting diamonds and observations of causality links or of abstract locations. However it is still debatable if the existing approaches are natural, or rather if

they are an *ad hoc* addition to the more basic interleaving semantics. In the paper a treatment of concurrent process calculi is proposed where the same operational and abstract concurrent semantics described in the literature now descend from general, uniform notions. More precisely we introduce a tile-based semantics for located CCS and we show it consistent with the ordinary concurrent (via permutation of transitions) and bisimilarity based location semantics. Tiles are rewrite rules with side effects, reminiscent of both Plotkin SOS and Meseguer rewriting logic rules. We argue that the tile model is particularly well suited for defining directly operational and abstract semantics of concurrent process calculi in a compositional style.

6. In this paper we aim at describing a framework for specifying the behaviour of a wide class of rule-based computational systems. We gathered our inspiration both from the world of term rewriting, in particular from the rewriting logic approach, and of concurrency theory, the structural operational semantics, the context systems and the structured transition systems approaches among others. Our framework recollects many properties from these sources: First, it provides a compositional way to describe both the states and the sequences of transitions performed by a given system, taking into account their distributed nature. Second, a suitable notion of typed proof allows us to recast also those formalisms relying on the notions of synchronization and side-effects to determine the actual behaviour of a system. Moreover, both an operational and an observational semantics can be introduced: Operationally, those sequences of transitions are identified, that denote a family of "computationally equivalent" behaviours. Observationally, those states are identified, that denote a family of "bisimilar" configurations. Finally, and as a further abstraction step, our framework can be conveniently recasted using double-categories, recovering also its operational semantics.

7. In this paper we propose a new approach to check history-preserving equivalence for Petri nets. Exploiting this approach, history-preserving bisimulation is proved decidable for the class of finite nets which are n-safe for some n. Moreover, since we map nets on ordinary transition systems, standard results and algorithms can be re-used, yielding for instance the possibility of deriving minimal realizations. The proposed approach can be applied also to other concurrent formalisms based on partial order semantics, like CCS with causality.

8. In addition to ordinary places, called stable, zero-safe nets are equipped with zero places, which in a stable marking cannot contain any token. An evolution between two stable markings, instead, can be a complex computation called stable transaction, which may use zero places, but which is atomic when seen from stable places: no stable token generated in a transaction can be reused in the same transaction. Every zero-safe net has an ordinary Place-Transition net as its abstract counterpart, where only stable places are maintained, and where every transaction becomes a transition. The two nets allow us to look at the same system from both an abstract and a refined viewpoint. To achieve this result no

new interaction mechanism is used, besides the ordinary token-pushing rules of nets. The refined zero-safe nets can be much smaller than their corresponding abstract P/T nets, since they take advantage of a transition synchronization mechanism. For instance, when transactions of unlimited length are possible in a zero safe net, the abstract net becomes infinite, even if the refined net is finite. In the second part of the paper two universal constructions - both following the Petri nets are monoids approach and the collective token philosophy - are used to give evidence of the naturality of our definitions. More precisely, the operational semantics of zero-safe nets is characterized as an adjunction, and the derivation of abstract P/T nets as a coreflection.

3.13 University of Sussex

3.13.1 Research directions

Value-passing processes

There were two publications in this area. In [Ra97a] an axiomatic characterisation of both late and early observational congruence for a class of fully parameterised processes is given. This result uses a new derivation rule which is an intuitive generalisation of unique fixpoint induction based on loop invariants for symbolic graphs. In the same paper it is shown that the new rule can actually be derived from existing formulations of unique fixpoint induction for value-passing calculi.

In [Ra97b] a first order modal mu-calculus is described which uses parameterised maximal fixpoints to describe safety and liveness properties of value-passing processes. A proof system for deciding if a process satisfies a formula is presented. Certain deduction rules of the system carry side conditions which leave auxiliary proof obligations of checking properties of the value language. Although the proof system is in general incomplete two completeness results are given for significant sub-logics.

Mathematical Models for concurrent languages

There are two streams of work here. The first, summarised in [Je97], gives a precise relationship between

- *mixed data and control flow graphs*, which are commonly used in software engineering as a semi-formal notation for describing and analysing algorithms
- and an abstract representation of formal semantics which uses partially traced Cartesian categories.

This document also contains some more speculative work on how closed structure (to represent higher-order functions) or two-category structure (to represent operational semantics) might be included in this graphical framework.

In [HaHen97e] traditional denotational techniques are used to give a denotational model of a subset of *core Facile.* Specifically this is applied typed call-by-value language which in addition to the usual types has an extra type for processes, which can exchange values along communication channels, a la *CCS.* The model is shown to be fully-abstract with respect to a version of *may testing.* It is notable that in order to achieve full-abstraction it is necessary to add to the language an operator which allows processes report the result of a computation.

Semantic theories for distributed processes

In [RieHen97a] location failure for distributed processes is studied in detail. A language is defined in which threads are run at distributed locations, and from this a semantic theory is developed using the basic notions of *observer* and *experiment.* Roughly, an observer interacts with a distributed process either by synchronizing with one of its threads or by halting one of its locations; two programs are equivalent if they present the same structure of observations, evolving over time. The main result of this work is to provide an alternative characterization of this *observational equivalence* which is suitable to automatic verification. This is done using a temporal logic to characterize the interdependence between synchronization events and failures.

In [RieHen97b] a more expressive language is presented, which includes agent migration, location failure and a rich system of *capabilities* used to tag communicated data. Intuitively, when the name, say, of a location is communicated, a set of capabilities for that location is communicated at the same time; these may include the capability to read data, to write data, to halt the location, to migrate to it, or to allocate fresh resources at it. We present a series of type systems for this language and show, through examples, that very expressive type systems are necessary in order to be able to type many important practical programs.

3.13.2 Personnel and Exchanges

The personnel at the University of Sussex who were actively involved in Confer-2 related research during the period covered by the progress report are currently Chrysafis Hartonas, Matthew Hennessy, Alan Jeffrey, Dusko Pavlovic, Julian Rathke, James Riely.

Matthew Hennessy and James Riely attended the first CONFER II workshop in Bologna.

Matthew Hennessy and Alan Jeffrey attended the second CONFER II workshop in Amsterdam.

3.13.3 Publications

[HaHen97e] C. Hartonas and M. Hennessy. Full abstractness for a functional/concurrent language with higher-order value-passing. Full version available as Computer Science Technical Report 1/97, University of Sussex, 1997. Available from http://www.cogs.susx.ac.uk/.

[Je97] A. Jeffrey. Flow graphs and semantics of programs. available from http://www.cogs.susx.ac.uk/users/alanje/premon/, 1997.

[Ra97a] J. Rathke. Unique fixpoint induction for value-passing processes. In *Proc. LICS'97, 12th Annual Symposium on Logic in Computer Science, Warsaw.* IEEE Computer Society Press, 1997. To appear.

[Ra97b] J. Rathke and M. Hennessy. Local model checking for value-passing processes. In *Proc. TACS'97, International Symposium on Theoretical Aspects of Computer Software, Sendai.* Springer-Verlag, 1997.

[RieHen97a] J. Riely and M. Hennessy. Distributed processes and location failures. Technical Report 2/97, 1997. Extended abstract presented at ICALP97.

[RieHen97b] J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *Proc. POPL'98, ACM Conference on Principle of Programming Languages, San Diego.*

3.14 University of Warwick

3.14.1 Research activity

Work has been done on theory of π-calculus, semantic definition of concurrent object-oriented programming languages, and correctness of transformation rules for programs and of concurrent algorithms on data structures.

In a process setting, the essence of confluence is that the occurrence of one action may not preclude others. In the case of π-calculus, where names may be extruded from their scopes, care is needed to find the right definition. Basic results on confluence in π-calculus are presented in [PW97]. In particular, conditions under which combinations of confluent, or almost-confluent, processes yield confluent systems are presented.

The view that general models of mobile processes, such as π-calculus and Higher-Order π-calculus, provide a good framework for the definition of a variety of concurrent object-oriented languages is illustrated in [LW97a]. A technique involving process continuations is used to give a natural and direct semantic definition for a language containing constructs for several kinds of inter-object communication. In the unpublished MSc dissertation [S97], an experimental extension of Pict with actor primitives is given semantic definition and implemented. Its author is now studying for a PhD at Cambridge.

In [LW97b], which was revised for publication this year, a theory of partial-confluence of processes is developed, in CCS and in π-calculus. The main theorem concerns systems which can be thought of as composed from two components: a questioner, and an answerer which may process questions concurrently. It gives conditions under which a system of this kind is behaviourally indistinguishable from the corresponding system in which the answerer may handle only one question at a time. This result, together with a semantic definition of the kind described above, is used to show the interchangeability in an arbitrary program context of two class definitions which prescribe binary-tree symbol-table data structures, one supporting concurrent operations, the other sequential.

In [PW97b], this work is extended to show the soundness of transformation rules for concurrent object-oriented programs which increase the scope for concurrent activity within the system prescribed by a program, without altering its observable behaviour. The rules allow one object to release another from a rendezvous before it has finished all the associated activity, and allow one object to delegate to a second object the responsibility for returning a result to a third object.

In [PW97c], the theory of [PW97b]is extended to encompass a larger class of answerers. Analogous behavioural indistinguishability results are obtained, for an appropriate class of questioners. The main advance is that whereas in the case of the symbol-table data structures, the answer to a question is determined at the moment the question is submitted, in [6] up to one state-changing internal action may occur in determining the answer.

This result is used in [PW97d] where algorithms for operating concurrently on a variant of the B-tree data structure are modelled and analysed using π-calculus. In this case the state-changing internal action is the act of locking the disk page in which the operation in question will take effect. Deficiences in and improvements to published algorithms were discovered.

The PhD thesis [P97phd] contains also other work on confluence of processes. The MSc dissertation [I97] begins to explore a new approach to modelling asynchronous communication and fault-tolerance in a process-calculus setting. Its author is now studying for a PhD at Edinburgh.

3.14.2 Workshops and site visits

Working-Group funds were used to support David Walker's participation in the Bologna workshop in March 1997, a visit to the Bologna site in July 1997 where the main focus of interaction was joint work on π-calculus with Davide Sangiorgi at INRIA Sophia-Antipolis, and a visit to the Edinburgh site in November 1996.

3.14.3 Publications

[PW97] A. Philippou and D. Walker, *On confluence in the π-calculus*, in Proceedings of International Colloquium on Automata, Languages and Programming, Bologna, July 1997, Springer-Verlag Lecture Notes in Computer Science, vol. 1256, 314–324.

[LW97a] X. Liu and D. Walker, *Concurrent objects as mobile processes*, in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, G. Plotkin, C. Stirling, and M. Tofte (eds.), MIT Press, to appear.

[S97] M. Sawle, *A π-calculus semantics for actor systems*, MSc dissertation, University of Warwick, September 1997.

[LW97b] X. Liu and D. Walker, *Partial confluence of processes and systems of objects*, Theoretical Computer Science B, to appear.

[PW97b] A. Philippou and D. Walker, *On transformations of concurrent object programs*, Theoretical Computer Science B, to appear.

[PW97c] A. Philippou and D. Walker, *Social confluence in client-server systems*, in Proceedings of Computer Science Logic, Utrecht, September 1996, Springer-Verlag Lecture Notes in Computer Science, vol. 1258, 385–398.

[PW97d] A. Philippou and D. Walker, *A rigorous analysis of concurrent operations on B-trees*, in Proceedings of Concurrency Theory, Warsaw, July 1997, Springer-Verlag Lecture Notes in Computer Science, vol. 1243, 361–375.

[P97phd] A. Philippou, *Reasoning about systems with evolving structure*, PhD thesis, University of Warwick, December 1996.

[I97] C. Ionitoiu, *An operational semantics for loosely coupled systems*, MSc dissertation, University of Warwick, September 1997.

Chapter 4

Deliverables

4.1 Workshop 1

At University of Bologna, with more than 30 participants.

Monday, March 24

9.00	Welcome
9.10 - 9.50	The pi-calculus in direct style G. Boudol
9.50 - 10.30	Pieces of pi: substitution and scoping J. Parrow
11.10 - 11.50	A denotational Model for a subset of Core Facile M. Hennessy
11.50 - 12.30	Encoding Mixed-guarded Choice U. Nestmann
12.45-14.15	Lunch
14.30 - 15.10	An asynchronous model of locality, failure, and process mobility R. Amadio
15.10 - 15.50	Distributed Processes and Location failures J. Riely
16.30 - 17.10	Locality based Linda: programming with explicit localities G. Ferrari
17.10 - 17.50	From Action Calculi to Linear Logic P. Gardner

Tuesday, March 25

9.10 - 9.50	Sequentiality in lambda calculus and PCF via origin tracking J.W. Klop
9.50 - 10.30	A Factorisation Theorem for external derivation P. Mellies
11.10 - 11.50	Zero Safe Nets R. Bruni
11.50 - 12.30	Graph rewriting and labelled transition semantics O. H.Jensen
12.45-14.15	Lunch
14.30 - 15.10	Local Channel Typing for a distributed pi-calculus P. Sewell
15.10 - 15.50	Interpreting Typed Object calculi into typed pi-calculus D. Sangiorgi
16.30 - 17.10	Implicit typing a la ML for the join calculus D. Remy
17.10 - 17.50	The implementation fo the join calculus L. Maranget

Wednesday, March 26

9.10 - 9.50	Distributed functional evaluation V. Danos
9.50 - 10.30	GOI and coherent semantic L. Regnier
11.10 - 11.50	AJM game semantics for classical LL P. Baillot
11.50 - 12.30	P = NP, up to sharing A. Asperti
12.45-14.15	Lunch

List of Participants: Roberto Amadio, Andrea Asperti, Patrick Baillot, Inge Bethke, Gerard Boudol, Roberto Bruni, Ilaria Castellani, Silvano Dal-Zilio, Vincent Danos, Gianluigi Ferrari, Philippa Gardner, Matthew Hennessy, Ole Hoegh Jensen, Jan Willem Klop, Cosimo Laneve, Carolina Lavatelli, James Leifer, Jean-Jacques Levy, Bjorn Lisper, Luc Maranget, Paul-Andre Mellies, Jean-Francois Monin, Uwe Nestmann, Joachim Parrow, Paola Quaglia, Laurent Regnier, Didier Remy, James Riely, Davide Sangiorgi, Peter Sewell, Jose-Luis Vivas.

4.2 Workshop 2

At CWI in Amsterdam, with 30 participants.

Monday, October 20

9.15	Welcome
9.30 - 10.10	Typing the use of resources in a concurrent calculus G. Boudol
10.10 - 10.50	The asynchronous pi-calculus, revisited M. Merro & D. Sangiorgi
11.30 - 12.10	On implementations and semantics of a concurrent P. Sewell
12.10 - 12.50	Functions in fusion: A symmetric calculus of mobile processes B. Victor
13.00-14.15	Lunch
14.30 - 15.10	Abstract Bohm trees, or games as abstract machines P.-L. Curien
15.40 - 16.20	Sequentiality in lambda calculus. I. Bethke & J.W. Klop

Tuesday, October 21

9.30 - 10.10	Bisimulations in the Join Calculus C. Laneve
10.10 - 10.50	A graphical presentation of categorical semantics A. Jeffrey
11.30 - 12.30	On asynchronous barbs, simulations, and fair testing G. Gonthier
12.45-13.45	Lunch
14.00 - 15.00	Games semantics and traditional denotational semantics V. Danos
15.40 - 16.20	Bisimulations and other games J. Loddo & S. Nicolet
19.00	Dinner

Wednesday, October 22

9.30 - 10.30	Secure implementation of channel abstractions C. Fournet
11.10 - 11.50	An implementation of distributed garbage collection F. le Fessant
11.50 - 12.30	On the distributed implementation of languages based on the chemical paradigm, C. Palamidessi
12.45-13.45	Lunch

List of Participants: Andrea Asperti, Inge Bethke, Gerard Boudol, Ilaria Castellani, Pierre Cregut, Pierre-Louis Curien, Silvano Dal-Zilio, Vincent Danos, Roberto Di Cosmo, Fabrice le Fessant, Cedric Fournet, Georges Gonthier, Matthew Hennessy, Alan Jeffrey, Jan Willem Klop, Cosimo Laneve, James Leifer, Jean-Jacques Levy, Jean Loddo, Paul-Andre Mellies, Massimo Merro, Jean-Francois Monin, Uwe Nestmann, Stephane Nicolet, Vincent van Oostrom, Catuscia Palamidessi, Femke van Raamsdonk, Peter Sewell, Lone Thomson, Bent Thomson, Bjorn Victor

4.3 Software deliverables

Several pieces of software have been constructed during the this year of CONFER-2. Note that most of these softwares were started during Confer-1, and that the software deliverables are only planned for at Milestone 3.

The following is a listing of constructed software during the first year of CONFER-2:

- PICT
  Cambridge — Typed higher-order programming language based on π-calculus
  University of Cambridge and Edinburgh — B. Pierce, D. Turner.
  Also with contributions of P. Sewell, P. Wojciechowski.
  Available at http://www.cs.indiana.edu/hyplan/pierce/ftp/pict
- The Join-Calculus
  INRIA-Rocquencourt — C. Fournet, L. Maranget.
  Also with contributions of G. Gonthier, J.-J. Lévy, D. Rémy. Available at http://pauillac.inria.fr/join/
- BOHM, Prototype compiler for λ-calculus, based on graph reduction
  INRIA — A. Asperti.
- An implementation for complete asynchronous distributed garbage collection.
  INRIA-Rocquencourt — F. le Fessant. Also with contributions of I. Piumarta, and M. Shapiro
- Automata-based verification for the JACK environment.
  Pisa — G. Ferrari, G. Ferro, S. Gnesi, U. Montanari, M. Pistore, G. Ristori.

Chapter 5

Progress

Reports are done along the 4 areas announced in page 4 of the technical annex.

5.1 Foundational models and abstract machines

This area treats of various models for rewrite systems with bound variables. It has actually 3 parts. First, we pursue the work on functional models, i.e. optimal reductions in the lambda calculus, sequentiality and axiomatic rewriting systems. Secondly, we consider new models for concurrency, ie action calculi and the tile model. Finally, we treat of the use of rewrite rules as a model for the optimisation of concurrent programs.

5.1.1 Optimal reductions in the λ-calculus

The theory of optimal sharing in β-reduction had many significant improvements recently (see the results of Asperti, Gonthier, Laneve, and Lévy in BRA Confer-1). In 1977, Lévy designed the notion of so-called family-complete reductions, optimal in the number of β-steps. Each of the elementary steps had to contract a set of redexes, and these steps were unit costs. In 1988, Lamping exhibited a data structure and an algorithm, fitting these costs. He heavily used shared structures to keep a single object to contract in each of the elementary steps of family-complete reductions. However, between each of these steps, several bookkeeping operations were needed to share or unshare objects.

Asperti and Mairson (Brandeis University) showed that the number of these bookeeping steps was very large. In fact their number is not *elementary recursive* in the size of the original term, decorated by the type of all its subterms, whereas the number of β-steps is polynomial. More precisely, it is polynomial in the size of some η-expansion of the initial term. This means that the notion of optimal "parallel" reduction, as formalized by Lèvy, is not *elementary recursive.* It cannot have a simple implementation since, essentially, *all the computational work* can be done via sharing. Actually, the result is a straightforward consequence of the fact that every term of the simply typed lambda calculus can be essentially reduced in a linear number of "parallel" (i.e. sharable) β-reduction. [AM98]

The result should not be understood in a too negative sense. On the road to this result, many interesting and fine properties have been found in the behaviour of redexes families, which can be of use for other areas such as sequentiality or strong normalisation.

[AM98] A. Asperti, H.Mairson *Optimal β-reduction is not elementary recursive.* Proc. of the twenty-fifth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'98).

5.1.2 Axiomatic Rewriting Systems

Melliès pursues his work on abstract rewriting systems which are an axiomatic presentation of rewriting systems with bound variables. This year he obtained a factorisation theorem in the sense of Freyd & Kelly. The theorem means that the efficient part of a computation can always be separated from its useless part modulo redex permutations, and that this separation is categorically functorial. This work was presented at the Confer workshop in Bologna and at the CTCS '97 conference in Santa Margherita.

Subsequently, Melliès has obtained a stability result in the sense of Berry expressing the existence of a cone $(e_i : M \to V_i)_{i \in I}$ of minimal computations from any term M to the set of its values V_i. The result whose proof relies heavily on the factorisation theorem is verified by any axiomatic rewriting system. In the lambda-calculus, it extends the result by Wadsworth that the head-reduction is the *minimal* reduction path from a lambda-term to its head-normal form. In the call-by-value lambda-calculus, the cone $(e_i : M \to V_i)_{i \in I}$ contains at most a singleton which describes the strategy of Landin SECD machine. In the lambda-sigma calculus, the existence of a cone and its finiteness implies the completeness of various weak strategies and the corresponding abstract machines. A paper containing these results has been submitted to LiCS '98.

[M97] P.-A. Melliès, *A factorisation theorem in Rewriting Theory.* Proc. of the Category Theory and Computer Science symposium (CTCS'97).

5.1.3 Sequentiality

Berry's Sequentiality Theorem for lambda calculus was generalised by J.W. Klop and I. Bethke. Using the setting of infinitary lambda calculus where Bohm trees are infinite normal forms, the technique of origin tracking (also used by Bertot at PLDI'91 and Abadi, Lévy and Lampson at ICFP'96) was employed to obtain the sequentiality property for lambda calculus. The infinitary lambda calculus differentiates in three versions according to the way syntactic depth is counted: the three versions yield ordinary lambda calculus with Bohm trees, the lazy lambda calculus with Levy-Longo trees, and a version introduced by Berarducci. For all three versions, Berry's sequentiality theorem is uniformly proved by the same arguments. Also, a strengthening of Berry's theorem was obtained that can be used to prove more undefinability results.

5.1.4 The Theory of Action calculi

In Confer-1, Action structures have been proposed as an algebra for both the syntax and the semantics of interactive computation. *Action calculi* are action structures of a concrete nature, with added structure, and which represent a wide variety of calculi of computation – functional, sequential, and concurrent. Each action in an action calculus is represented as an assembly of molecules; the syntactic binding of names is the means by which molecules are bound together. One action calculus differs from another only in its generators, called controls. The work on action calculi has been pursued this year in Cambridge in the following directions:

- **Labelled transition systems and bisimulation congruences** The semantics of a process calculus, several of which have been introduced during CONFER, can often be most intuitively given as a reduction relation, defined by a structural congruence and reduction axioms. The action calculus framework embraces a rather wide class of such definitions. A central problem is then that of deriving, from such a definition of reduction, a labelled transition semantics for which bisimulation is a satisfactory congruence. Considerable progress has been made, e.g. by Jensen [Jen98] for a form of graph rewriting approximating to a graphical presentation of action calculi, and by Sewell for special cases of term rewriting. Work on more general cases is being undertaken by Gardner, Jensen, Leifer, Milner and Sewell.

- **Expressiveness** A uniform framework for the operational semantics of calculi allows comparisons of expressiveness to be made. Using natural homomorphisms between control structures (the models of action calculi), it is possible to characterise important properties of calculi such as *mobility of channels*. Leifer has shown that the π-calculus is mobile in this sense, whilst many other calculi such as CCS, the λ-calculus, and Petri nets, are not. This work, extending the work of Mifsud at Edinburgh, gives a precise meaning to the intuitive sense in which the π-calculus is more expressive than, say, CCS.

- Gardner has introduced a type-theoretic description of action calculi [GH97] which develops her previous work on a name-free account of action calculi and gives a logical interpretation based on the general binding operators studied by Aczel. With her Ph.D student Hasegawa, at Edinburgh, she has shown that the type-theoretic account of higher-order action calculi corresponds to Moggi's commutative computational λ-calculus. With Gordon Plotkin and Andrew Barber, also at Edinburgh, they have introduced a further extension of action calculi, called the *linear* action calculi [BGHP97]. Linear action calculi extend the higher-order case, and the associated type theory corresponds to the linear type theories studied by Barber and Nick Benton of Cambridge. The conservative extension results follow from looking at the corresponding categorical models.

- **Examples** Several example action calculi have been considered, including a functional programming language with mutable store and exceptions, the ambient calculus of Abadi and Cardelli, and distributed process calculi.

- **Higher-order reflexive action calculi** The first of these examples required two previously separate enrichments of AC, namely, the higher-order and the reflexive extensions (the latter in a simplified form). The proof that an AC combining the two is well-defined reduces to a proof of strong normalisation for the higher-order beta-reduction of the molecular forms corresponding to this enriched AC. Leifer completed this proof and, in the process of doing so, developed several new techniques for handling molecular forms, including the formalisation of manipulations involving reflexive substitutions.

[BGHP97] Andrew Barber, Philippa Gardner, Masahito Hasegawa, and Gordon Plotkin. From action calculi to linear logic,. In *Proceedings of CSL 97, Aarhus*, 1997.

[Gar98] Philippa Gardner. Closed action calculi. *Theoretical Computer Science*, 1998. To appear.

[GH97] Philippa Gardner and Masahito Hasegawa. Types and models in higher-order action calculi. In *Proceedings of TACS 97, Sendai, Japan*, 1997.

[Jen98] Ole Høgh Jensen. PhD thesis, University of Cambridge. Forthcoming, 1998.

5.1.5 The tile model

The tile model relies on certain rewrite rules with side effects, called *tiles*, reminiscent of both Plotkin SOS and Meseguer rewriting logic rules. It provides a logic of concurrent systems with conditional state changes and synchronization, and has been shown to be especially useful for defining compositional models of mobile processes and causal and located concurrent systems. The tile model has been used for modelling coordination formalisms equipped with flexible synchronization primitives. The main advantage of the tile approach is the full compositionality of the underlying logic, which is able to handle computations of (nonclosed) system components as they were new rewrite rules specifying complex coordinators. [FM97a, GM97].

Ferrari and Montanari used tiles to give a tile-based semantics for located CCS in [FM97b]. They give a treatment of concurrent process calculi where the same operational and abstract concurrent semantics described in the literature now descend from general, uniform notions. More precisely, a tile-based semantics for located CCS is shown to be consistent with the ordinary concurrent (via permutation of transitions) and bisimilarity based location semantics. The tile model is particularly well suited for defining directly operational and abstract semantics of concurrent process calculi in a compositional style.

[FM97a] Ferrari, G., Montanari, U. *A Tile-Based Coordination View of Asynchronous π-calculus*, In Proc. MFCS'97, LNCS 1295, 1997.

[FM97b] Ferrari, G., Montanari, U. *Tiles for Concurrent and Located Calculi.* In Proc. EXPRESS'97, ENTCS 7, 1997.

[GM97] Gadducci, F. and Montanari, U., *The Tile Model.* In: Gordon Plotkin, Colin Stirling, and Mads Tofte, Eds., Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, to appear.

5.1.6 Correct Transformations of Nondeterministic Programs

Computing systems have two parts: a *computational* part and a *descriptive* part. The descriptive part specifies the environment in which the computational part is to execute, and it may be nondeterministic (e.g. asynchrony in a system with communicating processes). The computational part is most often deterministic and can be implemented in various ways, which gives opportunities for optimisations.

Traditional program transformations, like fold/unfold-transformations, are however not always correct in the presence of nondeterminism. Yet, there is a need to perform optimising transformations on programs containing an environmental part, especially if the language concepts are very high level (and thus expensive to implement). The motivation for this work is to provide a theory to support the design and implementation of languages where the computational part can be optimised as freely as possible while preserving the (possibly nondeterministic) semantics for the full system. The results are general, but the intended target is recursive languages with process constructs.

If the computational and descriptive parts (both described as rewrite rules) commute, it is however possible to prove the validity of unfold/fold program tranformations. This earlier work within Confer-1 has been completed during this year [Lisp98]. These results hold under the classical condition of "restricted folding-unfolding" [Cou90], which is quite restrictive with respect to the formation of new recursive definitions. We have made some initial investigations towards extending the results to the particular formation of new recursive definitions that typically occur during program specialization.

[Lisp98] B. Lisper. Computing in unpredictable environments: Semantics, reduction strategies, and program transformations. *Theoretical Comput. Sci.*, 190(1):61–85, Jan. 1998.

5.1.7 Flow graphs and semantics

In [Je97], Jeffrey gives a precise relationship between

- *mixed data and control flow graphs*, which are commonly used in software engineering as a semi-formal notation for describing and analysing algorithms
- and an abstract representation of formal semantics which uses partially traced Cartesian categories.

This document also contains some more speculative work on how closed structure (to represent higher-order functions) or two-category structure (to represent operational semantics) might be included in this graphical framework.

[Je97] A. Jeffrey. Flow graphs and semantics of programs. available from http://www.cogs.susx.ac.uk/users/alanje/premon/, 1997.

5.2 Calculi

The work done this year mainly deals with 4 main topics: new calculi for mobile and distributed processes, basic properties and types, bisimulations, proofs of protocols and transformational rules.

5.2.1 New Calculi for mobile and distributed processes

Boudol introduced the *blue calculus* [Bou97a]. It proposes a synthesis of the lambda-calculus and the pi-calculus. This calculus contains a new type system which incorporates both the notion of functional type and that of sort for communication channels. This type system has been further studied by Dal-Zilio in [Dal97], where it is shown to extend naturally to an implicit polymorphic type system à la ML.

These types only express very weak properties of the behaviour of processes. For instance, one cannot guarantee the absence of *deadlock*; here we mean by deadlock that a process calls a resource that will never become available. To overcome this lack of expressiveness, G. Boudol has proposed in [Bou97b] a more refined notion of type, which incorporates a part of Hennessy-Milner logic: a resource declaration is given a modal type indicating that a value of a certain type is available on a certain name. This new type system is shown to enjoy the subject reduction property. This research will be hopefully extended with a proof that typability implies the absence of deadlock.

In [Am97] Amadio presents a model of mobility based on the pi-calculus. He relies on an elementary typing, which ensures that in a system, at every moment, to a given name there corresponds a unique resource (whose state can change from one call to another, however). In this way, one can consider a site – or a locality – as a name associated to a resource whose state is either 'alive' or 'dead'. Amadio develops on this basis a formalism to speak about migration and failures, and he provides reasoning methods which use a notion of bisimulation.

In [RieHen97a] Riely and Hennessy studied in detail location failure for distributed processes. A language is defined in which threads are run at distributed locations, and from this a semantic theory is developed using the basic notions of *observer* and *experiment*. Roughly, an observer interacts with a distributed process either by synchronizing with one of its threads or by halting one of its locations; two programs are equivalent if they present the same structure of observations, evolving over time. The main result of this work is to provide an alternative characterization of this *observational equivalence* which is suitable to automatic verification. This is done using a temporal logic to characterize the interdependence between synchronization events and failures.

In [RieHen97b] Riely and Hennessy present a more expressive language, which includes agent migration, location failure and a rich system of *capabilities* used to tag communicated data. Intuitively, when the name, say, of a location is communicated, a set of capabilities for that location is communicated at the same time; these may include the capability to read data, to write data, to halt the location, to migrate to it, or to allocate fresh resources at it. We present a series of type systems for this language and show, through examples, that very expressive type systems are necessary in order to be able to type many important practical programs.

Lavatelli works on the definition of an algebraic semantics of λ_r based on approximants, in the flat scenario (where both abstractions and deadlock are observable), which turns out to be sound. Work in progress concerns completeness. The aim is to use these results as intermediate steps in the characterization of the flat semantics through an intersection type system. Another work in progress, by Lavatelli and Curien, concerns the use of techniques of logical relations to compare the inequational theories of two models of λ_r that have arisen naturally when trying to give a semantics to λ_r.

Now, Parrow and Victor have also developed the update-calculus where processes can perform *update* actions with side effects, and a scoping operator controls the extent of the effects [PV97]. In this way it incorporates fundamental concepts from imperative languages, concurrent constraints and from functional formalisms. Structurally it is similar to but simpler than the pi-calculus; it has only one binding operator and a symmetry between inpout and output.

Finally, in [SWP97], Sewell, Wojciechowski and Pierce have proposed a simple calculus of agents that can migrate between sites and can interact by both location dependent and location independent message passing. It has a precise reduction semantics. Implementations of the location independent primitives can be expressed as encodings, or compilations, of the whole calculus into the fragment with only location dependent communication. This allows the algorithms to be stated and understood precisely, a clean implementation strategy for prototype languages (such an encoding can be realized as one phase of a compiler), and reasoning. It also provides a setting in which multiple communication mechanisms can coexist gracefully.

[Am97] R. Amadio. An asynchronous model of locality, failure, and process mobility. In *Proc. Coordination 97, Springer Lect. Notes in Comp. Sci. 1282*, 1997.

[AmCou97] R. Amadio and S. Coupet-Grimal. Analysis of a guard condition in type theory (preliminary report). Technical Report TR 1997.245. Also RR-3300 INRIA, Université de Provence (LIM), 1997.

[Bou97a] G. Boudol. The pi-calculus in direct style. In *Proceedings POPL'97*, 1997.

[Bou97b] G. Boudol. Typing the use of resources in a concurrent calculus. In *Proceedings ASIAN'97, LNCS 1345*, 1997.

[Dal97] S. Dal-Zilio. Implicit polymorphic type system for the blue calculus. Technical Report RR-3244, INRIA, 1997.

[RieHen97a] J. Riely and M. Hennessy. Distributed processes and location failures. Technical Report 2/97, 1997. Extended abstract presented at ICALP97.

[RieHen97b] J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *Proc. POPL'98, ACM Conference on Principle of Programming Languages, San Diego.*

[PV97] J. Parrow and B. Victor. The update calculus (full version). Technical Report DoCS 97/93, Department of Computer Systems, Uppsala University, Sweden, Sept. 1997. Extended abstract to appear in the proceedings of AMAST'97.

[SWP97] Peter Sewell, Paweł T. Wojciechowski, and Benjamin C. Pierce. Location independence for mobile agents. Submitted for publication, 1997.

5.2.2 Basic properties and types

In [San97a], Sangiorgi formulates a type system for mobile processes that guarantees, in a distributed process, that at any moment a given name gives the access to a unique stateless resource. This discipline, called uniform receptiveness, is employed, for instance, when modeling functions, objects, higher-order communications, remote-procedure calls. The impact of the discipline on behavioural equivalences and process reasoning is studied. Some theory and proof techniques for uniform receptiveness is developed, and their usefulness on some non-trivial examples is illustrated.

In [PiSa97], Pierce and Sangiorgi investigate parametric polymorphism in message-based concurrent programs, focusing on behavioral equivalences in a typed process calculus analogous to the polymorphic lambda-calculus of Girard and Reynolds. Polymorphism constrains the power of observers by preventing them from directly manipulating data values whose types are abstract, leading to notions of equivalence much coarser than the standard untyped ones. The nature of these constraints is studied through simple examples of concurrent abstract data types and develop basic theoretical machinery for establishing bisimilarity of polymorphic processes. Some surprising interactions between polymorphism and aliasing are also observe, drawing examples from both the polymorphic pi-calculus and ML.

Philippou and Walker looks at confluence properties: in a process setting, the essence of confluence is that the occurrence of one action may not preclude others. In the case of π-calculus, where names may be extruded from their scopes, care is needed to find the right definition. Basic results on confluence in π-calculus are presented in [PW97] and [P97phd]. In particular, conditions under which combinations of confluent, or almost-confluent, processes yield confluent systems are presented.

In [LW97b], a theory of partial-confluence of processes is developed, in CCS and in π-calculus. The main theorem concerns systems which can be thought of as composed from two components: a questioner, and an answerer which may process questions concurrently. It gives conditions under which a system of this kind is behaviourally indistinguishable from the corresponding system in which the answerer may handle only one question at a time. This result, together with a semantic definition of the kind described above, is used to show the interchangeability in an arbitrary program context of two class definitions which prescribe binary-tree symbol-table data structures, one supporting concurrent operations, the other sequential.

In [NS97], Nestmann and Steffen also investigates confluence properties for concurrent systems of message-passing processes. In order to prove confluence for a given system, one has to demonstrate that for all states reachable by computation from the starting state, the 'flowing together' of possible computations is possible. However, it is possible to prove confluence properties for concurrent systems without having to generate all reachable states by use of a type system that supports a static analysis of possible sources of non-confluence. One may statically check the occurrences of critical situations by reducing them to the concurrent access on a shared communication port. For the technical development, one focuses on the setting of a polarized pi-calculus, whith the notion of port-uniqueness being formalised by means of overlapping-free context-redex decompositions.

Finally, in [Sew97a], Sewell considers global/local typing. In the design of distributed programming languages, there is a tension between the implementation cost and the expressiveness of the communication mechanisms provided. In [Sew97a] a static type system for a distributed π-calculus is introduced, in which the input and output capabilities of channels may be either global or local. This may allow compile-time optimization, but retains the expressiveness of channel communication. Subtyping allows all communications to be invoked uniformly.

[LW97b] X. Liu and D. Walker, *Partial confluence of processes and systems of objects*, Theoretical Computer Science B, to appear.

[NS97] U. Nestmann and M. Steffen. Typing confluence. In S. Gnesi and D. Latella, editors, *Second International ERCIM Workshop on Formal Methods in Industrial Critical Systems (Cesena, Italy, July 4–5, 1997)*, pages 77–101. Consiglio Nazionale Ricerche di Pisa, 1997. Also available as report ERCIM-10/97-R052, European Research Consortium for Informatics and Mathematics, 1997.

[PW97] A. Philippou and D. Walker, *On confluence in the π-calculus*, in Proceedings of International Colloquium on Automata, Languages and Programming, Bologna, July 1997, Springer-Verlag Lecture Notes in Computer Science, vol. 1256, 314–324.

[PiSa97] B. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. In *24th POPL*. ACM Press, 1997.

[San97a] D. Sangiorgi. The name discipline of receptiveness. volume 1256 of *Lecture Notes in Computer Science*. Springer Verlag, 1997. Full paper available electronically as `ftp://zenon.inria.fr/meije/theorie-par/-davides/Receptiveness.ps.Z`.

[Sew97c] Peter Sewell. Global/local subtyping for a distributed π-calculus. Technical Report 435, Computer Laboratory, University of Cambridge, August 1997. Available from `http://www.cl.cam.ac.uk/users/pes20/`.

5.2.3 Bisimulations

In [BFL98], Boreale, Fournet and Laneve developed a new theory of bisimulation for the join calculus. They consider open terms, and define two bisimulations, weak and asynchronous, both refinments of the observational equivalence. This gives a more compositional sense to the interaction of the terms of the join-calculus with contexts, and is a start point to logical systems for reasoning on join-terms.

More generally, many equivalences have been proposed for process algebra for the s-tudy of convergence, security or fairness. Gonthier started a comparison of equivalences used both in the pi and join calculus, mostly based on reduction semantics, observation of barbs and preservation by contexts. Many variants appear according to the way these elementary constructs are combined. But fortunately many lead to a same result. One shows that fair-testing is connected to coupled bisimulation, or that omega-limit bisimulation is also related to single-barb gfp bisimulation. This provides new proof techniques and a better understanding of process calculi. An article is in preparation, the work has been presented at the Confer Workshop in Amsterdam [Gon98].

Sewell has studied the semantics of PICT in [Sew97c], and has shown how the behaviour of implementations of the language relates precisely to the labelled transitions of the π-calculus. His results indicate that there is an essentially unique behavioural equivalence appropriate for PICT, from the large space of choices in the literature.

In [Ra97a] an axiomatic characterisation of both late and early observational congruence for a class of fully parameterised processes is given. This result uses a new derivation rule which is an intuitive generalisation of unique fixpoint induction based on loop invariants for symbolic graphs. In the same paper it is shown that the new rule can actually be derived from existing formulations of unique fixpoint induction for value-passing calculi.

In [Ra97b] a first order modal mu-calculus is described which uses parameterised maximal fixpoints to describe safety and liveness properties of value-passing processes. A proof system for deciding if a process satisfies a formula is presented. Certain deduction rules of the system carry side conditions which leave auxiliary proof obligations of checking properties of the value language. Although the proof system is in general incomplete two completeness results are given for significant sub-logics.

Abadi (DEC/SRC), Fournet and Gonthier studied secure implementation of communication in the join calculus. It is easy to write secure protocols when lexical scope is guaranteed, but one needs cryptographic means to enforce a secure lexical scope discipline. By compositionality, two protocols can be considered: one based on the redefinition of secure emission/reception primitives, another one based on firewalls for some locations. The correctness proofs use bisimulation techniques. A presentation by Fournet has been made at the DARPA workshop on *Foundations for Secure Mobile Code*, March 26-28 mars in Monterey [Fou97], and at the Confer meeting in Amsterdam. An article is in preparation [AFG98].

[AFG98] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. To be submitted for a conference, November 1997.

[BFL98] M. Boreale, C. Fournet, and C. Laneve. Bisimulations in the join-calculus. Submitted for a conference, September 1997.

[Fou97] C. Fournet. Security within a calculus of mobile agents? April 1997, http://www.cs.nps.navy.mil/research/languages/statements/fournet.ps.

[Gon98] G. Gonthier. On asynchronous barbs, simulations, and fair testing. Submitted to a conference, Octobre 1997.

[Ra97a] J. Rathke. Unique fixpoint induction for value-passing processes. In *Proc. LICS'97,* 12th *Annual Symposium on Logic in Computer Science, Warsaw.* IEEE Computer Society Press, 1997. To appear.

[Ra97b] J. Rathke and M. Hennessy. Local model checking for value-passing processes. In *Proc. TACS'97, International Symposium on Theoretical Aspects of Computer Software, Sendai.* Springer-Verlag, 1997.

[Sew97c] Peter Sewell. On implementations and semantics of a concurrent programming language. In *Proceedings of CONCUR '97. LNCS 1243*, pages 391–405. Springer-Verlag, 1997.

5.2.4 Proofs of protocols and transformational rules

In [San97J], Sangiorgi applies the proof techniques developed in [San97a] to solve a problem submitted by Cliff Jones. Jones has proposed transformations between concrete programs and general transformation rules that increase concurrency in a system of objects, and has raised the challenge of how to prove their validity. A proof of correctness of the hardest of Jones's concrete transformations is presented.

In [PW97b], Philippou and Walker use the technique based on partial confluence of processes to show the soundness of transformation rules for concurrent object-oriented programs which increase the scope for concurrent activity within the system prescribed by a program, without altering its observable behaviour. The rules allow one object to release another from a rendez-vous before it has finished all the associated activity, and allow one object to delegate to a second object the responsibility for returning a result to a third object.

In [PW97c], the theory of [PW97b] is extended to encompass a larger class of answerers. Analogous behavioural indistinguishability results are obtained, for an appropriate class of questioners. The main advance is that whereas in the case of the symbol-table data structures, the answer to a question is determined at the moment the question is submitted, in [PW97c] up to one state-changing internal action may occur in determining the answer. This result is used in [PW97d] where algorithms for operating concurrently on a variant of the B-tree data structure are modelled and analysed using π-calculus. In this case the state-changing internal action is the act of locking the disk page in which the operation in question will take effect. Deficiences in and improvements to published algorithms were discovered. In [I97] Ionitoiu begins to explore a new approach to modelling asynchronous communication and fault-tolerance in a process-calculus setting.

Another work on mobility [AmPra97] concerns the modelling of *mobile hosts* on a network in a simple name-passing process calculus. The protocol considered is a highly simplified version of proposals for *mobility support* in the version 6 of the Internet Protocols (IP). We concentrate on the issue of ensuring that messages to and from mobile agents are delivered without loss of connectivity. We provide three models of increasingly complex nature of a network of routers and computing agents that are interconnected via the routers: the first is without mobile agents and is treated as a specification for the next two; the second supports mobile agents, and the third additionally allows correspondent agents to *cache* the current location of a mobile agent. Following a detailed analysis of the three models to extract invariant properties, we show that the three models are related by a suitable notion of equivalence based on *bisimulation*.

[AmPra97] R. Amadio and S. Prasad. Modelling IP mobility. Technical Report TR 1997.244. Also RR-3301 INRIA, Université de Provence (LIM), 1997.

[San97J] D. Sangiorgi. Typed π-calculus at work: a proof of Jones's parallelisation transformation on concurrent objects. Presented at the Fourth Workshop on Foundations of Object-Oriented Languages (FOOL 4). To appear in *Theory and Practice of Object-oriented Systems*, 1997.

[PW97b] A. Philippou and D. Walker, *On transformations of concurrent object programs*, Theoretical Computer Science B, to appear.

[PW97c] A. Philippou and D. Walker, *Social confluence in client-server systems*, in Proceedings of Computer Science Logic, Utrecht, September 1996, Springer-Verlag Lecture Notes in Computer Science, vol. 1258, 385–398.

[PW97d] A. Philippou and D. Walker, *A rigorous analysis of concurrent operations on B-trees*, in Proceedings of Concurrency Theory, Warsaw, July 1997, Springer-Verlag Lecture Notes in Computer Science, vol. 1243, 361–375.

[P97phd] A. Philippou, *Reasoning about systems with evolving structure*, PhD thesis, University of Warwick, December 1996.

[I97] C. Ionitoiu, *An operational semantics for loosely coupled systems*, MSc dissertation, University of Warwick, September 1997.

5.3 Logics for Concurrency and λ-calculus

In this section, works has been carried out on semantic models of the λ-calculus, possibly extended by some δ-rules, and of linear logic. Denotational semantics is also used to provide models for concurrent languages and infinite streams. Research has been mainly conducted at Edinburgh, ENS, INRIA Sophia-Antipolis and Sussex.

5.3.1 Game semantics for the λ-calculus and linear logic

A key interest of game semantics is that it provides a framework within which one may conciliate dynamic and static aspects of meaning. It is expected that such semantics can help in complexity theory on the one hand and in extending denotational semantics to model concurrent and distributed languages (so-called process-calculi).

Abramsky has developed a concurrent version of game semantics, using the Kahn-Plotkin theory of concrete domains and sequential functions. This has been used to give a game semantics for Classical Linear Logic. It has also been used to give a model for logic with branching quantifiers and information hiding, as proposed by Hintikka. This work was presented in invited talks given at the Workshop on "Problems and Advance in the Semantics of Linear Logic" held in Utrecht, November 28th and 29th 1997, and at the Amsterdam Colloquium, December 17-20 1997.

Honda and Yoshida developed a version of game semantics for call-by-value PCF, and proved a full abstraction result. Their description of their model used some tools from the pi-calculus. Abramsky and McCusker gave a more abstract and general construction of call-by-value models from call-by-name ones, and showed that when applied to the call-by-name games model of PCF it gave rise to the call-by-value one of Honda and Yoshida; when applied to the model for the richer functional language FPC studied by McCusker in his thesis, it gave a fully abstract model for call-by-value FPC; and when applied to the model previously given by them for Idealized Algol, it gave a full abstract model for a fragment of ML with ground reference types. Abramsky, Honda and McCusker have obtained a fully abstract games model for general reference types; a paper containing this result has been submitted to LiCS '98.

Danos, Herbelin and Regnier explain, in [DHR96], the operational side of the two game models of Hyland-Ong (HO) and Abramsky-Jagadeesan-Malacaria (AJM) in terms of abstract machines. Building on this explanation, an embedding of the AJM model in the HO model is given that leads to an easy proof of completeness for the former.

In [BDER97], Baillot, Danos, Ehrhard and Regnier adapt the AJM construction to give the first known games model which encompasses all of classical linear logic. Composition of strategies is recast as a communication process. The exponential part of the model is based on the idea of sticking to strategies which contain all possible encodings of copy-tags. This works is pursued in [BDER97b], which gives an explicit transformation that goes from that linear logic games model to the traditional relational model. This concrete extensional collapse helps in understanding what games semantics is good to.

In [CuBoeh] Curien presents a formalism of trees with pointers, called abstract Böhm trees, that provide a suitable abstract framework in which various cut-free proofs or normal terms of several λ-calculus based languages (including PCF and Parigot's $\lambda\mu$-calculus) can be faithfully encoded. A simple abstract machine allows us to compute over abstract Böhm trees. This machine is closely related to Coquand's interaction sequences and debates. The execution over finite abstract Böhm trees always terminates. We introduce an abstract notion of type that fits the purpose of guaranteeing that the execution cannot go into deadlock, i.e., that it always reaches a satisfactory final state. The operational aspects of (untyped) Böhm trees are investigated in more depth in [CuHer96], a joint paper by Curien and Herbelin,

[AM97] S. Abramsky, G. McCusker, *Call-by-Value games (extended abstract)* To appear in Proc. of CSL'97.

[BDER97] P. Baillot, V. Danos, and T. E. andL. Regnier. Believe it or not, AJM's games model is a model of classical linear logic. In *Proc. 12th LICS*, pages 68–75. IEEE, 1997.

[BDER97b] P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Timeless games. In *Proc. CSL '97*, volume to appear. Springer Verlag, 1997.

[CuBoeh] P.-L. Curien. Abstract böhm trees. *Mathematical Structures in Computer Science*, 1998. to appear.

[CuHer96] P.-L. Curien and H. Herbelin. Computing with abstract böhm trees. In *Proc. of Third Fuji International Symposium on Functional and Logic Programming, World Scientific (Singapour)*, 1998.

[DHR96] V. Danos, H. Herbelin, and L. Regnier. Game semantics and abstract machines. In *Proc. 11th LICS*, pages 394–405. IEEE, 1996.

[HY97] K. Honda, N. Yoshida, *Game-theoretic analysis of Call-by-Value Computation (extended abstract).* To appear in Proc. of ICALP'97.

5.3.2 Denotational semantics

In [AmCou97] Amadio and Coupet-Grimal consider the issue of representing "infinite" and "total" objects like infinite streams or non-terminating processes in type theory. They present a realizability interpretation of co-inductive types based on *partial equivalence relations* (per's), and extract from the per's interpretation sound rules to type recursive definitions. These recursive definitions are needed to introduce 'infinite' objects of co-inductive type. One can show that the proposed type system enjoys the basic syntactic properties of subject reduction and strong normalization with respect to a confluent rewriting system first studied by Gimenez. The proposed type system is compared with those by Coquand and Gimenez. In particular, it provides a semantic reconstruction of Gimenez's system which suggests a rule to type nested recursive definitions.

The final version of the book [AmCu97] has been delivered. The book addresses the mathematical aspects of the semantics of programming languages. The mathematical objects manipulated are order-theoretic structures, and the languages we consider are typed lambda-calculi. The book considers various extensions of typed lambda-calculi to include non-determinism, communication, and concurrency, and it summarizes some of the work carried on in the Confer project.

In [HaHen97e] Hartonas and Hennessy use traditional denotational techniques to give a denotational model of a subset of *core Facile*. Specifically this is applied typed call-by-value language which in addition to the usual types has an extra type for processes, which can exchange values along communication channels, a la *CCS*. The model is shown to be fully-abstract with respect to a version of *may testing*. It is notable that in order to achieve full-abstraction it is necessary to add to the language an operator which allows processes report the result of a computation.

[Am97] R. Amadio. An asynchronous model of locality, failure, and process mobility. In *Proc. Coordination 97, Springer Lect. Notes in Comp. Sci. 1282*, 1997.

[AmCou97] R. Amadio and S. Coupet-Grimal. Analysis of a guard condition in type theory (preliminary report). Technical Report TR 1997.245. Also RR-3300 INRIA, Université de Provence (LIM), 1997.

[AmCu97] R. Amadio and P.-L. Curien. *Domains and Lambda-Calculi.* Cambridge University Press. To appear.

[AmPra97] R. Amadio and S. Prasad. Modelling IP mobility. Technical Report TR 1997.244. Also RR-3301 INRIA, Université de Provence (LIM), 1997.

[HaHen97e] C. Hartonas and M. Hennessy. Full abstractness for a functional/concurrent language with higher-order value-passing. Full version available as Computer Science Technical Report 1/97, University of Sussex, 1997. Available from http://www.cogs.susx.ac.uk/.

5.4 Programming Languages

This section is composed of 4 parts. First, there is the work on the Join Calculus language and PICT, both with public releases. Secondly, the work on the BOHM machine has been pursued. Thirdly, a prototype of LLinda has been implemented. Finally, practical experiments have been designed in the ICL GTD project and at CNET Lannion. These researchs have been carried out at INRIA Rocquencourt, Bologna, Pisa, ICL and CNET.

5.4.1 The Join Calculus Language

The join-calculus has been designed in BRA Confer-1 by Fournet and Gonthier. It is a variant of the PICT language, specially defined for a distributed implementation on any Unix-like environment. The basic difference with respect to PICT is to impose a single location for receptors on any communication channel.

Version 1.04 has been released in June by Fournet and Maranget. This prototype includes a distributed implementation of the join-calculus, which is about 20000 lines of Caml-light code. The system comprises a byte-code compiler, a run-time interpreter, an on-line documentation, a tutorial and many examples. As demo programs, we have a distributed calculation of the Mandelbrot set and several distributed games. In this release of the join-calculus, the syntax of the language is very similar with the one of Caml, and there is a type-inference system including a polymorphic typing discipline a la ML, a natural interface system for CAML-light, which gives access to the graphics library. Failures have also been implemented, but only the voluntary ones. True asynchronous permanent failures will be implemented in a future release.

An article on the type-inference algorithm and a demo were presented at CONCUR'97 [FLMR97] and at TACS'97 [Le97]. The system is available at http://www.-pauillac.inria/join/.

5.4.2 A distributed garbage collector

Le Fessant, Piumarta, and Shapiro expanded an acyclic distributed garbage collector (the cleanup protocol of Stub-Scion Pair Chains) with a detector of distributed cycles of free cells. The whole results in a complete and asynchronous distributed garbage collector.

The detection algorithm for free cycles is inspired by Hughes. A local collector marks outgoing references with dates, which are propagated asynchronously between spaces. A central server computes the minimal allowed dates, permitting cycles to be detected, cut and further collected.

This new algorithm is asynchronous and fault-tolerant. Furthermore, it can be adapted to large-scale systems. It also requires few resources and is easy to implement. In the mid-range, it will be included in a future release of the Join Calculus language.

[FLMR97] C. Fournet, C. Laneve, L. Maranget, and D. Rémy. Implicit typing à la ML for the join-calculus. In A. Mazurkiewicz and J. Winkowski, editors,

CONCUR '97: Concurrency Theory (8th International Conference, Warsaw, Poland, July 1997), volume 1243 of *LNCS*, pages 196–212. Springer, 1997.

[leF97] F. le Fessant, I. Piumarta, and M. Shapiro, An implementation for complete asynchronous distributed garbage collection. Submitted for publication, November 1997, available electronically (`http://www-sor.inria.fr/docs/`).

[FouMa97] C. Fournet and L. Maranget. The join-calculus language. Available electronically (`http://pauillac.inria.fr/join`), June 1997.

[Le97] J.-J. Lévy. Some results in the join-calculus. In M. Abadi and T. Ito, editors, *TACS '97: Theoretical Aspects of Computer Software*, volume 1281 of *LNCS*, pages 233–249. Springer, 1997.

5.4.3 Distributed PICT

PICT is a language, based on the π-calculus, designed by Pierce and Turner. PICT now has a stable public release; its design is described in [PT97]. Pierce, has moved from Cambridge to Indiana. Sewell, Wojciechowski, and Pierce (see section 5.2), work on a study in more detail of the design, semantic definition and implementation of communication primitives by which mobile agents can interact. This has led to a prototype implementation, building on that of PICT, which is ongoing, as is work on reasoning, in particular on correctness and robustness results.

PICT has also been used to model concurrent objects. The view that general models of mobile processes, such as π-calculus and Higher-Order π-calculus, provide a good framework for the definition of a variety of concurrent object-oriented languages is illustrated in [LW97a]. A technique involving process continuations is used to give a natural and direct semantic definition for a language containing constructs for several kinds of inter-object communication. In the unpublished MSc dissertation [S97], Sawle gives a semantic definition to an experimental extension of Pict with actor primitives and implements a prototype. Sawle now studies for a PhD at Cambridge.

[LW97a] X. Liu and D. Walker, *Concurrent objects as mobile processes*, in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, G. Plotkin, C. Stirling, and M. Tofte (eds.), MIT Press, to appear.

[PT97] Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. Technical Report CSCI 476, Computer Science Department, Indiana University, 1997. To appear in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, MIT Press.

[S97] M. Sawle, *A π-calculus semantics for actor systems*, MSc dissertation, University of Warwick, September 1997.

5.4.4 The BOHM machine

Asperti, Giovannetti, and Naletto are still working on the implementation of Lamping's graph reduction algorithm for optimal reduction of functional languages (the Bologna Optimal Higher-order Machine [AG98]). We have almost completed a new "light" version, essentially inspired by Girard's Light Linear Logic. The *light* version provides a much sharper syntactical control over the shape of sharing graphs than that provided by the rough encoding of intuitionistic implication into linear logic, with a significant impact on the efficiency of the reduction.

BOHM is a simple interpreter written in C, and it runs on most systems having a C compiler, including Personal Computers. BOHM's source language constitutes the full core of a typical (dynamically-typed) lazy functional language, including a few datatypes, conditional expressions, recursive values and multiple recursion and lazy pattern matching.

BOHM's performance has been compared with two standard and fully developed implementations of functional languages: Caml-light (Rocquencourt, France) and Yale Haskell (UK). It is always comparable with that of Yale Haskell (i.e. approximately ten time slower than Caml-light), while it can run surprisingly faster on many interesting examples. Two typical cases are the incremental modification of arrays represented as functions from indices to values, and the direct evaluation of Denotational Semantics for imperative languages.

[BOHM] Asperti, A., Giovannetti, C., Naletto, A.: "The Bologna Optimal Higher-order Machine". Internal Report UBLCS-95-9, Laboratory for Computer Science, University of Bologna, Italy.

[AG98] A. Asperti, S. Guerrini *The Optimal Implementation of Functional Programming Languages.* To appear in the *"Cambridge Tracts in Theoretical Computer Science"* Series, Cambridge University Press, 1998.

5.4.5 LLinda

De Nicola, Ferrari, and Pugliese design a kernel programming language, called `LLinda`, to support a programming paradigm where processes as well as data can migrate from one computing environment to another. The language is equipped with a sophisticated type system.

LLinda (Locality based Linda) is a variant of the Linda system of David Gelernter, which supports a programming paradigm where agents can migrate from one computing environment to another. It is possible to define a type system for LLinda that permits statically checking access rights violations of mobile agents. Types are used to describe processes intentions (read, write, execute, ...) relatively to the different localities they are willing to interact with or they want to migrate to. The type system is used to determine the operations that processes want to perform at each locality, to check whether they comply with the declared intentions and whether they have the necessary rights to perform the intended operations at the specific localities.

[NFP97] De Nicola, R., Ferrari, G., Pugliese, R., Coordinating Mobile Agents via Blackboards and Access Rights, In Proc. COORDINATION'97, LNCS 1282, 1997.

5.4.6 Practical uses of Agents and Mobility

L. Leth and B. Thomsen worked on designing programming languages and advanced end-user applications, to be serving as a "laboratory" in which theoretical results are transferred into industrial usage.

A new research programme, called the ICL GTD (Group Technical Directorate) framework in Agents and Mobility, has been set up. The programme encourages ICL businesses to take part in the research process from an early stage to ensure they are prepared for knowledge and technology transfer as results emerge. The framework includes setting up a virtual laboratory in collaboration with Fujitsu Laboratories, coordinating sponsored medium/longer term research at universities and external research laboratories (ANSA at APM, Cambridge and IC-Parc at Imperial College), a number of research and development projects with ICL businesses (TeamWare, Financial Services, ProcessWise and DAIS) and coordinating and participating in ICL efforts in defining market and technical strategies, architectural and methodological developments across the company.

There have been activities on understanding the relationship between mobile agents, higher order processes, Milner's action structures, distributed concurrent functional programming languages, such as Facile and Business Process Modeling, in particular the POSD formalism developed by ICL ProcessWise. Ppreliminary results may be viewed on the web (http://www.prattens.demon.co.uk/webposd/posdw.htm).

At CNET Lannion, Brisset started to evaluate the usefulness of the *join-calculus language* for real-world applications. He takes advantage of the features of the join-calculus to design a very primitive distributed search engine that indexes text files residing on a network of workstations. Each workstation runs an autonomous agent which first builds an indexed database of the local files, then answers keyword-based queries while continuously updating the database. The only central element is a name-server whose function is mainly to transmit queries and answers between these agents and client processes. Compared to popular WWW search engines like AltaVista, which fetch files from servers through the Internet and process them in a high-performance, centralized database server, this distributed architecture yields lower network usage (assuming low query rates) and allows the database to be kept consistent with the original files at a much lower cost.

[LT97a] Leth Thomsen, L., and Thomsen, B.: "Mobile Agents – THE new paradigm in computing", ICL Systems Journal, Volume 12, Issue 1, pp. 14–40, May 1997.

[LT97b] Thomsen, B.: "Programming Languages, Analysis Tools and Concurrency Theory", ACM Computing Surveys *28A*(4), December 1996. (http://www.acm.org/-pubs/articles/journals/surveys/1996-28-4es/a57-thomsen/a57-thomsen.html)

5.4.7 Proofs systems

Ferrari Ferro, Gnesi, Montanari, and Pistore, developed an automata-based verification environment for the π-calculus. The environment takes a direct advantage of a general theory which allows to associate ordinary finite state automata to a wide class of π-calculus agents, so that equivalent automata are associated to equivalent agents. A key feature of the approach is the re-use of efficient algorithms and verification techniques (equivalence checkers and model checkers) which have been developed and implemented for ordinary automata. The verification environment has been implemented on top of an existing verification environment for process calculi: the JACK environment.

CNET is also involved in adapting the Coq theorem prover to specify and prove safety and liveness properties of protocols and of distributed algorithms in a modular way. This works implements results already obtained in 1996 by Crégut and Heyd with the theory of Unity. In order to apply Coq-Unity to real-world protocol, a translation from SDL is used. This method is currently applied on an ATM protocol, and is research in progress.

Chapter 6

Appendices

[AM98] A. Asperti, H.Mairson *Optimal β-reduction is not elementary recursive.* Proc. of the twenty-fifth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'98).

[AG98] A. Asperti, S. Guerrini *The Optimal Implementation of Functional Programming Languages.* To appear in the *"Cambridge Tracts in Theoretical Computer Science"* Series, Cambridge University Press, 1998.

[BFL97] M. Boreale, C. Fournet, C. Laneve: *Bisimulations in the Join-Calculus.* October 1997. Submitted.

[FLMR97] C. Fournet, C. Laneve, L. Maranget, D. Rèmy: *Implicit Typing à la ML in the Join-Calculus.* In Proceeding CONCUR '97, LNCS 1243.

[BGHP97] Andrew Barber, Philippa Gardner, Masahito Hasegawa, and Gordon Plotkin. From action calculi to linear logic,. In *Proceedings of CSL 97, Aarhus*, 1997.

[Gar98] Philippa Gardner. Closed action calculi. *Theoretical Computer Science*, 1998. To appear.

[GH97] Philippa Gardner and Masahito Hasegawa. Types and models in higher-order action calculi. In *Proceedings of TACS 97, Sendai, Japan*, 1997.

[Jen98] Ole Høgh Jensen. PhD thesis, University of Cambridge. Forthcoming, 1998.

[PT97] Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. Technical Report CSCI 476, Computer Science Department, Indiana University, 1997. To appear in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, MIT Press.

[Sew97a] Peter Sewell. Global/local subtyping for a distributed π-calculus. Technical Report 435, Computer Laboratory, University of Cambridge, August 1997. Available from `http://www.cl.cam.ac.uk/users/pes20/`.

[Sew97b] Peter Sewell. Nonaxiomatisability of equivalences over finite state processes. *Annals of Pure and Applied Logic*, 1997. To appear.

[Sew97c] Peter Sewell. On implementations and semantics of a concurrent programming language. In *Proceedings of CONCUR '97. LNCS 1243*, pages 391–405. Springer-Verlag, 1997.

[SWP97] Peter Sewell, Paweł T. Wojciechowski, and Benjamin C. Pierce. Location independence for mobile agents. Submitted for publication, 1997.

[AM97] S. Abramsky, G. McCusker, *Call-by-Value games (extended abstract)* To appear in Proc. of CSL'97.

[HY97] K. Honda, N. Yoshida, *Game-theoretic analysis of Call-by-Value Computation (extended abstract).* To appear in Proc. of ICALP'97.

[M97] P.-A. Melliès, *A factorisation theorem in Rewriting Theory.* Proc. of the Category Theory and Computer Science symposium (CTCS'97).

[BDER97] P. Baillot, V. Danos, and T. E. andL. Regnier. Believe it or not, AJM's games model is a model of classical linear logic. In *Proc.* 12^{th} *LICS*, pages 68–75. IEEE, 1997.

[BDER97b] P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Timeless games. In *Proc. CSL '97*, volume to appear. Springer Verlag, 1997.

[CuBoeh] P.-L. Curien. Abstract böhm trees. *Mathematical Structures in Computer Science*, 1998. to appear.

[CuHer96] P.-L. Curien and H. Herbelin. Computing with abstract böhm trees. In *Proc. of Third Fuji International Symposium on Functional and Logic Programming, World Scientific (Singapour)*, 1998.

[DHR96] V. Danos, H. Herbelin, and L. Regnier. Game semantics and abstract machines. In *Proc.* 11^{th} *LICS*, pages 394–405. IEEE, 1996.

[LT97a] Leth Thomsen, L., and Thomsen, B.: "Mobile Agents – THE new paradigm in computing", ICL Systems Journal, Volume 12, Issue 1, pp. 14–40, May 1997.

[LT97b] Thomsen, B.: "Programming Languages, Analysis Tools and Concurrency Theory", ACM Computing Surveys *28A*(4), December 1996. (http://www.acm.org/pubs/articles/journals/surveys/1996-28-4es/a57-thomsen/a57-thomsen.html)

[AFG98] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. To be submitted for a conference, November 1997.

[BFL98] M. Boreale, C. Fournet, and C. Laneve. Bisimulations in the join-calculus. Submitted for a conference, September 1997.

[leF97] F. le Fessant, I. Piumarta, and M. Shapiro, An implementation for complete asynchronous distributed garbage collection. Submitted for publication, November 1997, available electronically (`http://www-sor.inria.fr/docs/`).

[Fou97] C. Fournet. Security within a calculus of mobile agents? April 1997, http://www.cs.nps.navy.mil/research/languages/statements/fournet.ps.

[FLMR97] C. Fournet, C. Laneve, L. Maranget, and D. Rémy. Implicit typing à la ML for the join-calculus. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR '97: Concurrency Theory (8th International Conference, Warsaw, Poland, July 1997)*, volume 1243 of *LNCS*, pages 196–212. Springer, 1997.

[FouMa97] C. Fournet and L. Maranget. The join-calculus language. Available electronically (`http://pauillac.inria.fr/join`), June 1997.

[Gon98] G. Gonthier. On asynchronous barbs, simulations, and fair testing. Submitted to a conference, Octobre 1997.

[Le97] J.-J. Lévy. Some results in the join-calculus. In M. Abadi and T. Ito, editors, *TACS '97: Theoretical Aspects of Computer Software*, volume 1281 of *LNCS*, pages 233–249. Springer, 1997.

[Ne97a] U. Nestmann. What is a 'good' encoding of guarded choice? In C. Palamidessi and J. Parrow, editors, *EXPRESS '97: Expressiveness in Concurrency (Santa Margherita Ligure, Italy, September 8–12, 1997)*, volume 7 of *ENTCS*. Elesevier Science Publishers, 1997.

[NePie97] U. Nestmann and B. C. Pierce. Decoding choice encodings. In U. Montanari and V. Sassone, editors, *CONCUR '96: Concurrency Theory (7th International Conference, Pisa, Italy, August 1996)*, volume 1119 of *LNCS*, pages 179–194. Springer, 1996. Revised full version as report ERCIM-10/97-R051, European Research Consortium for Informatics and Mathematics, 1997.

[NS97] U. Nestmann and M. Steffen. Typing confluence. In S. Gnesi and D. Latella, editors, *Second International ERCIM Workshop on Formal Methods in Industrial Critical Systems (Cesena, Italy, July 4–5, 1997)*, pages 77–101. Consiglio Nazionale Ricerche di Pisa, 1997. Also available as report ERCIM-10/97-R052, European Research Consortium for Informatics and Mathematics, 1997.

[Am97] R. Amadio. An asynchronous model of locality, failure, and process mobility. In *Proc. Coordination 97, Springer Lect. Notes in Comp. Sci. 1282*, 1997.

[AmCou97] R. Amadio and S. Coupet-Grimal. Analysis of a guard condition in type theory (preliminary report). Technical Report TR 1997.245. Also RR-3300 INRIA, Université de Provence (LIM), 1997.

[AmCu97] R. Amadio and P.-L. Curien. *Domains and Lambda-Calculi.* Cambridge University Press. To appear.

[AmPra97] R. Amadio and S. Prasad. Modelling IP mobility. Technical Report TR 1997.244. Also RR-3301 INRIA, Université de Provence (LIM), 1997.

[Bou97a] G. Boudol. The pi-calculus in direct style. In *Proceedings POPL'97*, 1997.

[Bou97b] G. Boudol. Typing the use of resources in a concurrent calculus. In *Proceedings ASIAN'97, LNCS 1345*, 1997.

[Dal97] S. Dal-Zilio. Implicit polymorphic type system for the blue calculus. Technical Report RR-3244, INRIA, 1997.

[PiSa97] B. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. In *24th POPL.* ACM Press, 1997.

[San97a] D. Sangiorgi. The name discipline of receptiveness. volume 1256 of *Lecture Notes in Computer Science.* Springer Verlag, 1997. Full paper available electronically as `ftp://zenon.inria.fr/meije/theorie-par/-davides/Receptiveness.ps.Z`.

[San97J] D. Sangiorgi. Typed π-calculus at work: a proof of Jones's parallelisation transformation on concurrent objects. Presented at the Fourth Workshop on Foundations of Object-Oriented Languages (FOOL 4). To appear in *Theory and Practice of Object-oriented Systems*, 1997.

[Cou90] B. Courcelle. Recursive applicative program schemes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 9, pages 459–492. Elsevier Science Publishers B. V., 1990.

[Lisp98] B. Lisper. Computing in unpredictable environments: Semantics, reduction strategies, and program transformations. *Theoretical Comput. Sci.*, 190(1):61–85, Jan. 1998.

[PV97] J. Parrow and B. Victor. The update calculus (full version). Technical Report DoCS 97/93, Department of Computer Systems, Uppsala University, Sweden, Sept. 1997. Extended abstract to appear in the proceedings of AMAST'97.

[NFP97a] De Nicola, R., Ferrari, G., Pugliese, R., Locality Based Linda: Programming with Explicit Localities, In Proc. TAPSOFT'97, LNCS 1214, 1997.

[NFP97b] De Nicola, R., Ferrari, G., Pugliese, R., Coordinating Mobile Agents via Blackboards and Access Rights, In Proc. COORDINATION'97, LNCS 1282, 1997.

[FFGMPR] Ferrari, G., Ferro, G., Gnesi, S., Montanari, U., Pistore, M., Ristori, G., An automata based verification environment for mobile processes, In Proc. TACAS'97, LNCS 1217, 1997.

[FM97a] Ferrari, G., Montanari, U. *A Tile-Based Coordination View of Asynchronous π-calculus*, In Proc. MFCS'97, LNCS 1295, 1997.

[FM97b] Ferrari, G., Montanari, U. *Tiles for Concurrent and Located Calculi.* In Proc. EXPRESS'97, ENTCS 7, 1997.

[GM97] Gadducci, F. and Montanari, U., *The Tile Model.* In: Gordon Plotkin, Colin Stirling, and Mads Tofte, Eds., Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, to appear.

[MP97] Montanari, U. and Pistore, M., Minimal Transition Systems for History-Preserving Bisimulation. In Proc. STACS'97, LNCS 1200, 1997.

[BM97] Bruni, R., Montanari, U., Zero-Safe Nets, or Transition Synchronization Made Simple In Proc. EXPRESS'97, ENTCS 7, 1997.

[HaHen97e] C. Hartonas and M. Hennessy. Full abstractness for a functional/concurrent language with higher-order value-passing. Full version available as Computer Science Technical Report 1/97, University of Sussex, 1997. Available from http://www.cogs.susx.ac.uk/.

[Je97] A. Jeffrey. Flow graphs and semantics of programs. available from http://www.cogs.susx.ac.uk/users/alanje/premon/, 1997.

[Ra97a] J. Rathke. Unique fixpoint induction for value-passing processes. In *Proc. LICS'97,* 12*th* *Annual Symposium on Logic in Computer Science, Warsaw.* IEEE Computer Society Press, 1997. To appear.

[Ra97b] J. Rathke and M. Hennessy. Local model checking for value-passing processes. In *Proc. TACS'97, International Symposium on Theoretical Aspects of Computer Software, Sendai.* Springer-Verlag, 1997.

[RieHen97a] J. Riely and M. Hennessy. Distributed processes and location failures. Technical Report 2/97, 1997. Extended abstract presented at ICALP97.

[RieHen97b] J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *Proc. POPL'98, ACM Conference on Principle of Programming Languages, San Diego.*

[PW97] A. Philippou and D. Walker, *On confluence in the π-calculus*, in Proceedings of International Colloquium on Automata, Languages and Programming, Bologna, July 1997, Springer-Verlag Lecture Notes in Computer Science, vol. 1256, 314–324.

[LW97a] X. Liu and D. Walker, *Concurrent objects as mobile processes*, in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, G. Plotkin, C. Stirling, and M. Tofte (eds.), MIT Press, to appear.

[S97] M. Sawle, *A π-calculus semantics for actor systems*, MSc dissertation, University of Warwick, September 1997.

[LW97b] X. Liu and D. Walker, *Partial confluence of processes and systems of objects*, Theoretical Computer Science B, to appear.

[PW97b] A. Philippou and D. Walker, *On transformations of concurrent object programs*, Theoretical Computer Science B, to appear.

[PW97c] A. Philippou and D. Walker, *Social confluence in client-server systems*, in Proceedings of Computer Science Logic, Utrecht, September 1996, Springer-Verlag Lecture Notes in Computer Science, vol. 1258, 385–398.

[PW97d] A. Philippou and D. Walker, *A rigorous analysis of concurrent operations on B-trees*, in Proceedings of Concurrency Theory, Warsaw, July 1997, Springer-Verlag Lecture Notes in Computer Science, vol. 1243, 361–375.

[P97phd] A. Philippou, *Reasoning about systems with evolving structure*, PhD thesis, University of Warwick, December 1996.

[I97] C. Ionitoiu, *An operational semantics for loosely coupled systems*, MSc dissertation, University of Warwick, September 1997.