CONFER

CONcurrency and Functions:

Evaluation and Reduction

Basic Research Action

Project Number: 6454

Periodic Progress Report
October 7, 1994

Contents

Chapter 1

Overview

This report contains the Second Periodic Progress Report for ESPRIT BRA Nr. 6454 (CONFER).

The report contains 4 main parts: management at consortium level and at each site, deliverables (programs of each CONFER workshops 3 and 4, software deliverables), a progress report describing the technical work achieved during the second year, and appendices listing CONFER 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. Lone Leth and Bent Thomsen from ECRC greatly helped in the writing and the assembly of the document. Gérard Boudol from INRIA Sophia-Antipolis prepared the area report on Calculi, Rajagopal Nagarajan from Imperial College did the area report on Logics for Concurrency and the λ-calculus, Lone Leth and Bent Thomsen from ECRC wrote the area report on Programming Languages.

Chapter 2

Executive summary

The overall objective of the CONFER action is 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 organized around four main areas of work:

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

The objectives for the second year of CONFER put forth at the end of Year 1 of CONFER have been achieved at the end of Year 2. Significant results beyond these objectives have also been obtained. Furthermore, results obtained in the CONFER action are now having a considerable impact in many areas beyond the objectives of CONFER. Thus the second year of CONFER has been very successful.

In the area of Foundational Models and Abstract Machines, progress has been done in the theory of Actions Structures where action calculi have been defined. Also, various frameworks for dealing with bound variables have been precisely studied: combinatory systems, interaction systems, λ-calculus with explicit recursion. The work on explicit substitutions is pursued within the π-calculus or higher-order unification. Meta-theory has been also accomplished by considering abstract reduction systems and categorical models of term rewriting systems.

In the Calculi area the work has mainly dealt with the semantics of the π-calculus and other related formalisms. The work in this area has been subdivided into studies of the notion of bisimulation, non-interleaving semantics and relating calculi. All of the work in the area of Calculi is motivated by the desire of having a satisfactory semantic theory which is well suited for verification purposes.

Bisimulation is a powerful proof techniques which may be implemented in software verification tools. The understanding of bisimulation for higher order calculi has been advanced, however, for higher order calculi of concurrent processes the right notion of bisimulation is not easy to define.

The non-interleaving semantics was developed for CCS-like process algebras with the idea that a semantics reducing parallel composition to sequential non-determinism is not entirely appropriate for dealing with distributed systems. Both causal and spatial approaches are being considered. A remarkable result is that these approaches may be reduced to name passing and

interleaving in the π-calculus, thus demonstrating the fundamental nature of name passing. This enables the proof techniques for the π-calculus to be applied to non-interleaving semantics.

Several higher order process calculi have emerged, and relating these calculi have been an obvious needed task. Various sequential models of computations have also been compared.

In the area of Logics for Concurrency and the λ-calculus, work has proceeded on developing type systems for processes, based on principles derived from the work on Interaction Categories, to address issues in synchronous and asynchronous concurrent computation, verification of concurrent systems and mobility. These are non-trivial applications of the Interaction Category paradigm. There has been much work based on Linear Logic, Girard's **LU** and Categorical Logic as a basis for typed frameworks of processes. Game Semantics and Linear Logic have been used to advance the state-of-the-art in the theory of functional computation.

In the area of Programming Languages concrete results have started to emerge. There is now an advanced implementation of Pict, and some advanced demonstration programs have been developed.

The semantic foundation of important techniques on compile time optimisations via unfolding of programs has been studied. This work continues previous activities regarding termination properties of unfolding established last year. The latest results are expected to be mature enough to be applied in prototype compilers.

A programme of research – the Geometry of Implementations – aimed at developing efficient implementations of functional programs based on Girard's Linear Logic and the Geometry of Interaction semantics has been initiated. This work is derived from previous CONFER results of Gonthier et. al. on optimal reduction implementations and the work of Danos and Regnier on local and asynchronous β-reduction.

The Calumet demonstrator written in Facile presented at the first CONFER review has been developed into a robust application running on wide area networks. Calumet is now used frequently for presentations at ECRC.

The first release of Facile – the Facile Antigua Release – has been announced on the news net and made available for research and educational purposes. Technology transfer to the ECRC shareholders has also taken place [1].

The summaries in chapter 5 clearly reflect that the central idea of crossfertilization of ideas among fellow researchers in the consortium is very lively. 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. Many of these results are obtained in collaboration between sites, enabled through visits and longer stays. Some results are even produced in collaboration with leading researchers outside the consortium.

During Year 2 of CONFER two workshops have been held. The first was organised by Jan Willem Klop and Femke van Raamsdonk and took place in September 1993 at CWI. The workshop had 19 presentations and 35 participants. The first Year review took place on October 1st, 1993 at CWI. The second workshop in Year 2 was organised by Lone Leth and Bent Thomsen and took place in April 1994 at ECRC. The workshop had 21 presentations and 37 participants. A panel discussion on "What is mobility" was held at this workshop.

The action has produced a large number of reports and several of these have been published at conferences, international workshops and journals.

Several Ph.D.'s are in preparation in the action.

As in last years report, we would like to point out that the results achieved in CONFER so far are the result of the involvement of a large research community at each site, involving

[1] It should be noted that the Facile project was already in progress before the beginning of CONFER and the dimensions and scope of the project are wider than what can be considered as strictly relevant to the CONFER BRA.

not only the researchers strictly supported by the CONFER funding. Furthermore, a number of researchers outside the consortium are contributing to the effort. To this regard it is worth mentioning the impact that the results produced in CONFER on formalisms such as Linear Logic, π-calculus, CHOCS and CHAM are having on the research community not only in Europe. Furthermore, these results are being ued in other BRAs such as CLICS, ACCLAIM and LOMAPS. Beyond the research community in Europe, researchers in Japan and in the U.S. have started to use results form CONFER and to work on areas related to CONFER.

Objectives of the CONFER project for Year 3

This section contains a description of the objectives of the CONFER project for Year 3. It is a refinement of the addendum to the first periodic progress report of November 11th, 1993 which describes objectives of the CONFER project for Year 2 and Year 3.

Area 1: Foundational models and abstract machines

The group at Edinburgh (Cambridge) will more develop the new theory of Action Structures and Calculi, one of the possible goals being to capture both elementary interaction and calculi with bound variables. Progress in understanding of termination in corresponding frameworks (strong normalisation) is also possible. Finally, λ-reducers with sharing would have more official theory.

Area 2: Calculi

It is expected that the semantic foundation of Facile will be further investigated. In particular further studies of the constructs for distributed computing are expected. A collaboration with the CONFER group at University of Pisa has been initiated on this subject. More concretely this collaboration intends to apply the methodologies developed by G. Ferrari, U. Montanari and P. Quaglia in Year 1 of CONFER to the Facile language.

Area 3: Logics for Concurrency and λ-calculus

The collaboration between ECRC and SICS initiated last year on developing logics for reasoning about concurrent functional programs has been extended to include the group at LIENS. The objective of creating a full specification logic may be beyond what can be expected to be achieved in the CONFER action.

Area 4: Programming Languages

Future plans for the development of the Facile programming language include porting the software to other hardware platforms (such as Intel x86 processors).Related to this is work in progress on interoperability between Facile systems on different platforms. We currently use a system called the contract mechanism for connecting systems written in Facile with systems written in different languages. We expect to address questions such as interoperability with systems implementedin different programming languages (such as C, C++, prolog) via industrial standards (possibly the emerging CORBA standard). The question of interoperability may turn out to be a more involved research topic than what can be expected to be achieved in the CONFER action. It is expected that the Calumet system will be released to the scientific community and that technology transfer to the ECRC share holder companies will take place in the near future.

Chapter 3

Management

3.1 Consortium level

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

The following activities at the consortium level have taken place during the second year of the CONFER project:

Two workshops have been held. The first took place in September 1993 at CWI with 19 presentations and 35 participants. The second workshop took place in April 1994 at ECRC with 21 presentations and 37 participants. The first annual review to place on October 1st 1993 at CWI.

At the workshop at ECRC a management meeting was held to plan for the second annual review.

The fifth CONFER workshop will take place at the beginning of October at Imperial College, prior to the annual review on October 7th, 1994.

The ftp site at Imperial College has been operational for quite some time. All documents produced in CONFER can be found at this site (theory/CONFER at theory.doc.ic.ac.uk).

To disseminate information about the progress of CONFER the (technical) coordinators produced an overview of the first Year results. This report was published in Bulletin of EATCS, Number 52, February 1994, pp. 164-174.

3.2 CWI

3.2.1 Research directions

In the past year CWI has worked – and is continuing to work – in the following directions.

Higher-order rewriting

(E.g. in the format of Combinatory Reduction Systems or CRSs.) Roughly, this consists of term rewriting with bound variables. Various calculi featuring in the project have this characteristic: lambda calculi, pi-calculi. In the past year essential progress has been made: Van Raamsdonk proved in cooperation with Van Oostrom that also weakly orthogonal higher-order rewrite systems are confluent, thereby extending the well-known confluence for orthogonal systems. (A preliminary announcement of this result was already in last year's report.) The paradigm of a weakly orthogonal higher-order rewrite system is lambda calculus with beta and eta rule; also the typed versions such as system F with beta and eta. Also a useful "modularity" result was obtained: the disjoint union of left-linear confluent higher-order rewrite systems is again confluent. Actually, a refined version of that theorem states that confluence is also preserved in combinations of left-linear confluent higher-order rewrite systems when trivial critical pairs are allowed, generated by rules from different systems. (Inside each system separately there may be nontrivial critical pairs.)

Furthermore, Van Raamsdonk and Van Oostrom have worked out a general framework for higher-order rewriting (HORSs, Higher Order Rewriting Systems), where there is the parameter of the "substitution calculus" that is employed. For instance, Nipkow's Higher-order Rewrite Systems (HRSs) employ as substitution calculus the simply typed lambda calculus, while Klop's CRSs have underlined lambda calculus (as in "Finite Developments") as substitution calculus. The substitutions calculus is the underlying mechanism governing how actual rule instantiations are formed. Much of this work is reported in Van Oostrom's Ph.D. thesis, written under supervision of J.W. Klop at the Free University Amsterdam; this work formally is not in the project, but is influential for our work in the project.

Term Rewriting and Lambda Calculus with explicit recursion

The second major research direction of CWI continued the work initiated last year on term graph rewriting and cyclic lambda graph rewriting. This work is done in cooperation with Zena Ariola (University of Oregon). The origin of this work was in the former ESPRIT BRA Semagraph, now a WG. The addition which makes it interesting in CONFER, is the consideration of bound variables in the form of recursion variables: a term graph can be described as a system of recursive equations, in fact as a subclass of the well-known and much studied Recursive Program Schemes. The issue of bound variables becomes prominent when nested systems of such recursion systems are considered. Added to lambda calculus, we have in fact lambda calculus with "explicit recursion" - somewhat in analogy with the lambda calculi with "explicit substitution" or lambda sigma calculi that are also studied in our project. The objects of interest are "cyclic lambda graphs with a modular structure". Graphically, the modular structure (arising from nesting of recursion systems) consists of boxes, possibly nested, around parts of the graph; the inside of a box cannot be addressed directly, but only through its root. The β-rule gets in this framework an extremely simple form:

$$(\lambda\alpha.M)N \to \langle M \mid \alpha = N\rangle$$

where the role change in the binding of alpha is remarkable: first bound by lambda, later by the recursion construct. In functional programming these constructs are well-known as

the "let" and "letrec"; but as yet in our opinion the theory behind these constructs is not developed. It turns out for instance that lambda calculus with letrec, or our lambda calculus with explicit recursion, has a highly nontrivial confluence problem, consisting in first designing suitable rewrite rules and second in proving it confluent. In fact, a naive but "obvious" version of defining β-reduction on cyclic lambda terms turns out to be essentially non-confluent, unless one forbids the presence of certain cycles in the graph. Another way out is the introduction of boxes as a means of restricting all too liberal "copying" of parts of the graph. Interestingly, some of the rules are similar to the rules for lambda sigma calculus; and our calculus may be seen as a lambda calculus with explicit cyclic substitutions.

Process Algebra

This is work done by CWI's subcontractor, the University of Amsterdam (J.A. Bergstra and co-workers). In the report "Process Algebra with Combinators" a typed combinatory process algebra is introduced, that combines process algebra in the ACP framework with types and the classical combinators I, K, B, C. These serve to eliminate recursion variables altogether (thus having the opposite concern compared to (2) above), so that computations can be done in an entirely equational way. As an extended example the simple Alternating Bit Protocol is verified using only first-order equational logic. Another endeavour has been to eliminate recursion variables in process specifications in favour of iteration using variants of Kleene star. Axiomatisations have been given and different versions compared. Other work of a more practical signature was reported in the report The Toolbus - a component interconnection architecture. Basic frameworks here are Process Algebra, and the specification formalisms ASF and SDF.

3.2.2 Persons and exchanges

CWI participants are: Jan Willem Klop, Fer-Jan de Vries (until February 1, 1994), Femke van Raamsdonk. Part of the work is subcontracted to the University of Amsterdam: Jan Bergstra, Piet Rodenburg.

Cosimo Laneve, October 7, 1993, talk: About paths in lambda terms. Luke Ong, Univ. of Cambridge, February 22, 1994: The full abstraction problem for PCF.

3.2.3 Perspectives, work in progress

Work in progress.

1. Van Raamsdonk, in cooperation with Van Oostrom (presently at NTT Research Labs Tokyo): A new formalism of rewriting is introduced based on the analogy "rewriting = substitution + rules". The attempt is to find sufficient conditions on rewrite rules which allow to reduce optimality of rewriting to optimality of the logic embodied in the substitution mechanism. This work is related to and is strongly inspired by work of Gonthier, Abadi and Lévy and work (also in this project) of Asperti and Laneve.

2. Van Raamsdonk, in cooperation with Van Oostrom and Khasidashvili (University of East-Anglia). Rewriting with higher-order environments is introduced, where a context (possibly involving binders) can be replaced by another context. Furthermore, in the framework of Higher-Order Rewriting Systems (as in Van Oostrom's thesis and in the paper "Weak orthogonality implies confluence: the higher-order case", by van Oostrom & van Raamsdonk) we aim to study rewriting on terms containing some version of "explicit substitutions". We aim to show that these two approaches are equivalent and that they allow to simulate an implementation of optimal reduction for the case of untyped lambda-calculus.

3. J.W. Klop, in cooperation with Zena Ariola (Oregon). Continuation of development of confluent calculi for cyclic graphs, or phrased differently, explicit recursion; both as extension of the first-order case of rewriting and the lambda calculus. In cooperation with M.R. Sleep, J.R. Kennaway (both at the University of East-Anglia) and F.J. de Vries (until February 1994 involved in our project and presently at NTT Research Lab, Kyoto) we are developing the theory of infinite lambda calculus where terms are possibly infinite trees and rewrite sequences may have infinite length. This calculus is partly well-known as it contains the "Boehm-trees". Understanding of the infinite lambda calculus turned out to be important as an (operational) semantics for cyclic lambda graphs; after unwinding such a graph we are in the infinite lambda calculus.

3.2.4 Publications, technical reports

- J.A. Bergstra, P. Klint. *The Toolbus: a Component Interconnection Architecture.* Technical Report P9408, March 1994, Programming Research Group, University of Amsterdam
- V. van Oostrom. *Confluence for Abstract and Higher-order Rewriting*, Ph.D. thesis, March 1994, Free University of Amsterdam.
- J.W. Klop, V. van Oostrom, F. van Raamsdonk. *Combinatory Reduction Systems: introduction and survey.* TCS, Vol.121, Nrs.1-2, Dec. 1993, guest eds. M. Dezani-Ciancaglini, S. Ronchi Della Rocca, M. Venturini-Zilli, A Collection of Contributions in Honour of Corrado Boehm on the Occasion of his 70th Birthday, p.279-308.
- J.A. Bergstra, I. Bethke, A. Ponse. *Process Algebra with Combinators.* In: Proc. 7th Workshop CSL '93 (Computer Science Logic), Swansea 1993, eds.: E. Borger, Y. Gurevich, K. Meinke, Springer LNCS 832, pp.36-65.
- J.A. Bergstra, I. Bethke, A. Ponse. *Process algebra with iteration and nesting.* Technical Report P9314b, Programming Research Group, University of Amsterdam, 1994.
- Z. Ariola, J.W. Klop. *Cyclic Lambda Graph Rewriting.* In: Proc. 9th Annual IEEE Symposium on Logic in Computer Science (LICS '94), Paris, July 1993, pp. 416-425.
- V. van Oostrom & F. van Raamsdonk. *Comparing Combinatory Reduction Systems and Higher-order Rewrite Systems.* In: J. Heering, K. Meinke, B. Möller, T. Nipkow (Eds.), Higher-Order Algebra, Logic and Term Rewriting (HOA '93), Lecture Notes in Computer Science, Vol. 816, pp. 276-304.
- V. van Oostrom & F. van Raamsdonk. *Weak Orthogonality Implies Confluence: The Higher-Order Case.* In: A. Nerode, Yu.V. Matiyasevich (Eds.), Logical Foundations of Computer Science (LFCS '94), Lecture Notes in Computer Science, Vol. 813. pp. 379-392

3.3 University of Edinburgh

3.3.1 Research directions

The major areas of research of the Edinburgh group have been π-calculus and action structures.

There has been strong progress in action structures. The main contribution has been the definition of *action calculi*, which are concrete action structures representing many familiar calculi (Petri nets, π-calculus, λ-calculus) in a single framework. The foundations for this work were laid in the preceding year, but many publications and new technical developments occurred within the review period. Significant work has also been done in defining and analysing the category of *control structures*, which is intended to accommodate the model theory of action structures. An alternative presentation of action calculi in which names are less prominent, and the connection between higher order action calculi and the lambda calculus have been investigated.

Research has continued on developing the semantics of π-calculus and the programming language Pict, which is based on the π-calculus. Various *true-concurrent* behavioural equivalences for π-calculus have been examined and carefully compared with the ordinary *interleaving* equivalences. A study of open bisimulation and barbed bisimulation in π-calculus has been made. Other works focusing on the notion of bisimulation include: a study of proof techniques for bisimulation, aimed at relieving the effort needed to prove bisimilarity results; a study of the meaning of bisimulation in higher-order calculi. Work also continued on applications of π-calculus theory to parallel object-oriented programming languages.

The implementation of the Pict language has been substantially refined, and novel aspects of the type system (higher-order polymorphism, extensible records, type inference) have been examined. The compiler is now fast enough to begin experimenting with medium-scale examples: a concurrent graphics toolkit and a demonstration minesweeper game that runs to over three thousand lines of Pict are now available.

3.3.2 Perspectives, work in progress

Action structures. We anticipate progress mainly in the topic of control structures. Here, two important topics come into focus: (1) the generalised treatment of *bisimilarity* in action calculi, as a special kind of morphism of control structures; (2) the investigation the *structure of names* – which have hitherto been treated as an unstructured set.

π-calculus. We have recently isolated a calculus, called πI, which lies in between π-calculus and CCS. πI appears to have an expressiveness comparable to that of π-calculus, but yet an algebraic theory very close to that of CCS. The main motivation for this study is to understand the reasons for the gap between CCS and π-calculus, in terms of expressiveness and algebraic theory. We have also begun to explore the theory of confluence in π-calculus and to apply it to problems in parallel object-oriented programming.

As for the language Pict, our main focus now is on cleaning up and documenting the implementation for a first public release. We plan to write two or more retrospective papers on our experience with the language design and implementation.

3.3.3 Person and exchanges

The Edinburgh group involved in the project comprises Robin Milner, Davide Sangiorgi, Benjamin Pierce, David Turner, Peter Sewell, Alex Mifsud, Philippa Gardner, John Power and David Walker. Davide Sangiorgi is full-time employed within the project.

David Turner, Peter Sewell and Alex Mifsud are PhD students, under the supervision of Robin Milner whose theses are relevant to the CONFER project and are under completion in this period.

Benjamin Pierce has visited twice INRIA-Rocquencourt, to discuss issues related to the Pict implementation. Davide Sangiorgi has visited CWI in the period 15 October – 15 November 1993, Pisa in June 1994, and INRIA-Sophia Antipolis in May 1994. These visits have had a strong impact on the works [8] and [9] in section 3.3.5 below. David Walker has visited Pisa in July 1994 to give a seminar and discuss research with CONFER partners.

3.3.4 Publications

1. Robin Milner, *An action structure for synchronous π-calculus*, Proc. FCT Conference, Szeged, Hungary, LNCS, Vol 710, August 1993, 87–105.

2. Robin Milner, *Action calculi, or concrete action structures*, Proc. MFCS Conference, Gdansk, Poland, LNCS, Vol 711, September 1993, 105–121.

3. Robin Milner, *Higher-order action calculi*, to appear in Proc. CSL conference, Swansea, October 1993.

4. *Pi-nets: a graphical form of pi-calculus*, Proc. ESOP'94, LNCS Vol788, Springer-Verlag, April 1994, 26–42.

5. *Bisimulation is not finitely (first-order) equationally axiomatisable*, in Proc. LICS '94.

6. Martin Steffen and Benjamin Pierce, *Higher-Order Subtyping*, Proc. IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET), S. Miniato, Italy, June 94, to appear.

7. Davide Sangiorgi, *Bisimulation in higher-order calculi*, Proc. IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET), S. Miniato, Italy, June 94, to appear.

8. Davide Sangiorgi, *Locality and Non-interleaving Semantics in Calculi for Mobile Processes*, in Proc. International Symposium on Theoretical Aspects of Computer Science (TACS '94), LNCS 789, Springer Verlag.

9. David Walker, *Algebraic proofs of properties of objects*, Proc. 5th European Symposium on Programming, Edinburgh, April 1994, D. Sannella (ed.), Springer-Verlag LNCS vol. 788, 501-516.

10. David Walker, *On bisimulation in the pi-calculus*, to appear in Proc. 5th International Conference on Concurrency Theory, Uppsala, August 1994, Springer-Verlag LNCS.

3.3.5 Unpublished research reports

1. Robin Milner, *Action calculi IV: molecular forms*, November 1993.

2. Robin Milner, *Action calculi V: reflexive molecular forms*, June 1994. (with Appendix by Ole Jensen)

3. Robin Milner, *Action calculi VI: strong normalisation at higher-order*, December 1993.

4. Robin Milner, Alex Mifsud and John Power *Control Structures*, June 1994.

5. Philippa Gardner, *Closed Action Calculi*, July 1994.

6. Benjamin Pierce, Didier Rémy and David N. Turner, *A Typed Higher-Order Programming Language Based on the Pi-Calculus*, 1994.

7. Benjamin Pierce, *Programming in the Pi-Calculus: An Experiment in Programming Language Design*, February 1994.

8. Michele Boreale and Davide Sangiorgi, *A study of causality in the π-calculus*, Submitted for publication.

9. Davide Sangiorgi, *On the bisimulation proof method*, July 1994.

3.3.6 Software

1. Benjamin Pierce, Didier Rémy and David Turner, *Pict: A typed, higher-order concurrent programming language based on the π-calculus*, 1994.

3.4 ECRC

The major research direction of the group at ECRC is the Facile programming language, its semantics, implementation and its application in developing significant demonstrators of industrial relevance.

In the 2nd Year of CONFER the group at ECRC has completed a first release of the Facile programming language – the Facile Antigua Release. The Facile Antigua Release is an industrial strength implementation of a distributed higher order concurrent functional programming language based on the theoretical models (such as the λ-calculus, CCS, the π-calculus and CHOCS) related to the CONFER action. The Facile Antigua Release is implemented by modifying and extending the Standard ML of New Jersey implementation. The software is documented through a user guide (ECRC/M2/R1), as well as numerous papers on its semantics and abstract implementations.

Since January 1994 the Facile Antigua Release has been in alpha test at sites in Europe and the US. Feedback from the alpha test sites has led to minor debugging and has resulted in a rather stable implementation.

Technology transfer to the ECRC share holder companies ICL, Bull and Siemens took place in July 1994. The software is currently under evaluation in several departments.

In July 1994 the availability of Facile Antigua Release has been publicly announced to the research community via the news network and announcements to all ESPRIT partners via email.

The Calumet cooperative application for teleconferencing, demonstrated at the 1st Year review at CWI, has been restructured and rendered more fault tolerant. A new user interface programmed in C++ has been developed. The previous user interface was programmed in the TUBE system which requires a LeLisp system. The LeLisp system comes under a license, which has proved to be problematic for dissemination since not many sites have a LeLisp license. The new user interface can display LaTeX style slides with graphics in TIFF format. The Calumet system is now used frequently at ECRC for internal presentations and it has been demonstrated on wide area networks at AT&T (Murray Hills, New-Jersey) and between ECRC, CWI (Amsterdam) and CMU (Pittsburgh). Jean-Pierre Talpin has produced two reports (ECRC/M2/R4, ECRC/M2/R12) describing the aspects of the Calumet system and its implementation. The report (ECRC/M2/R12) is considered a joint CONFER/CONCUR2 deliverable. Jean-Pierre Talpin, Philippe Marchal and Klaus H. Ahlers have produced a user manual for the Calumet system (ECRC/M2/R13).

Several theoretical developments have been achieved during the 2nd Year of CONFER. Sanjiva Prasad has been studying a core language of communicating applicative processes from a formulae-as-types, programs-as-proofs perspective. The idea is to view communicating applicative processes as concurrent interacting proof procedures, with interprocess communication viewed as a form of proof normalisation. The logical framework proposed is a conservative extension of an intuitionistic logic with features to handle concurrency and communication. It is presented as a "quasi-intuitionistic" sequent calculus within the framework of Girard's *unified logic* **LU**. Two technical reports (ECRC/M2/R5, ECRC/M2/R6) concern the consequences, especially on the cut elimination result for **LU**, of adding to **LU** (1) a rule (Mingle), similar to "Mix" of linear logic, and (2) a new encoding of implication — roughly as !(!A -o !B). A term assignment relates communicating applicative programs to deductions in that quasi-intuitionistic fragment of **LU**. The novelty lies in relating communication primitives to the "classical" cut rules of **LU**. The notion of permuting inference rules, with the help of structural rules, is used to show that every sequent deducible in the fragment has a term assignment. The operational semantics (both labelled and unlabelled reduction) is then shown to correspond to eliminating cuts under a particular strategy. The program transformations induced by the permutation of

rules also suggest a connection with concepts such as "structural equivalences", "ionisation" and "magical mixing" used in Chemical Abstract Machines and traditional (hitherto untyped) process calculi (ECRC/M2/R11).

While visiting ECRC Roberto Amadio has studied various theoretical aspects of Facile compilations, such as CPS transformations and transformation of synchronous to asynchronous communication (ECRC/M2/R3, this report is also considered a deliverable for ENS). At ECRC Roberto Amadio finalised some work on various notions of bisimulation for the π-calculus. This work is reported in ECRC/M2/R2, also considered a deliverable for ENS. Roberto Amadio, Lone Leth and Bent Thomsen have studied the connection between the concurrent functional core of Facile and the π-calculus. This has been done through a series of transformations which are shown to be adequate. The notion of barbed bisimulation developed by Davide Sangiorgi and Robin Milner in the 1st Year of CONFER has been used to place the concurrent functional core of Facile and the π-calculus on an equal semantic footing (ECRC/M2/R8, this report is also considered a deliverable for ENS).

Roberto Amadio and Sanjiva Prasad have studied some aspects of the distributed computing model of Facile, *e.g.*, locality of processes and channel allocation, and the effect of location failure on program behaviour. They have presented an extension of the π-calculus with located channels and actions, with location names as first-class data, and where locations may fail. Locality is not observed directly and becomes apparent only on location failure as the disappearance of certain interaction capabilities (this distinguishes the approach from previous work, notably by the Sophia group, where distribution is directly observed). The behaviour of programs is formalised using the notion of barbed bisimulation, developed by Milner and Sangiorgi (following Pnueli). The calculus is related to a simply-sorted π-calculus by a translation where a protocol is used to represent the location dependencies of actions and channels. The translation is shown to be adequate with respect to the barbed semantics (ECRC/M2/R10, this report is also considered a deliverable for ENS).

The work by Lone Leth and Bent Thomsen on describing aspects of the Facile implementation using the CHAM framework has been extended to cover aspects of physical distribution. Furthermore, more faithful models of channel management have been developed. These developments have been described in the paper: "Facile Chemistry Revised" (ECRC/M2/R9). This paper is a revised version of the paper "Some Facile Chemistry" (ECRC/M1/R1).

3.4.1 Project level administration

At the administrative level ECRC has involvement in project management at the consortium level assisting Jean-Jacques Lévy in the technical coordination of the action. A visible result of this activity is the EATCS publication of the 1st Year report on the CONFER project (ECRC/M2/R7).

The 4th CONFER workshop was arranged by Lone Leth and Bent Thomsen and held at ECRC (April 18th-20th, 1994). The workshop had 37 participants with 21 presentations.. A discussion session focusing on the question of: "What is mobility" took place.

3.4.2 Persons and exchanges

The following ECRC personnel has been engaged in the action in Year 2 of CONFER: Alessandro Giacalone, Andre Kramer, Tsung-Min Kuo, Lone Leth, Sanjiva Prasad, Jean-Pierre Talpin and Bent Thomsen.

Fritz Knabe (Ph. D. student), Klaus H. Ahlers, Pierre-Yves Chevalier (visitor), Philippe Marchal and Chris Crampton have also contributed to the development of Facile and the Calumet system.

Alessandro Giacalone and Sanjiva Prasad have left the project on 1st of September, 1994.

Members of the group have taken part in the two CONFER workshops (no. 3 and 4) held in Year 2. Andre Kramer, Alessandro Giacalone, Jean-Pierre Talpin, Lone Leth and Bent Thomsen took part in the 3rd workshop. Andre Kramer, Jean-Pierre Talpin, Lone Leth and Bent Thomsen gave demonstrations of the Facile system during the workshop. Bent Thomsen gave a talk on "Towards compositional reasoning about Facile programs" and Andre Kramer gave a talk on "Distributing Facile".

Jean-Pierre Talpin, Lone Leth and Bent Thomsen took part in the 1st Year review. Bent Thomsen gave the area report on programming languages. Jean-Pierre Talpin and Andre Kramer (sitting at ECRC) demonstrated the Calumet system at the 1st Year review.

Pierre-Yves Chevalier, Alessandro Giacalone, Andre Kramer, Fritz Knabe, Tsung-Min Kuo, Lone Leth, Jean-Pierre Talpin and Bent Thomsen took part in the 4th workshop. Bent Thomsen gave a talk on "An overview of Facile Antigua Release". Jean-Pierre Talpin and Andre Kramer demonstrated the Facile Antigua Release.

26th of June to 28th of June, 1994, Lone Leth and Bent Thomsen visited the CONFER group at the university of Pisa.

20th of July to 22nd of July, 1994, Sanjiva Prasad visited the CONFER group at CNRS and INRIA-Lorraine, Nancy, France.

Roberto Amadio from CNRS and INRIA-Lorraine, Nancy, France, was on an extended visit to ECRC for six months from 6th of September, 1993, to February, 1994. He worked together with Sanjiva Prasad, Lone Leth and Bent Thomsen on the formal foundation of Facile.

A two day meeting with Prof. U. Montanari, Dr. G. Ferrari, F. Gadducci and P. Quaglia took place at ECRC (April 14th-15th).

3.4.3 Perspectives, work in progress

Research has progressed in all aspects of the groups involvement with CONFER.

Due to the changes in personnel and ECRC's involvement in the LOMAPS 8130 action we have decided to move further work on *Sorts & Types* to LOMAPS.

On *Abstract Machines, Primitive Constructs* we consider our investigation using the CHAM framework completed. If time permits we may investigate the co-existence of multiple communication paradigms. The work on applications has pointed out that it will be beneficial to look at other communication paradigms such as broadcasting and asynchronous point-to-point. Currently these may be implemented on top of hand shake communication, as is the case for the broadcast library used in the Calumet application. However, for efficiency reasons it may be interesting to consider broadcasting a primitive and have it co-exist with hand shake communication.

On *Dynamic Behaviour* it is expected that the semantic foundation of Facile will be further investigated. In particular further studies of the constructs for distributed computing are expected. A collaboration with the CONFER group at University of Pisa has been initiated on this subject. More concretely this collaboration intends to apply the methodologies developed by G. Ferrari, U. Montanari and P. Quaglia in Year 1 of CONFER to the Facile language.

The collaboration with SICS initiated last year on developing logics for reasoning about concurrent functional programs has been extended to include the group at LIENS.

On *Programming Languages* we have plans for the further development of the Facile programming language to include porting the software to other hardware platforms (such as Intel x86 processors). Related to this is work in progress on interoperability between Facile systems on different platforms. We currently use a system called the contract mechanism developed by Chris Crampton for connecting systems written in Facile with systems written in different languages. We expect to address questions such as interoperability with systems implemented

in different programming languages (such as C, C++, prolog) via industrial standards (possibly the emerging CORBA standard). This may replace the contract mechanism. Furthermore, we hope to address issues related to mobile computing, in particular running Facile on portable devices and interacting with Facile systems running on stationary networks.

Related to the question of running Facile on portable devices a new direction for applications has just been initiated. The model of higher order mobile processes supported by Facile seems a good candidate for structuring software which has to operate in a mobile environment. We are currently conducting experiments to verify this hypothesis.

It is expected that the Calumet system will be released to the scientific community and that technology transfer to the ECRC share holder companies will take place in the near future.

3.4.4 Publications, technical reports

A set of 13 reports has been produced as deliverables during Year 2 of CONFER:

ECRC/M2/R1: Thomsen, B., Leth, L., Prasad, S., Kuo, T.-S., Kramer, A., Knabe, F., Giacalone, A.: "Facile Antigua Release – Programming Guide", Technical report ECRC-93-20, 1993.

ECRC/M2/R2: Roberto M. Amadio and Otmane Ait-Mohamed. "An Analysis of π-calculus Bisimulations", Technical report ECRC-94-2, 1994.

ECRC/M2/R3: Roberto M. Amadio. "Translating Core Facile", Technical report ECRC-94-3, 1994.

ECRC/M2/R4: Jean-Pierre Talpin. "The Calumet Experiment - Part I: An Implementation of Group-Communication Protocols in Facile", Technical report ECRC-94-4, 1994.

ECRC/M2/R5: Sanjiva Prasad. "Cut Elimination for LU with Mingle", Technical report ECRC-94-10, 1994.

ECRC/M2/R6: Sanjiva Prasad. "The Positive Intuitionistic Fragment of LU", Technical report ECRC-94-11, 1994.

ECRC/M2/R7: Levy, J.-J., Thomsen, B., Leth, L., Giacalone, A.: "First Year Report for Esprit Basic Research Action 6454-CONFER – CONcurrency and Functions: Evaluation and Reduction", Bulletin of EATCS, Number 52, 1994, pp. 164-174.

ECRC/M2/R8: Roberto M. Amadio, Lone Leth and Bent Thomsen: "From Concurrent Functional Programs to Mobile Processes", submitted for publication.

ECRC/M2/R9: Leth, L., Thomsen, B.: "Facile Chemistry Revised", Technical report ECRC-94-36, 1994.

ECRC/M2/R10: Roberto M. Amadio and Sanjiva Prasad: "Localities and Failures", Technical report ECRC-94-18, 1994.
This paper has been accepted for FST&TCS'14 to be held in Madras, India, December 1994.

ECRC/M2/R11: Sanjiva Prasad: "Towards a Formulae-as-Types View of Communicating Applicative Programs (Extended Summary)", Technical report ECRC-94-32, 1994.

ECRC/M2/R12: Jean-Pierre Talpin: "The Calumet Experiment in Facile - A Model for Group Communication and Interaction Control in Cooperative Applications", Technical report ECRC-94-26, 1994.

ECRC/M2/R13: Talpin, J.-P., Marchal, P., and Ahlers, K.: "Calumet - A Reference Manual", Technical report ECRC-94-30, 1994.

The papers ECRC/M2/R1, R4, R5, R6, R9, R10, R11, R12 and R13 have been placed at the CONFER ftp site at Imperial College.

3.4.5 Software

The main thrust of work at ECRC during Year 2 of CONFER has been on completing the software systems:

Facile The group at ECRC has completed a first release of the Facile programming language – the Facile Antigua Release. Technology transfer to the ECRC share holder companies, ICL, Bull and Siemens took place in July 1994. The software is currently under evaluation in several departments, and exploitation plans are expected to be developed in the near future. In July 1994 the availability of Facile Antigua Release has been publicly announced to the research community via the news network and announcements to all ESPRIT partners via email.

Calumet The Calumet system has been restructured and rendered more fault tolerant. It is expected that the Calumet system will be released to the scientific community and that technology transfer to the ECRC share holder companies will take place in the near future.

We would like to point out that the above pieces of software are clearly not the results of the work delivered by just 1.5 FTE ESPRIT funding. It is the result of the involvement of the whole group at ECRC working on Facile and Calumet. It is our policy to make the results of the entire group available whenever possible.

3.5 ENS

3.5.1 Outline

The group at LIENS together with the associated researchers from Marseille, Nancy and Paris has produced significative contributions in the following research areas:

- Concurrent and Distributed Process Calculi.
- Optimal Reduction in the λ-calculus.

The group carries on mainly theoretical research in all areas of the CONFER project. This year has been marked by an intense exchange with other CONFER sites which has resulted in a number of joint papers. In particular, the work on calculi has been carried on in strict collaboration with ECRC, INRIA-Sophia, and SICS. R. Amadio has visited ECRC during six months (September 93-February 94) and SICS during one month (April 94). R. Amadio and C. Lavatelli have had frequent exchanges with G. Boudol from INRIA-Sophia (G. Boudol is directing Lavatelli's thesis and he has served as a referee for Amadio's *habilitation*). The work on optimal reduction has been carried on by V. Danos and L. Regnier in strict collaboration with A. Asperti (U. Bologna, previously INRIA-Rocquencourt) and C. Laneve (INRIA-Sophia).

3.5.2 Persons

The following researchers are engaged in the action: Roberto Amadio (CNRS-INRIA, Nancy), Pierre-Louis Curien (CNRS, LIENS), Vincent Danos (CNRS, Paris VII), Carolina Lavatelli (PhD Student, LIENS Paris), and Laurent Regnier (CNRS, Marseille). This year P.-L. Curien has been on sabbatical at Beijing University, during this period R. Amadio and C. Lavatelli have assured the site management.

3.5.3 Reports and Publications

A set of four reports has been produced as deliverables during Year 2 of CONFER:

1. R. Amadio and O. Ait-Mohamed. *An analysis of π-calculus bisimulations.* Technical Report 94-2, European Computer-Industry Research Center, 1994.
2. R. Amadio. *Translating core Facile.* Technical Report 94-3, European Computer-Industry Research Center, 1994.
3. R. Amadio and S. Prasad. *Localities and failures.* Technical Report 94-18, European Computer-Industry Research Center, 1994.
4. A. Asperti, V. Danos, C. Laneve and L. Regnier, *Paths in lambda-calculus.* Proceedings IEEE-LICS 94, Paris.

Deliverables 1, 2, and 3 relate to the *Calculi* area. Deliverable 4 relates to the *Foundational models and abstract machines* area.

3.5.4 Description of Technical Contributions and Related Work

1. R. Amadio, in collaboration with L. Leth, S. Prasad, and B. Thomsen from ECRC, has investigated the concurrent and distributed semantics of the Facile programming language and its relatives: π-calculus, Chocs,...

   In particular, the (dynamic) semantics of a "core Facile" language has been studied. This should be taken as a basis for the definition of abstract machines, the transformation of programs, and the development of modal specification languages. Roughly, the "core Facile" language is a call-by-value simply typed λ-calculus enriched with *parallel composition*, *dynamic channel generation*, and *input-output synchronous communication primitives*. We remark two main contributions. First, a new semantics based on the notion of *barbed bisimulation* is introduced. It is argued that the derived equivalence provides a more satisfying treatment of restriction, in particular by proving the adequacy of a natural translation of Facile into π-calculus it is suggested that this approach is in good harmony with previous research on the semantics of sub-calculi of Core Facile such as *Chocs* and π-calculus. Second, various aspects of Facile compilation are analysed at an abstract level. In particular, an "asynchronous" version of the Facile language is introduced, say F_a, where asynchrony here means that a process terminates after having performed an output action. It is shown that there is an adequate CPS (Continuation Passing Style) translation from core Facile to F_a. Moreover, an abstract machine which executes code of F_a programs is described. This machine is basically a multiset of environment machines for eager functional evaluation enriched with a mechanism for the dynamic generation of channel names.

   A second work, in collaboration with S. Prasad, concerns the formalisation and analysis of the notion of locality and failure in the Facile distributed programming language. Towards this end, a simple extension of the π-calculus with located actions and channels and with location names as first-class data is presented The interaction between localities and failures distinguishes our approach from previous ones (notably from the Sophia group) where the notion of locality is considered in isolation. It is argued that the combination of these two features leads, at least from the distributed programming viewpoint, to a more natural semantics. A translation of this calculus into a standard simply-sorted π-calculus is then presented and shown to be adequate with respect to a barbed bisimulation based semantics. In the translation each location is represented by a special process which interacts, by means of a simple protocol, with any process of the original program that wants to access resources depending on that location.

   Finally, a revised version of Amadio's work on various notions of bisimulation for the π-calculus has been made available. Of particular interest appears to be the introduction of a refinement of the notion of "late" bisimulation called "uniform". Roughly, the latter corresponds to the idea of treating the formal parameter of an input prefix as a "logical" variable. This notion, and the related notion of open bisimulation independently proposed by D. Sangiorgi, have been recently given a unified treatment in the work developed by the Pisa site.

2. V. Danos and L. Regnier in collaboration with C. Laneve and A. Asperti have shown the equivalence of different notions of paths in the λ-calculus. This is the outcome of a series of discussions between these researchers during the Sophia (January 93), Edinburgh (May 93) and Amsterdam (September 93) CONFER meetings.

   This work unifies several notions arising in the theory of β-reduction, namely labelled reductions, redex families, sharing reductions, geometry of interaction, and virtual reduc-

tions. All these notions relate to the problem of defining a "local" implementation of β-reduction. Eventually, we expect that these ideas will lead to novel techniques for the parallel execution of functional languages.

In the following we survey the three different notions of paths which have been shown to coincide. Lévy took hold in 1978 of the difficult notion of two redexes being created in the "same" way during a reduction (in which case they were said to belong to the same "family"). Then he labelled terms and made β-reductions act on labels so that two redexes were in the same family iff they had the same labels. Fifteen years later, labels were again considered by Asperti and Laneve where they were identified with *legal paths*. More precisely: 1) labels of redexes in any reduct N of M denote paths in M; 2) those paths are legal; 3) conversely, any legal path in M denotes a label of a redex to appear somewhere in the set of reducts of M. Legality is a simple and effective condition that intuitively asks for enough symmetry in the path so that the reduction may unfold it into a redex.

In the meantime, people were seeking for a shared reduction faithfully implementing the notion of families, i.e., a reduction where families could be said to be reduced in one step. Such a reduction was discovered by Lamping and Kathail in 1990 (subsequent simplifications were given by Gonthier et. al. and Asperti). The invariants used to prove the correctness of Lamping's implementation were *consistent paths*.

Finally Girard unveiled in 1988 an interpretation of the cut-elimination procedure for linear logic. Again the alternative computation could be defined as the computation of a particular set of paths on proofs, namely *regular paths* which were defined through an algebraic and computational device the *dynamic algebra* (this was extended to the λ-calculus in Danos and in Regnier PhD thesis).

We conclude by remarking that similar results could be formulated in the framework of proof-nets: a syntax for linear logic where paths are easily defined and manipulated. This point is important since, as shown in recent work by Danos et. al., other calculi can be embedded in linear logic, e.g. the $\lambda\mu$-calculus of Parigot.

3. C. Lavatelli has proposed, in collaboration with G. Boudol (INRIA-Sophia), a domain theoretic interpretation of the λ^r-calculus. This is a lambda calculus with resources, proposed by G. Boudol, which is non-deterministic and allows explicit control of arguments at the syntactic level. It has been shown (by Boudol and his collaborators) that this calculus allows to clarify the relationships between the call-by-name λ-calculus and its π-calculus interpretation.

   The proposed domain theoretic interpretation relies on the following domain equation involving a powerdomain construction which is needed for the interpretation of resources.

$$D = (\mathcal{M}(D) \to D)_\perp$$

   The points of $\mathcal{M}(D)$ are roughly multisets of points of D, and solutions of this equation are meant to be found among the class of prime algebraic complete lattices. A filter domain is defined which solves the equation; moreover, the related interpretation of terms is shown to coincide with the set of types that can be assigned to the term in a type system with intersection types à la Coppo. A previous result by Boudol shows that the "compact version" of this system is adequate w.r.t. the operational semantics of λ^r. A report on this work will appear soon.

3.5.5 Perspectives

R. Amadio has started a collaboration with M. Dam (SICS) on the specification and verification of higher-order process calculi. We have proposed a specification language à la Hennessy-Milner that is sufficiently expressive to capture a suitable notion of higher-order bisimulation and we are currently investigating the possibility of developing sound and complete "compositional" proof systems. Another (related) problem that he plans to study in collaboration with S. Prasad (ECRC) concerns the investigation of a programs-as-proofs view of certain fragments of higher order concurrent languages.

V. Danos and L. Regnier will pursue their work on optimal reduction. A new interesting direction concerns the connections between the structures developed in Girard's geometry of interaction and certain game-theoretic structures that have been recently developed to provide fully abstract models of PCF (notably in the work of Abramsky, Jagadeesan, and Malacaria at Imperial College, and of Hyland and Ong at Cambridge).

Finally, C. Lavatelli plans to investigate the full abstraction of the filter model interpretation of the λ^r-calculus described above.

3.6 Imperial College, London

Research directions

Research done at Imperial College has covered three out of four areas of the CONFER project. However, most of it has been concentrated on Logics for Concurrency and λ-calculus.

The main thrust has been on developing type systems for concurrency. By virtue of the Curry-Howard isomorphism, there is a logic associated with these type theories and this allows reasoning about typed terms. This approach provides a sound logical basis for our type systems. There has been work on a wide variety of type systems for different concurrent paradigms and for verification of concurrent programs. Most of these typed frameworks are based on Interaction Categories. The work on Internal Language for Interaction Categories tightens the correspondence between proofs and processes in the same spirit as the familiar correspondence between λ-calculus, Intuitionistic Logic, and Cartesian Closed Categories. Some work has also been done on generalising process constructs so that Interaction Categories arise as particular instances of this generalisation.

In the λ-calculus or functional setting, here has been some important results extending previous work on proof nets for Intuitionistic Linear Logic to include exponentials. The work on Interaction Orders as Games can also be considered as a useful piece of research in this area, linking algebraic lattices and games. Work is nearing completion in the area of the Geometry of Interaction Machine which forms a basis for correct, efficient implementations of functional programs.

Personnel and exchanges

Personnel involved in CONFER at this site are Samson Abramsky, Simon Gay, François Lamarche, Ian Mackie, Pasquale Malacaria, Greg Meredith, Rajagopal Nagarajan and Duško Pavlović. Michael Huth, who has recently left the college, participated in one of the workshops. Samson Abramsky, Pasquale Malacaria, François Lamarche, and Duško Pavlović have participated in other projects as well — for the purposes of deliverables, we have made sure that we separate out the work as much as possible.

Two PhDs are nearing completion — that of Simon Gay and Ian Mackie, both under the supervision of Samson Abramsky.

Respective titles:

- Linear Types for Communicating Processes
- The Geometry of Implementation

Both these topics are closely connected to the aims of the CONFER project. Within the last year Greg Meredith has started his PhD work under Abramsky's supervision and his work is also appropriate to the project.

Ian Mackie has made visits to Paris on several occasions and has had useful discussions with Vincent Danos (CNRS, Paris VII) and Laurent Regnier (CNRS, Marseille) both of whom are attached to the ENS site. Pasquale Malacaria has arrived at Imperial from Ecole Normale Supérieure during the summer of last year.

We have had several discussions with researchers at the University of Edinburgh regarding possible connections between the work on Interaction Categories and that on Action Structures.

Sanjiva Prasad from ECRC, Munich visited us on one occasion and we discussed possible relationships between his work and ours.

Our site members have actively participated in the two CONFER workshops held in the last year, giving several presentations, and have had useful exchange of ideas with other CONFER participants.

Publications

The results obtained by the group have given rise to the following publications (note that here we only list papers which are complete or for which acceptance notice has already arrived):

[1] S. Abramsky, S. J. Gay, and R. Nagarajan. Interaction Categories and Typed Concurrent Programming. *Deductive Program Design: Proceedings of the Marktoberdorf International Summer School*, 1994. NATO ASI Series F: Computer and Systems Sciences, Springer Verlag. To appear.

[2] R. L. Crole, S. J. Gay, and R. Nagarajan. An Internal Language for Interaction Categories. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

[3] M. Huth. Interaction Orders as Games. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

[4] F. Lamarche Hypercoherences are Chu Spaces. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

[5] F. Lamarche. Proof Nets for Intuitionistic Linear Logic I: Essential nets. Report available by anonymous ftp from `theory.doc.ic.ac.uk`.

[6] F. Lamarche. Dialectics: a model of linear logic and PCF. Report available by anonymous ftp from `theory.doc.ic.ac.uk`.

[7] I. C. Mackie. The Geometry of Interaction Machine. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

[8] P. Malacaria. Studying equivalences of transition systems with algebraic tools. *Theoretical Computer Science* To appear.

[9] D. Pavlović Categorical logic of concurrency and interaction, I: Synchronous processes. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

3.7 INRIA-Rocquencourt

3.7.1 Research directions

INRIA Rocquencourt has worked in the following directions.

- Optimal reductions. This work started 2 years ago. It is continued with journal version of two 1992-papers, which are very tedious to write. Asperti (with Laneve) pursued their work on extending this work to interaction systems.
- Abstract reductions systems. It is a difficult game where axiomatisation of reductions in the λ-calculus or in Klop's combinatory systems, by only considering nesting of redexes and their relation with binders. For the standardisation theorem, finite developments have been treated thoroughly.
- Explicit substitutions. Melliès found a counter-example to the termination of simply typed lambda-calculus with explicit substitutions. Thérèse Hardin with Gilles Dowek and Claude Kirchner worked on higher-order unification via explicit substitutions.

3.7.2 Persons and exchanges

INRIA Rocquencourt participants are: Jean-Jacques Lévy, Damien Doligez, Eric Duquesne, Georges Gonthier, Thérèse Hardin, Luc Maranget, Paul-André Melliès and Didier Rémy. Part of the work is subcontracted to the University of Bologna: Andrea Asperti.

Benjamin Pierce has visited twice INRIA-Rocquencourt and worked with Didier Rémy of issues related to the Pict implementation. Ian Mackie made a 1-month visit to INRIA.

3.7.3 Perspectives, work in progress

In June 1994, Damien Doligez and Georges Gonthier started a formal proof of the concurrent garbage collector algorithm that they have presented at the CONFER workshop in Amsterdam and at POPL'94 [7]. This proof is sufficiently complicated to require a mechanical treatment, since already many bugs have already been found in a previous version of the algorithm with a manual proof. This proof demands more logic than any finite domain analysis prover could provide.

The proof uses TLA framework of Lamport on top of the TLP system of Guttag, Horning and Engberg. TLP, although very weak from a functional point of view, provides an Emacs interface and a preprocessor for the syntax of TLA. The proof is long, very combinatorial, and heavily uses rewritings. It will be a good benchmark when recoded in HOL and/or the Calculus of Constructions. At present, the invariants of memory layout already needs 600 lines and 400 rules of TLP. However, this proof is very risky, since such a long proof in TLP has not been done before. (The work of Doligez and Gonthier has been reported in the first annual CONFER report under the area programming languages).

In the area of abstract reduction systems, an axiomatic proof of strong normalisation still has to be discovered. The normalisation proofs are extremely complicated in the λ-calculus. It would be splendid to get an intuitive proof, making us able to understand not only strong normalisation of the Curry simply typed calculus, but also of second order. It is a difficult game where the fine structure of bound variables and binders still has to be discoverered. Work done in optimal reductions for the λ-calculus and Interaction Systems should greatly help.

In Interaction systems, it is also necessary to finish the theory. For instance, are there supersets of Interaction systems with the same nice properties?

3.7.4 Publications, technical reports

[1] A. Asperti. Linear logic, comonads, and Optimal Reductions. Fundamenta Informaticae, Special Issue devoted to Categories in Computer Science (invited paper). V.22, n.1, 1994.

[2] A. Asperti and C. Laneve. Interaction systems I: The theory of optimal reductions. Mathematical Structures in Computer Science. To appear.

[3] A. Asperti and C. Laneve. Paths, computations and labels in the λ-calculus. Theoretical Computer Science, Special Issue devoted to RTA '93 (Montreal).

[4] A. Asperti and C. Laneve. The family relation in Interaction Systems. Proc. of the International Symposium on Theoretical Aspects of Computer Science (TACS'94), Sendai, Japan. April 1994.

[5] A. Asperti, V. Danos, C. Laneve, L. Régnier. Paths in the λ-calculus. Three years of communications without understandings. Proc. of the International Symposium on Logic in Computer Science (LICS'94), Paris, France. 1994.

[6] A. Asperti. $\delta \circ !\epsilon = 1$. Internal Report of the Dipartimento di Matematica, Univeristà di Bologna. 1994.

[7] Damien Doligez, Georges Gonthier, Portable, Unobtrusive Garbage Collection for Multiprocessor Systems, Twenty-First Annual ACM Symposium on Principles of Programming Languages, Portland.

[8] Gilles Dowek, Thérèse Hardin, Claude Kirchner, Higher-order unification via explicit substitutions, INRIA report.

[9] Paul-André Melliès, Typed λ-calculi with explicit substitutions may not terminate, submitted to TLCA'95.

3.8 INRIA-Sophia

3.8.1 Research directions

The aim of the INRIA group at Sophia is to study and develop theoretical frameworks for parallel computations. We have pursued the research in the areas of Calculi and Foundational Models and Abstract Machines. We mainly focus our research on the lambda-calculus and related systems, working in the following directions:

- optimal computations. Significative results have been obtained, concerning in particular the equivalence of several notions of paths in the lambda-calculus. These results allow one to prove the correctness of the implementation of optimal computations.
- lambda-calculus with multiplicities. This refinement of the lambda-calculus is a useful tool to study the relationships between the pi-calculus and the lambda-calculus. We are currently working on the denotational semantics of this calculus, and on the characterisation of its discriminating power, with respect to various observation criteria.
- chemical abstract machines. We have applied this framework to the several particular cases of parallel computations, and most notably to the weak lambda-calculus with sharing, and the call-by-need evaluation.

3.8.2 Persons and exchanges

The following researchers are involved in the project at Sophia Antipolis: G. Berry (Research Director, Ecole des Mines de Paris), G. Boudol (Research Director, INRIA), I. Castellani (Researcher, INRIA), C. Laneve (HCM Fellowship), Ch. Retoré (post-doc, INRIA).

We have active collaborations with other sites in the CONFER Project, and especially with the ENS group in Paris. G. Boudol made several visits there to work with C. Lavatelli. C. Laneve visited CWI for one week. He produced joint work with V. Danos, L. Régnier (ENS, Paris) and A. Asperti (INRIA, Rocquencourt). The group at Sophia also had visits of D. Sangiorgi (University of Edinburgh), R. Amadio (ENS, Paris) and P.-A. Melliès (INRIA, Rocquencourt).

3.8.3 Perspectives, work in progress

We are currently working on the lambda-calculus with multiplicities, in two directions: denotational semantics and observational semantics. Regarding the denotational semantics, we use a domain equation involving the construction of a domain of multisets over a given domain. This is used to interpret the resources, which are multisets of terms of the calculus. We have to relate this denotational semantics with the one given by Boudol using an adaptation of the intersection type discipline. Showing that they coincide would entail the adequacy of the denotational semantics. We also plan to investigate the full-abstraction problem for this interpretation.

The lambda-calculus has been introduced as a means to study the relationships between the pi-calculus and the lambda-calculus. We are currently investigating the question of characterising the discriminating power of multiplicities over lambda-terms, with respect to various observational semantics, involving different relations between proper termination, deadlock and divergence. Our aim is to show, using Lévy-Longo trees, that as far as the lambda-calculus is concerned, the pi-calculus and the lambda-calculus with multiplicities have the same discriminating power. We also plan to characterise some orderings on Lévy-Longo trees as observational preorders induced by the lambda-calculus with multiplicities.

The general purpose of the two research directions just described is to understand the interplay between functional and concurrent computations. Our goal is to show that functional languages are more withstanding a concurrent environment than one could think. More specifically, we aim at establishing that they are sensitive to the lack of resources, but not that much to the non-determinism inherent to parallel computations.

3.8.4 Publications, technical reports

[INRIA/Sophia/M2/1] A. Asperti, C. Laneve, *The family relation in interaction systems*, TACS Symposium, Sendai, April 1994.

[INRIA/Sophia/M2/2] A. Asperti, V. Danos, C. Laneve, L. Régnier, *Paths in the lambda-calculus*, 9th LICS, Paris, July 1994.

[INRIA/Sophia/M2/3] A. Asperti, C. Laneve, *Interaction Systems 1: The theory of optimal reductions*, to appear in Mathematical Structures in Computer Science.

[INRIA/Sophia/M2/4] A. Asperti, C. Laneve, *Interaction Systems 2: The practice of optimal reductions*, full, revised version, submitted to Theoretical Computer Science.

[INRIA/Sophia/M2/5] G. Boudol, *Some chemical abstract machines*, in "A Decade of Concurrency, Reflections and Perspectives", LNCS 803, 1994.

3.9 Università di Pisa

3.9.1 Research directions

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

Studies have been carried out on the development of a uniform framework for the π-calculus: a π-calculus with explicit substitution. It has been shown that the explicit handling of name instantiation permits to reduce the π-calculus transitional semantics to a standard SOS framework. In this perspective, an extension of the meta-theory of Structured Operational Semantics (SOS) based on formats for the SOS inductive rules has been presented. This extension allows a common treatment of SOS rules where labels of transitions do explicitly have a bit of structure and where also certain classes of predicates may occur. Besides the π-calculus, the main example of application of this meta-theory has been the research on typed transition systems. More precisely, a new paradigm, *additive concurrency*, where typed behaviours of concurrent programs are described in terms of matrix calculi is proposed.

A non interleaving model (concurrent semantics) of the π-calculus has been also developed. The concurrent semantics of the π-calculus is given in terms of graph rewriting and it is reminiscent of the definition of non interleaving CCS via Petri nets.

Finally, some efforts have been devoted to understand categorical models for terms rewriting systems. Categorical models are proved to be expressive for defining concurrent semantics of term rewriting systems in a clean algebraic way.

3.9.2 Persons and exchanges

The group at Pisa involved in the project consists of Ugo Montanari, Gianluigi Ferrari, Paola Quaglia, Fabio Gadducci and Marco Pistore.

Just before the CONFER workshop in Munich, Montanari, Ferrari, Quaglia and Gadducci visited ECRC. This visit was very useful for understanding some issues of the semantics of higher order process calculi (e.g. Facile). Members of the ECRC group (Bent Thomsen and Lone Leth) visited Pisa from 25 to 28 June. This visit was part of the cooperation which has been initiated with ECRC. Finally, David Walker visited Pisa from 7 to 12 July.

3.9.3 Perspectives, work in progress

Future work includes investigating the π-calculus with explicit substitution to cover directly some of the issues of higher order calculi. On this theme a collaboration with the group at ECRC is just started. Moreover, we will complete the development of non interleaving semantics of the π-calculus from the point of view of graph rewriting systems. Finally, further studies are planned to analyze the relationships with Action Calculi.

3.9.4 Publications, technical reports

1. Ferrari, G., Montanari, U., Quaglia, P., A π-calculus with Explicit Substitution: the Late Semantics, In Proc, MFCS'94, LNCS 841, 1994.

2. Ferrari, G., Montanari, U., Quaglia, P., A π-calculus with Explicit Substitution. Full version submitted for publication, 1994.

3. Ferrari, G., Montanari, U., Typed Additive Concurrency, Submitted for Pubblication, August 1994.

4. Ferrari, G., Montanari, U., Turning SOS Rules into Operations, August 1994.

5. Corradini, Gadducci F., Montanari, U., Prime event structures and categorical models of term rewriting, August 1994.

6. Montanari, U., Pistore, M., Concurrent Semantics for the π-calculus, July 1994.

3.10 SICS

3.10.1 Research directions

During the second year of CONFER, work at SICS has mainly been conducted on Programming Languages and on Calculi. In the former, Björn Lisper has done foundational work on correctness criteria for unfolding nondeterministic constructs at compile-time. In the latter, Björn Victor and Faron Moller have developed the Mobility Workbench to cater for the full polyadic π-calculus and significantly improved the efficiency; in order to establish correctness of the tool the theory of weak open bisimulations had to be developed. This work has resulted in one Licentiat thesis (Björn Victor).

Within the areas Foundations and Calculi, Joachim Parrow and Mads Dam have updated their contributions to the first year deliverable (on model checking, axiomatisations, and interaction diagrams) for journal publication.

SICS is also part of the related ESPRIT projects CONCUR2 and LOMAPS. Within CONCUR2 we are active in case studies, tool construction and algorithms for infinite state spaces. The two first are relevant also for CONFER: one of the CONCUR2 case studies used the polyadic π-calculus to analyse resource allocation protocol in high-speed networks, and the Mobility Workbench is a joint deliverable of CONCUR2 and CONFER, applying the tool builder expertise from CONCUR2 to the emerging theories from CONFER.

Within LOMAPS we have begun a more concentrated effort on temporal logics for the pi-calculus and higher order processes. This work is relevant to CONFER and our work is to some extent based on [D93]. Spin offs of this work under the LOMAPS operational semantics work theme include a decidability result for late and early, strong and weak bisimulation equivalence applied to finite control pi-calculus agents.

3.10.2 Persons and exchanges

During the second year at SICS, Mads Dam, Lars-Henrik Erikson, Björn Lisper, Faron Moller, Joachim Parrow and Björn Victor have been active on CONFER. Mads Dam and Lars-Henrik Erikson moved from CONFER to LOMAPS during fall 1993, so their deliverables will be reported under that project. Faron Moller arrived from Edinburgh in March 1994, and presently works on both CONCUR2 and CONFER.

The collaboration between Joachim Parrow and Davide Sangiorgi (Edinburgh) has continued and resulted in an updated version of the work on axiomatisations.

We have participated in the CONFER workshops. Björn Victor has presented a CONFER paper at CAV'94 in Stanford, California.

3.10.3 Perspectives, work in progress

Future work includes investigating the weak open bisimulation equivalence — its axiomatisation, and its relation to the weak late bisimulation equivalence; developing algorithms for finding minimal distinctions and matchings for which two agents are open equivalent; developing modal logics and model checking for the open bisimulation; investigating the open bisimulation equivalence in the presence of a *mis*match operator.

We will also investigate to what extent our results on unfoldings can be applied to other reduction systems than such given by plain term rewriting systems. Here, interaction with research in the other areas of the project could be helpful. Once it is understood how our results can be extended, experiments should be undertaken with some real concurrent language.

3.10.4 Publications, technical reports

P93 J. Parrow. "Interaction Diagrams". In de Bakker, de Roever and Rozenberg (Eds.): *A Decade of Concurrency*, REX school/Symposium, The Netherlands June 1993, Pages 477–508.

PS93 J. Parrow and D. Sangiorgi: "Algebraic Theories for Name-Passing Calculi". In de Bakker, de Roever and Rozenberg (Eds.): *A Decade of Concurrency*, REX school/Symposium, The Netherlands June 1993, Pages 509–529. Published as Springer Verlag LNCS 803 (1994). Extended version accepted for publication in *Information and Computation*.

D93 M. Dam. "Model Checking Mobile Processes". In Best (Ed.): *Proceedings of CONCUR'93*, pages 22–36 Published as Springer Verlag LNCS 715 (1993). Extended version accepted for publication in *Information and Computation*.

VM94 B. Victor and F. Moller. "The Mobility Workbench — A Tool for the π-Calculus". In Dill (Ed.): *Proceedings of CAV'94*, pages 428–440, Published as Springer Verlag LNCS 818 (1994).

V94 B. Victor. "A Verification Tool for the Polyadic π-Calculus". Licentiat thesis, Department of Computer Systems, Uppsala University, May 1994.

3.10.5 Software

The Mobility Workbench (MWB) is a tool for manipulating and analysing mobile concurrent systems described in the π-calculus. The new version of the MWB supports the polyadic π-calculus, and has vastly improved performance (for some examples with a factor more than 100) over the version reported in last year's progress report.

Chapter 4

Deliverables

4.1 Workshop 3

At CWI in Amsterdam, with 35 participants.

Tuesday, September 28

09:30	coffee
10:00-10:15	Jean-Jacques Lévy opening
10:15-11:00	Ugo Montanari Process calculi as (concurrent) Term Rewriting Systems
11:00-12:00	Jan Bergstra Process algebra with combinators
12:00-12:30	Georges Gonthier Portable, unobstructive garbage collection for multiprocessor systems
12:30-14:00	lunch
14:00-14:45	Bent Thomsen Towards compositional reasoning about Facile programs
14:45-15:15	Andre Kramer Calumet: a desk-top conferencing system
15:15-15:45	tea
15:45-16:45	Simon Gay Categories for asynchronous processes
16:45-17:30	Jan Willem Klop Modular term graph rewriting

Wednesday, September 29

09:00-09:30	coffee
09:30-10:30	Gérard Berry
	Preemption in concurrent systems
10:30-11:00	coffee
11:00-12:00	Gérard Boudol
	Lambda calculus with multiplicities
12:00-12:30	Paul-André Melliès
	An abstract theorem of finite developments
12:30-14:00	lunch
14:00-14:30	Andrea Asperti
	About the optimal implementation of lambda calculus
14:30-15:15	Cosimo Laneve
	About paths in lambda terms
15:15 - 15:45	tea
15:45-16:45	Vincent Danos
	A proof that Lambda* (Girard's GOI algebra)
	computes regular paths
16:45-17:30	Laurent Regnier
	Connection between dynamic graphs and sharing graphs
19:00	dinner

Thursday, September 30

09:00-09:30	coffee
09:30-10:30	Raja Nagarajan
	Types for concurrency
10:30 - 11:00	coffee
11:00-12:00	Samson Abramsky
	Full abstraction for PCF
12:00	Bjorn Victor
	demo
12:30-14:00	lunch
14:00-15:00	Davide Sangiorgi
	Locality and true-concurrency in pi-calculus
15:00 - 15:30	tea
15:30-16:30	Robin Milner
	Action Structures

Friday, October 1

REVIEW DAY

4.2 Workshop 4

At ECRC in Munich with 37 participants.

Monday, April 18

08.45-09.15	Registration
09.15-09.25	Welcome and Opening Alessandro Giacalone
09.25-09.35	Introduction Jean-Jacques Lévy
09.35-10.05	Actors and Interaction Categories: Preliminary Investigations Greg Meredith
10.05-10.35	On bisimulation in the pi-calculus David Walker
10.35-11.00	Coffee Break
11.00-11.45	Translating Core Facile Roberto Amadio
11.45-12.15	A pi-calculus with logic variables and its application to higher-order concurrent constraint programming Gert Smolka
12.15-12.45	The Geometry of Interaction Machine Ian Mackie
12.45-14.00	Lunch
14.00-15.00	Compositional Verification of Deadlock-freedom Raja Nagarajan
15.00-15.30	Unfolding of Programs with Nondeterminism Bjoern Lisper
15.30-15.50	Tea Break
15.50-16.50	A typed Calculus for Synchronous Processes Simon Gay
16.50-17.05	Business meeting
17.05-17.30	System Demo Gert Smolka

Tuesday, April 19

09.00-09.15	Typed explicit substitutions may not terminate Paul-Andre Mellies
09.15-10.00	Higher Order Bisimulations and Axiomatisability Peter Sewell
10.00-10.45	Interaction Orders as Games Michael Huth
10.45-11.10	Coffee Break
11.10-11.55	Pomset Logic Christian Retore
11.55-12.40	Localities and Failures Roberto Amadio
12.40-14.00	Lunch
14.00-15.30	Discussion Session: "What is Mobility?"
15.30-15.50	Tea Break
15.50-16.50	A pi-calculus with explicit substitutions Paola Quaglia
16.50-17.15	System Demo of Facile Jean-Pierre Talpin & Andre Kramer

Wednesday, April 20

09.15-10.15	Control Structures Robin Milner
10.15-11.00	Typed Additive Concurrency GianLuigi Ferrari
11.00-11.20	Coffee Break
11.20-12.05	On control Structures Alexander Mifsud
12.05-12.40	An overview of Facile Antigua Release Bent Thomsen
12.50-14.00	Lunch
14.00-14.45	Process Algebra with iteration and nesting Jan Bergstra
14.45-15.45	On the proof net problem for Intuitionistic Linear Logic Francois Lamarche
15.45-16.00	Closing Remarks and Tea Break

4.3 Software deliverables

Several pieces of software have been constructed during the first and second year of CONFER. Some of them have already been made available via the CONFER ftp site at Imperial College while others are expected to be made available in the near future. Note that the software deliverables are only planned for at Milestone 3, but some of the pieces of software were actually shown at the first year review in Amsterdam. Since then most of the software has been further improved and stabilised.

The following is a listing of constructed software during the first two years of CONFER:

- Facile programming language
  ECRC — A. Giacalone, F. Knabe, A. Kramer, T.M. Kuo, L. Leth, S. Prasad, B. Thomsen. Also with contributions of P. Cregut, P-Y. Chevalier, J.-P. Talpin and C. Crampton.

- Calumet
  ECRC — A. Kramer, J.-P. Talpin. Also with contributions from K. Ahlers and P. Marchal.

- Prototype compiler for λ-calculus, based on graph reduction
  INRIA — A. Asperti.

- Portable, unobtrusive garbage collection for multiprocessor systems
  INRIA — D. Doligez, G. Gonthier, J.J. Lévy.

- Lilac: a prototype functional programming language based on Linear Logic
  Imperial College — I. Mackie.

- Pict – Typed higher-order programming language based on π-calculus
  University of Edinburgh — B. Pierce, D. Rémy, D. Turner.

- The Mobility Workbench (MBW) — a tool for manipulating and analyzing mobile concurrent systems described in the π-calculus
  University of Edinburgh — Faron Moller, Davide Sangiorgi, SICS — Björn Victor.

Some of these pieces of software will be demonstrated at the CONFER workshop at Imperial College in London and at the Annual review.

As mentioned software deliverables are only due at Milestone 3 at which point the descriptions will be provided as stipulated in the technical annex.

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 is twofolded. Firstly, it studies the abstraction of interaction. This is largely covered by Milner's action structures. It is also work by Ferrari and Montanari. Secondly, it treats calculi with bound variables in many ways. Very little is known in literature on these calculi, much less than for standard algebraic rules. This second part corresponds to works on optimal reductions, abstract reduction systems, explicit substitutions, CRS, cyclic CRS, and again action structures.

5.1.1 Action Structures

There has been strong progress in action structures. The main contribution has been the definition of *action calculi*, which are concrete action structures representing many familiar calculi (Petri nets, π-calculus, λ-calculus) in a single framework. The foundations for this work were laid in the preceding year, but many publications and new technical developments occurred within the review period. Significant work has also been done in defining and analysing the category of *control structures*, which is intended to accommodate the model theory of action structures. An alternative presentation of action calculi in which names are less prominent, and the connection between higher order action calculi and the lambda calculus have been investigated.

1. *Action calculi, or concrete action structures*, by Robin Milner, Proc. MFCS Conference, Gdansk, Poland, LNCS, Vol 711, September 1993, 105–121.

   Abstract Action structures have previously been proposed as an algebra for both the syntax and the semantics of interactive computation. Here a class of concrete action structures called action calculi is identified, which can serve as a non-linear syntax for a wide variety of models of interactive behaviour. They generalise a previously defined action structure for the π-calculus. One action calculus differs from another only in its generators, called *controls*.

   Several action calculi are presented, giving essentially the same power as the π-calculus. An action calculus is also outlined for PT nets – a class of Petri nets – parametrized upon their places and transitions.

Finally, action calculi are characterized as the free algebras in a sub-variety of action structures, namely those which satisfy certain additional axioms.

2. *Higher-order action calculi*, by Robin Milner, Proc. CSL conference, Swansea, October 1993.

   Abstract Action calculi are a broad class of algebraic structures, including a formulation of Petri nets as well as a formulation of the π-calculus. Each action calculus $\mathsf{HAC}(\mathcal{K})$ is generated by a particular set $\mathcal{K}$ of operators called *controls*. The purpose of this paper is to extend action calculi in a uniform manner to higher-order. A special case is essentially the extension of the π-calculus to higher order by Sangiorgi. To establish a link between the interactive and functional paradigms of computation, a variety of the λ-calculus is obtained as the extension of the smallest action calculus $\mathsf{HAC}(\emptyset)$.

   The dynamics of higher-order action calculi is presented, blending communication –for example in process calculi– with reduction as in the λ-calculus. Strong normalisation is obtained for reduction. A set of equational axioms is given for higher-order action calculi. Taking the quotient of $\mathsf{HAC}(\emptyset)$ by a single extra axiom $\boldsymbol{\eta}$, a cartesian-closed category is obtained.

   An ultimate goal of the paper is to combine process calculi and functional calculi, both in their formulation and in their semantics.

3. *Pi-nets: a graphical form of pi-calculus*, by Robin Milner, Proc. ESOP'94, LNCS Vol788, Springer-Verlag, April 1994, 26–42.

   Abstract An action calculus which closely corresponds to the π-calculus is presented in graphical form, as so-called π-*nets*. First an elementary form of π-net, with no sequential control, is presented. Then, using a construction by Honda and Tokoro, it is shown informally that by adding a single control construction **box** to elementary π-nets, the sequential control present in the π-calculus can be recovered. (Another construction, **rep**, provides replication.) The graphical presentation suggests a few interesting variants of this control regime, which are studied briefly. The main purpose of the paper is to explore informally the power and utility of graphical forms of the π-calculus, in the context of action calculi. It also suggests that graphical forms of other action calculi should be explored.

4. *Action calculi IV: molecular forms*, by Robin Milner, November 1993.

   Abstract Action calculi were introduced as a subclass of action structures comprising a variety of computational calculi. Each action calculus $\mathsf{AC}(\mathcal{K})$ is determined by a set $\mathcal{K}$ of *control* operators, together with their *reaction rules*.

   The claim that action calculi provide a valuable framework rests upon a technical result: that each $\mathsf{AC}(\mathcal{K})$ can be presented either as a term algebra subject to a few simple equational axioms (independent of $\mathcal{K}$), or as an algebra of *molecular forms*. The latter play a dual role; thay act as *normal forms* for the term algebra under the axioms (and hence play an important part in the theory of action calculi), and they serve as a natural and modular mode of expression, even suitable for programming.

   The technical result that these two presentations are isomorphic is proved in detail here.

   Higher-order action calculi have a richer algebraic theory than action calculi; one axiom is strengthened and a further one is added. They also have a molecular presentation. Again, the isomorphism of two presentations of higher-order action calculi is proved here in detail.

5. *Action calculi V: reflexive molecular forms*, by Robin Milner (with Appendix by Ole Jensen). June 1994.

   Abstract The key property of action calculi is that each calculus $\mathsf{AC}(\mathcal{K})$ can be presented in two ways: as the quotient of a term algebra by some simple axioms, and as an algebra of *molecular forms*. in a previous paper the isomorphism between the two presentations is proved in detail.

   The present paper adapts that proof to admit *reflexion* into action calculi, in the form of a unary reflexion operator $\Uparrow_p$ for each prime arity p. This operator was first studied axiomatically by Stefănescu in 1986. His work was in the context of Elgot's iterative theories; he called the operator "feedback", and pointed out some of its advantages compared with Elgot's iteration operator. The work reported below was done in ignorance of Stefănescu's results, but the results are essentially the same. The main difference is that the algebra of action calculi includes an abstraction operator, so an additional axiom is needed to express how feedback relates to this operator.

6. *Action calculi VI: strong normalisation at higher-order*, by Robin Milner, December 1993.

   Abstract This paper has a single purpose: to prove that β-reduction is strongly normalising and confluent in higher-order action calculi. The proof adapts the standard method for the simply-typed λ-calculi summarised by Barendregt in his book.

7. *Control Structures*, by Robin Milner, Alex Mifsud and John Power, June 1994.

   Abstract Action calculi have been defined as a broad class of action structures with added structure; they are syntactic in nature. Each action calculus $\mathsf{AC}(\mathcal{K})$ is determined essentially by a set $\mathcal{K}$ of *controls*. Different concrete models of concurrent computation such as Petri nets, the typed λ-calculus and the π-calculus, are obtained by different choices of control set $\mathcal{K}$. The aim of this work is to define a class (in fact a category) of action structures with added structure which can be regarded as possible semantic interpretations of the calculus $\mathsf{AC}(\mathcal{K})$; thus $\mathsf{AC}(\mathcal{K})$ is characterized as the initial object in this category. The objects are called *control structures* over $\mathcal{K}$.

   The paper contains the technical formulation of the category of control structures and their homomorphisms, and the proof that $\mathsf{AC}(\mathcal{K})$ is initial. Moreover, the category is closed under factorisation by arbitrary congruences on control structures.

   The paper concludes with an outline of work in progress to identify particular control structures of interest, such as the quotient of an action calculus for the π-calculus under a bisimulation congruence.

5.1.2 Structured Operational Semantics Revisited

Ferrari and Montanari address the problem of extending the SOS metatheory to handle actions with structure. To this purpose they present a generalization of De Simone format called *Algebraic De Simone Format*, AdS for short, where labels of transitions form an algebra with several operations called *Observation Structure*. The idea is that the observation structure specifies the common structure of the experiments which are allowed in the concrete observational models of process behaviours. In other words, the observation structure individuates a class of algebras which can be regarded as the class of possible observational models. With a series of examples they show how several observational models of process calculi are handled within this framework simply by providing suitable concrete models of the observation structure. It is proved that, for all models of the observation structure, bisimulation is always a congruence for all the process combinators whose behaviour can be described within the AdS format. Moreover, the

strategy proposed by Aceto, Bloom and Vaandrager, which yields finite algebraic laws of bisimulation congruence for any process calculus specified by structural rules in the GSOS format, is generalized to the case of actions with structure. The main benefit of this generalization is that it makes less *ad hoc* the use of auxiliary operators in the axiomatic characterization of bisimulation semantics of *static* and *mobile* process calculi. Finally, a smooth generalization of the notion of observation structure permits to deal with SOS semantics in which certain classes of predicates may occur.

5.1.3 Optimal reductions

In 1992, the works of Lamping and Gonthier renewed this topic, which has already been described in last year report. However, two very long papers by Martín Abadi, Georges Gonthier and Jean-Jacques Lévy on the λ-calculus and linear logic are still under rewriting for journal versions.

Asperti and Laneve pursued the work on extending it to interaction systems. These are higher order rewriting systems generalizing Lafont's interaction nets, retaining the idea of binary interaction between constructors and destructors through distinguished, complementary ports. Interaction systems also form a sub-class of combinatory reduction systems of Klop, containing the λ-calculus. The theoretical aspects of Interaction Systems are investigated in [2, 4]. A major part of the research consisted in adapting and extending to interaction systems the notion of optimal computations, as introduced by Lévy for the λ-calculus.

In particular, it has been shown that the Lamping-Gonthier optimal implementation of the λ-calculus can be smoothly extended to interaction systems. A prototype compiler based on this technique has been implemented by Asperti and Laneve. The compiler implements Lamping-Gonthier's reduction rules following a sort of "lazy" strategy (reacall however that a whole *family* of redex is reduced at a time). Reduction is pursued up to the weak head normal form of the term, by iterating the reduction of (the redex-family of) the leftmost outermost redex in the term. The leftmost outermost redex is looked for by maintaining an auxiliary stack for the main spine of the term (in a way similar to typical supercombinators implementations, such as the G-machine).

Some students under the supervision of of Asperti are actually working to extend the original system (regarding pure lambda calulus) with new features. In particular, Andrea Naletto implemented the garbage collector, while Cecilia Giovannetti is adding primitive types (integers, boolean etc.) and some additional control structure (if-then-else, recursion) in the spirit of Interaction Systems. Asperti is moreover investigating new reduction rules for "safe operators" [1] that seems to drastically reduce the actual performance of the evaluation (partially solving the well known problem of accumulation of control operators).

In the same theme, Asperti and Laneve have further studied Lévy's notion of family of redexes of a λ-term, characterizing this notion by means of paths in the term. The idea is that redexes in the same family are created by contractions on a unique common path in a suitable graphical representation of the initial term (where a bound variable refers to the corresponding abstraction) [3]. This provides new evidence about the common nature of redexes of the same family, and therefore also about the possibility of sharing their reduction. Their characterization may be seen as an alternative viewpoint on graph reduction techniques of Lamping and Gonthier implementing optimal reductions. This point, together with the relation with a different notion of path inspired by Girard's Geometry of Interaction, has been clarified in [5].

5.1.4 Higher-order rewriting

Roughly, this consists of term rewriting with bound variables. Various calculi featuring in the project have this characteristic: lambda calculi, pi-calculi. In the past year, essential progress has been made: Van Raamsdonk proved in cooperation with Vincent van Oostrom that also weakly orthogonal higher-order rewrite systems are confluent, thereby extending the well-known confluence for orthogonal systems. (A preliminary announcement of this result was already in last year's report.) The paradigm of a weakly orthogonal higher-order rewrite system is lambda calculus with beta and eta rule; also the typed versions such as system F with beta and eta. Also a useful *modularity* result was obtained: the disjoint union of left-linear confluent higher-order rewrite systems is again confluent. Actually, a refined version of that theorem states that confluence is also preserved in combinations of left-linear confluent higher-order rewrite systems when trivial critical pairs are allowed, generated by rules from different systems. (Inside each system separately there may be nontrivial critical pairs.)

Furthermore, Van Raamsdonk and Van Oostrom have worked out a general framework for higher-order rewriting (HORSs, Higher Order Rewriting Systems), where there is the parameter of the *substitution calculus* that is employed. For instance, Nipkow's Higher-order Rewrite Systems (HRSs) employ as substitution calculus the simply typed lambda calculus, while Klop's CRSs have underlined lambda calculus (as in *Finite Developments*) as substitution calculus. The substitutions calculus is the underlying mechanism governing how actual rule instantiations are formed. Much of this work is reported in Van Oostrom's Ph.D. thesis, written under supervision of J.W. Klop at the Free University Amsterdam; this work formally is not in the project, but is influential for our work in the project.

5.1.5 Term Rewriting and Lambda Calculus with explicit recursion

The second major research direction of CWI continued the work initiated last year on term graph rewriting and cyclic lambda graph rewriting. This work is done in cooperation with Zena Ariola (University of Oregon). The origin of this work was in the former ESPRIT BRA Semagraph, now a WG. The addition which makes it interesting in Confer, is the consideration of bound variables in the form of recursion variables: a term graph can be described as a system of recursive equations, in fact as a subclass of the well-known and much studied Recursive Program Schemes. The issue of bound variables becomes prominent when nested systems of such recursion systems are considered. Added to lambda calculus, we have in fact lambda calculus with *explicit recursion* - somewhat in analogy with the lambda calculi with *explicit substitution* or lambda sigma calculi that are also studied in our project. The objects of interest are *cyclic lambda graphs with a modular structure*. Graphically, the modular structure (arising from nesting of recursion systems) consists of boxes, possibly nested, around parts of the graph; the inside of a box cannot be addressed directly, but only through its root. The β-rule gets in this framework an extremely simple form:

$$(\lambda\alpha.M)N \to \langle M \mid \alpha = N\rangle$$

where the role change in the binding of alpha is remarkable: first bound by lambda, later by the recursion construct. In functional programming these constructs are well-known as the *let* and *letrec*; but as yet in our opinion the theory behind these constructs is not developed. It turns out for instance that lambda calculus with letrec, or our lambda calculus with explicit recursion, has a highly nontrivial confluence problem, consisting in first designing suitable rewrite rules and second in proving it confluent. In fact, a naive but *obvious* version of defining β-reduction on cyclic lambda terms turns out to be essentially non-confluent, unless one forbids the presence of certain cycles in the graph. Another way out is the introduction of boxes as a

means of restricting all too liberal *copying* of parts of the graph. Interestingly, some of the rules are similar to the rules for lambda sigma calculus; and our calculus may be seen as a lambda calculus with explicit cyclic substitutions.

5.1.6 Abstract reductions systems

Paul-André Mellies is writing his Phd dissertation about Abstract Rewriting Systems. The thesis contains axiomatic proofs of the finite development lemma, the standardisation theorem (with critical pairs or not), and the strong normalisation theorem for simple types. He revisited any of these subjects this year, and many parts of the work were improved. The standardisation part was particularly studied, many axioms were weakened from the LICS'92 joint work with Gonthier and Lévy. leading yet to stronger properties and structures.

He has also worked about the property of strong normalisation. The better understanding of this property for simply typed systems gave him some insight about the termination of the typed $\lambda\sigma$-calculus. He showed that infinite computations are possible on a typed $\lambda\sigma$-term. This gives a counter-example to the general conjecture that typed λ-calculi with explicit substitutions do strongly terminate. Similar examples exist in the Categorical Combinators calculus of Curien and in the $\lambda\sigma_{\Uparrow}$-calculus of Hardin and Lévy. He presented these examples during the CONFER workshop in München (April 1994). Two new questions arise from this work: how should to handle composition of substitutions to obtain both confluence on open terms and termination? What are the terminating strategies inside existing typed λ-calculi with explicit substitutions?

5.1.7 Explicit substitutions

As just above mentioned, Paul-André Melliès found a counter-example to the termination of simply typed lambda-calculus with explicit substitutions.

Thérèse Hardin with Gilles Dowek and Claude Kirchner worked on higher-order unification via explicit substitutions. It is well-known that higher order unification consists in solving equations in some $\beta\eta$ theory. Instead of creating artificial β-redexes (as in the resolution methods defined by Huet in his dissertation), one can show that the explicit treatment of substitutions is more natural, and leads to higher order unification in the equational theory of the $\lambda\sigma$-calculus, where with use of grafting one solves equations in a more first-order way. The natural outcoming algorithms are expressed by transformations rules. In these algorithms, the distinction between substitutions initiated by reduction and substitutions of unification variables permits to avoid the encoding of the scoping constraints due to $\beta\eta$-reduction, which is one of the standard burdens of previous algorithms. The resulting solving process decomposes Huet's algorithm in elementary steps that avoids unnecessary ones and are close to the implementation level.

5.1.8 Categorical Models of Term Rewriting Systems

In the latest years there has been a growing interest towards categorical models for term rewriting systems. Categorical models are obtained by interpreting rewriting steps as arrows of a category or as 2-cells of a 2-category. These models can be used to naturally equip rewriting systems with a concurrent semantics in a clear algebraic way.

Corradini, Gadducci and Montanari studied and compared some (concurrent) models previously proposed in the literature. It is showed that these models fail to capture a *fully satisfactory* notion of concurrent semantics for rewriting systems. For instance, it turns out that the derivation space of Meseguer's Concurrent Term Rewriting (i.e., the set of coinitial computations ordered by prefix) associated to each term fails in general to form a prime event structure. In-

stead, the resulting derivation space in Stell's model of Sesqui categories prime event structure, but too few computations are identified, such that only *disjoint concurrency* can be expressed.

5.1.9 Interrelations between sites and to other areas. Future work

There is an active collaboration between INRIA-Rocquencourt, INRIA-Sophia, Imperial College and ENS in the domain of optimal reductions (joint papers, visits, constant connexion). In CRS and abstract reduction systems, CWI and INRIA-Rocquencourt cooperate (Melliès is on leave to Amsterdam).

Future work is very promising in Actions Structures and Calculi and in theory of bound variables. Right now, it is impossible to guess which setting will win. Some work has to be done to connect various theories, especially with the axiomatics of Action Calculi.

5.1.10 Reports on foundational models

[1] A. Asperti. Linear logic, comonads, and Optimal Reductions. Fundamenta Informaticae, Special Issue devoted to Categories in Computer Science (invited paper). V.22, n.1, 1994.

[2] A. Asperti and C. Laneve. Interaction systems I: The theory of optimal reductions. Mathematical Structures in Computer Science. To appear.

[3] A. Asperti and C. Laneve. Paths, computations and labels in the λ-calculus. Theoretical Computer Science, Special Issue devoted to RTA '93 (Montreal).

[4] A. Asperti and C. Laneve. The family relation in Interaction Systems. Proc. of the International Symposium on Theoretical Aspects of Computer Science (TACS'94), Sendai, Japan. April 1994.

[5] A. Asperti, V. Danos, C. Laneve, L. Régnier. Paths in the λ-calculus. Three years of communications without understandings. Proc. of the International Symposium on Logic in Computer Science (LICS'94), Paris, France. 1994.

[6] A. Asperti. $\delta \circ !\epsilon = 1$. Internal Report of the Dipartimento di Matematica, Univeristà di Bologna. 1994.

[7] Damien Doligez, Georges Gonthier, Portable, Unobtrusive Garbage Collection for Multiprocessor Systems, Twenty-First Annual ACM Symposium on Principles of Programming Languages, Portland.

[8] Gilles Dowek, Thérèse Hardin, Claude Kirchner, Higher-order unification via explicit substitutions, INRIA report.

[9] Paul-André Melliès, Typed λ-calculi with explicit substitutions may not terminate, submitted to TLCA'95.

[10] Corradini, A., Gadducci F., Montanari, U., Prime event structures and categorical models of term rewriting, August 1994.

[11] Ferrari, G., Montanari, U., Turning SOS Rules into Operations, August. 1994.

[12] *An action structure for synchronous π-calculus*, Robin Milner, Proc. FCT Conference, Szeged, Hungary, LNCS, Vol 710, August 1993, 87–105.

[13] Action calculi, or concrete action structures", Robin Milner, Proc. MFCS Conference, Gdansk, Poland, LNCS, Vol 711, September 1993, 105–121.

[14] Higher-order action calculi, Robin Milner, to appear in Proc. CSL conference, Swansea, October 1993.

[15] *Pi-nets: a graphical form of pi-calculus*, Proc. ESOP'94, LNCS Vol788, Springer-Verlag, April 1994, 26–42.

[16] *Action calculi IV: molecular forms*, Robin Milner, November 1993.

[17] *Action calculi V: reflexive molecular forms*, Robin Milner, June 1994. (with Appendix by Ole Jensen)

[18] *Action calculi VI: strong normalisation at higher-order*, Robin Milner, December 1993.

[19] *Control Structures*, Robin Milner, Alex Mifsud and John Power June 1994.

5.2 Calculi

This section reports on the work of the second year of CONFER in the area of Calculi. A considerable effort from various sites has been made in this area, resulting in 18 reports and publications. Although each of these papers contains original contributions, one should note that they constitute a coherent corpus of research. This shows that various sites of CONFER share common interest in some specific themes of the project. From this point of view, one must especially mention the cooperation between R. Amadio from ENS-Paris and B. Thomsen, L. Leth and S. Prasad from ECRC-Munich. They produced joint work which is on the boundary of the areas of Calculi and Programming Languages.

The work done in the area of Calculi mainly deals with the semantics of the π-calculus and other related formalisms. However, to organise the presentation of this work, it is convenient to distinguish three main topics:

- bisimulations,
- non-interleaving semantics,
- relating calculi.

This division should not obliterate the connections between the various topics: for instance some works on "non-interleaving semantics" show that the π-calculus is expressive enough to encode these semantics within the standard bisimulation approach. The work on "relating calculi" is also concerned with proofs of properties of processes, and this is the main subject of the works presented in our first section.

5.2.1 Bisimulations

The well-known notion of *bisimulation equivalence*, due to Milner and Park, provides a semantics for the class of calculi where the agents' behaviour consists in performing actions along with state transitions. Moreover, the notion of bisimulation provides a very powerful *proof technique* for establishing properties of agents. This proof technique is implemented in software verification tools that can be used to perform sophisticated verifications. As we shall see, all the works presented below are concerned with this use of bisimulations as a basis for formal proofs.

It has been noted that for higher-order calculi of concurrent processes, the right notion of bisimulation is not so easy to define, and that some variations are possible. This applies to the π-calculus, where one passes names during a communication, and even more crucially to calculi, like CHOCS or HOπ, where agents can be transmitted in a communication. There are several difficulties. One, specific of the π-calculus, is due to the fact that there is some freedom in the determination of the moment at which an actual value is substituted for a parameter name. That is, the notion of action and resulting state is not as clear as in CCS for instance. Another difficulty is to deal with the private names in an appropriate way.

Three papers – see the references below – address the problem of defining bisimulations for the π-calculus. In [AmaAit94], the authors first give a characterisation of the original "early" bisimulation of Milner, Parrow and Walker, by means of an equivalence where the output actions are observed by means of contexts of the calculus. This is similar in spirit to the "barbed congruence" of Milner and Sangiorgi, which was shown to coincide with the "early" congruence. Then Amadio and Ait-Mohamed argue that, from the point of view of automated verification, the "early" and "late" bisimulations are not very convenient. Then they propose to treat an input name as a "logical variable", subject to some constraints. This leads them to

the definition of the *uniform* bisimulation which is stronger than the previously known notions of bisimulations.

A general framework to deal with the various notions of π-calculus bisimulations has been set up by Montanari et al. The papers [FerMonQua94a, FerMonQua94b] develop the work started in the first year of CONFER. The authors apply Curien and Lévy's construct of *explicit substitution* to the π-calculus. In this unified framework one may retrieve the late and early semantics by defining suitable instantiation strategies. The authors also introduce a *lazy* semantics, similar to the "uniform" semantics of Amadio and Ait-Mohamed, and to the "open" bisimulation of Sangiorgi (see last year's report, [San93d]).

The third paper addressing π-calculus bisimulations is the one of Walker [Wal94a]. The purpose of the paper is to characterise Sangiorgi's "open" bisimulation, which is particularly well suited for computer aided verification, as a "barbed congruence". This amounts to introducing a new operator to the π-calculus, by means of which the discriminating power of the contexts is enhanced. Not surprisingly, this operator bears some resemblances to the explicit substitutions.

All these papers contribute to clarify the situation of the various possible definitions of π-calculus bisimulations. One should notice that they are all motivated by the desire of having a satisfactory semantic theory which is well suited for verification purposes. Although some further studies will certainly emerge, one may think that this topic is rather well understood by now. Regarding the point of defining bisimulations for higher-order calculi, there is a contribution by Sangiorgi [San94a]. This work develops the idea of *barbed congruence* of Milner and Sangiorgi, applying it to the case of agent passing calculi (in particular to HOπ), where it seems to be the most appropriate notion – if not the only one. In higher-order calculi, one cannot simply use the standard notion of bisimulation, because agents perform communication actions that carry agents. Therefore one should at least regard two actions carrying equivalent agents as identical. Sangiorgi convincingly argues that this is not enough in the presence of private communication names. Then he proposes a very simple notion of equivalence which, roughly speaking, identifies agents that have the same communication capabilities in every context. Like Morris' contextual equivalence, barbed congruence may be defined for any calculus equipped with a notion of reduction and a convergence predicate. However, Sangiorgi observes that this semantic equivalence is not very well suited, as it stands, for practical verifications, because of the universal quantification over contexts it involves. He then shows that barbed congruence has a direct characterisation: it coincides with the so-called *normal* bisimulation, which provides a tool for proving agents to be barbed congruent.

Two other contributions are relevant to this section on bisimulations. Both are investigating general questions about this kind of equivalence. The first, by Sewell [Sew94], addresses the problem of finite axiomatisability of the bisimulation. He shows that even for a very simple calculus of finite state processes, or more accurately a simple subset of HOπ, the bisimulation is not first order finitely axiomatisable. This contrasts with previous results concerning the algebra of finite state processes without the "nil" agent. In [San94b], Sangiorgi makes some observations on the proof technique associated with bisimulation. A bisimulation is a binary relation which is stable by transitions, and one may prove the equivalence of agents by exhibiting a bisimulation that relates these agents. However, quite often the natural candidate relation that one can think of is not stable. Therefore one is lead to consider functions over relations, such that one progresses by transitions from a relation to a function of it. Sangiorgi gives some non-trivial conditions on such functions to ensure that bisimulation is preserved. He then applies these general conditions to the proof of some non-trivial properties in CCS and the π-calculus.

References

[AmaAit94] R. Amadio and O. Ait-Mohamed: *An Analysis of π-calculus Bisimulations*, Technical Report ECRC-94-2, 1994.

[FerMonQua94a] G. Ferrari, U. Montanari and P. Quaglia: *A π-calculus with Explicit Substitution: the Late Semantics*, to appear Proc. MFCS 94, LNCS, 1994.

[FerMonQua94b] G. Ferrari, U. Montanari and P. Quaglia: *A π-calculus with Explicit Substitution*, submitted to TCS (Special Issue MFCS'94), 1994.

[San94a] D. Sangiorgi: *Bisimulation in higher-order calculi*, Proc. IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET), S. Miniato, Italy, June 94.

[San94b] D. Sangiorgi: *On the bisimulation proof method*, Tech. Rep. ECS–LFCS–94–299, University of Edinburgh.

[Sew94] P. Sewell: *Bisimulation is not finitely (first-order) equationally axiomatisable*, LICS 94, Paris, July 1994.

[Wal94a] D. Walker: *On bisimulation in the pi-calculus*, in Proc. 5th International Conference on Concurrency Theory CONCUR'94, Uppsala, August 1994.

5.2.2 Non-interleaving semantics

The non-interleaving semantics was developed for CCS-like process algebras with the idea that a semantics reducing parallel composition to sequential non-determinism is not entirely appropriate for dealing with distributed systems, that are supposed to run on decentralised systems. On the other hand, a clear advantage of the standard approach is its nice mathematical theory, allowing for rigorous proofs of properties of concurrent systems. An obvious direction of research was then to investigate the meaning of non-interleaving semantics for higher order calculi. One may distinguish two different approaches to the non-interleaving semantics: the *causal* approach, where concurrency is regarded as causal independence, and the *spatial* approach, where concurrency is given by the distributed nature of a system.

Both these approaches have been applied to the π-calculus by Sangiorgi [San94c, BorSan94]. In the first paper, Sangiorgi uses Boudol et al's formalisation of distributed bisimulation, namely the *location equivalence*. In this setting, the standard bisimulation is refined by allowing the observer to see whether two active components of a distributed system are located in the same site or not. It is not difficult to adapt this to the π-calculus, thus obtaining a location equivalence for it. Sangiorgi shows in [San94b] that one can exploit the naming facilities of the π-calculus to encode the "located agents" as ordinary π-agents, so that two agents are location equivalent iff their encoded versions are bisimilar. In a sense, this is another experiment with the expressive power of the π-calculus, since this means that truly distributed processes can be "implemented" using name passing. Moreover, this provides a proof technique, reducing the proof of location equivalence to that of usual bisimilarity.

In [BorSan94], Boreale and Sangiorgi apply the causal approach to the π-calculus. This involves some subtleties, because a new form of dependency arises in the π-calculus: an action may be dependent on another one just because it is a communication on a private channel which is "opened" by its transmission to another agent. As a matter of fact, this kind of dependency is already taken into account by the standard interleaving semantics. Again, the main result of [BorSan94] is that the causal semantics can be directly expressed within the "ordinary" π-calculus, equipped with the standard bisimulation. The technique is the same as for locations:

the extra structure over processes that is needed to deal with the causal dependencies can be encoded by means of π-calculus agents. The same conclusion can be drawn, that is one may prove causal equivalences using the standard interleaving approach. Moreover, a comparison can be made between the two non-interleaving approaches within the π-calculus.

The paper [MonPis94] by Montanari and Pistore also deals with a "truly concurrent" semantics for the π-calculus. However, the methodology they use is quite different from the one found in previous works and is reminiscent of the definition of non interleaving CCS via Petri nets. Instead of refining the operational semantics of the calculus – for instance by decorating the transitions with informations about localities or causal dependencies –, Montanari and Pistore introduce a *graph rewriting* operational semantics for the π-calculus (it would be interesting to compare this with other "graphical approaches" to the π-calculus, by Milner and Parrow for instance). They note that the induced notion of causality is slightly more liberal than the one in [BorSan94]. As usual, the "graphical" operational semantics serves as a basis for defining various abstract semantics. This is done by varying the notion of observation. Montanari and Pistore consider three different kinds of observations: interleaving, partial ordering and mixed ordering. As expected, the equivalence induced by interleaving observations is the ordinary early observational equivalence, and the mixed ordering equivalence is finer than both the interleaving and partial order equivalences. On the other hand, a surprising fact is that the partial order equivalence is not finer than the interleaving equivalence. The reason is that interleaving observations implicitely assume a centralized name generator, while partial ordering observations do not. Thus, in addition to the usual examples where partial ordering observations are more discriminating, it is possible to give examples where, due to certain symmetries of the state, partial ordering observations are less discriminating than interleaving observations.

The work by Amadio and Prasad [AmaPra94] is also concerned with non-interleaving semantics: it uses the notion of locality, addressing the question of how to reason about located processes in the presence of *node failure*. To this end they extend the π-calculus, allowing both agents and channels to be assigned to locations, which may fail. The behaviour of an agent depends on the status of the locations of the channels on which it may communicate. This is intended to provide a theoretical formalisation of some Facile's implementation features. Amadio and Prasad define a barbed congruence over this calculus, which does not take into account the actual names of the locations (as in the location equivalence), yet is strictly more discriminating than the usual interleaving semantics, due to the observation of the potential node failures. As in the papers [San94c, BorSan94], Amadio and Prasad show that the calculus with localities and failures can be directly encoded into the "standard" π-calculus. This amounts to implementing each location as a π-agent managing the access to the resources depending on that location. This is again an example of the expressive power of the π-calculus. The authors also discuss the proof method induced by their result: as in the afore-mentioned works on non-interleaving semantics, one may reduce the proof of equivalence of agents in the presence of nodes failures to the proof of equivalence of their encoded version.

References

[AmaPra94] R. Amadio and Sanjiva Prasad: *Localities and Failures*, Technical Report ECRC-94-18, 1994.

[MonPis94] U. Montanari and M. Pistore: *Concurrent Semantics for the π-calculus*, July 1994.

[San94c] D. Sangiorgi: *Locality and Non-interleaving Semantics in Calculi for Mobile Processes*, in Proc. International Symposium on Theoretical Aspects of Computer Science (TACS '94), LNCS 789.

[BorSan94] M. Boreale and D. Sangiorgi: *A Fully abstract semantics of causality in the π-calculus*, Tech. Rep. ECS–LFCS–94–297, University of Edinburgh.

5.2.3 Relating calculi

Various calculi for higher order communicating processes have been proposed so far, and an obvious task was to investigate their relationships. As a matter of fact, this comparison should also include sequential models of computation. Some results were obtained in the past few years, notably by Milner and Sangiorgi, who showed that the π-calculus is expressive enough to encode various reduction strategies in the λ-calculus, and to "implement" agent passing in concurrent calculi. Also Walker started the task of studying object based programming using the π-calculus. These works not only demonstrate the universality of name passing, as advocated by Milner, but also could serve as a formal basis for reasoning about higher order concurrency and other computational paradigms. We have seen in the previous section that reasoning in various settings may often amount to using standard bisimulation techniques in the π-calculus. Obviously translating some language into the π-calculus provides a similar proof technique. The papers presented below do not all follow this "reductionist" approach, however.

The paper [Wal94b] by Walker is a development of his previous work on translating parallel object-oriented programming languages into the π-calculus – or more precisely into an extension of the π-calculus, including some features of value-passing CCS, which provides a good framework for understanding the behaviour of mobile systems. The translation is used as the basis for an investigation of programs transformations. In particular, Walker proves some new results concerning replicators, which he then uses to show the soundness of certain specific transformations.

A similar study is undertaken by Amadio in [Ama94c] (see also [AmaLetTho94]) regarding the language Facile, with the aims of providing a basis for the definition of abstract machines, the transformations of programs, and the development of modal specification languages. As a preliminary step, Amadio concentrates on Core Facile, which is, roughly speaking, a simply typed call-by-value λ-calculus, enriched with parallel composition, channel generation and input-output synchronous communication. In particular, a channel is any expression of channel type. It has to be evaluated before a communication can take place on this channel. The language also includes a control operator (call/cc). Amadio shows how to define some Facile features in the core language. Then he introduces a simplified version of the language, the *asynchronous* version (where, as in the π-calculus, the output construct is restricted to be non-blocking) and he shows that the synchronous Core Facile may be adequately translated into the asynchronous one. Moreover, this language can be further simplified, using a continuation passing transformation to eliminate the control operator. This reduces the problem of defining an abstract machine for Core Facile. Indeed Amadio defines such a machine for the restricted language. This is a "chemical solution" of call-by-value environment machines, enriched with a mechanism for the dynamic generation of channel names. Finally Amadio shows that the restricted language, asynchronous Core Facile without control operator, may be translated into the (asynchronous) π-calculus (see also [AmaLetTho94]). Throughout his work, Amadio uses the barbed congruence of Milner and Sangiorgi as a criterion for the adequacy of the various translations.

The works reported on in the rest of this section present "direct" studies of process calculi, not referring to the π-calculus or any other higher-order calculus. In [FerMon94], Ferrari and Montanari propose a new paradigm, *additive concurrency*, where typed behaviours of concurrent programs are described in terms of matrix calculi. In this setting, the operation of matrix product is the fundamental primitive operation to compose typed behaviours. The authors use "dynamic matrices", which have extensible dimensions (rows and columns), allowing the

product to be always defined. In fact, dynamic matrices, beyond a principal type, have any type larger than it. Ferrari and Montanari show that dynamic matrices can be used to describe behaviours of concurrent programs which can change dynamically their amount of parallelism, i.e. can fork and join. They prove that dynamic matrices are exactly the morphisms of a suitable category with biproducts (categories where product and coproduct coincide). The main consequence of this result is an axiomatic characterisation of dynamic matrices. The usefulness of the proposed paradigm is demonstrated by showing that some interesting properties of process calculi can be analysed in terms of dynamic matrices.

In the paper [BerBetPon94a] a typed combinatory process algebra is introduced, that combines process algebra in the ACP framework with types and the classical combinators I, K, B, C. These serve to eliminate recursion variables altogether, so that computations can be done in an entirely equational way. As an extended example the simple Alternating Bit Protocol is verified using only first-order equational logic. Another endeavour has been to eliminate recursion variables in process specifications in favour of iteration using variants of Kleene star. This is done in [BerBetPon94b] where axiomatisations have been given and different versions compared. The paper [BerKli94] reports on a more practical work, studying the Toolbus, a component interconnection architecture, using the frameworks of Process Algebra, and the specification formalisms ASF and SDF.

References

[Ama94c] R. Amadio: *Translating Core Facile*, Technical Report ECRC-94-3, 1994.

[AmaLetTho94] R. Amadio, L. Leth and B. Thomsen: *From Concurrent Functional Programs to Mobile Processes*, submitted for publication.

[BerBetPon94a] J.A. Bergstra, I. Bethke and A. Ponse: *Process Algebra with Combinators*, in Proc. 7th Workshop CSL '93 (Computer Science Logic), Swansea 1993, LNCS 832, 36–65.

[BerBetPon94b] J.A. Bergstra, I. Bethke and A. Ponse: *Process algebra with iteration and nesting*, Technical Report P9314b, Programming Research Group, University of Amsterdam, 1994.

[BerKli94] J.A. Bergstra and P. Klint: *The Toolbus: a Component Interconnection Architecture*, Technical Report P9408, March 1994, Programming Research Group, University of Amsterdam.

[FerMon94] G. Ferrari and U. Montanari: *Typed Additive Concurrency*, August 1994, submitted for publication.

[Wal94b] D. Walker: *Algebraic proofs of properties of objects*, in Proc. 5th European Symposium on Programming ESOP'94, Edinburgh, April 1994, D. Sannella (ed.), Springer-Verlag LNCS vol. 788, 501–516.

5.2.4 Interrelations between sites and to other areas. Future work

We have already mentioned the relation between ECRC (B. Thomsen, L. Leth and S. Prasad) and ENS (R. Amadio, who visited ECRC for six month). This collaboration resulted in joint work on the mathematical foundations of the programming language Facile, developed and implemented at ECRC. This work is also relevant to the area of Foundational Models and Abstract Machines. A cooperation between ECRC, ENS and SICS also started this year,

aiming at developing logics for reasoning about concurrent functional programs. This topic is relevant to the areas of Logics and Calculi, and related to the work done in the area of Programming Languages.

The work by S. Prasad, studying a formulae-as-types/programs-as-proofs interpretation of a fragment of Girard's Unified Logic (LU), though primarily concerned with questions of logic, is clearly relevant to the Calculi area too. In his work, S. Prasad develop a typed calculus of "communicating applicative processes". Moreover, his joint work with R. Amadio on localities and failures shows a connection with previous work done at Sophia and Pisa. This also holds regarding the work of D. Sangiorgi on non-interleaving semantics. More generally, one may note from the report above that techniques developed within the project, concerning bisimulations (e.g. barbed bisimulations) and translations in the π-calculus (initiated by Milner and Sangiorgi), are widely used by most members.

The work on the λ-calculus with multiplicities is still a topic of active collaboration between Sophia (G. Boudol and C. Laneve) and ENS (C. Lavatelli). Finally some works reported in the area of Foundational Models, on the CHAM, and most notably on Action Calculi by R. Milner are also strongly connected with what is done in the Calculi area.

It is expected that the work on semantics foundations of distributed programming, and particularly of the Facile language, will be further developed. A collaboration between ECRC and Pisa has been initiated on this subject, and the cooperation between ECRC and ENS will be maintained. An emerging topic is the question of determinacy, or more accurately of confluence: one would like to find some characterisations of subsets of various calculi where the reduction relation does not involve conflicts. This is of clear importance for the programming languages, since confluent programs should be easier to understand, to verify and to maintain. Although no contribution on the λ-calculus with multiplicities was delivered this year, work is in progress on this subject. We plan to characterise the discriminating power of this calculus, and to compare it with the one of the π-calculus. This could also be used for studying translations of other calculi into the π-calculus.

5.3 Logics for Concurrency and λ-calculus

5.3.1 Summary

Here we describe research carried out in the "Logics for Concurrency and λ-calculus" area. Work has been carried out mainly at two sites: the Department of Computing, Imperial College, London and ECRC, Munich. Work done can be divided into the following categories:

- Types for Concurrency (S. Abramsky, S. J. Gay, R. Nagarajan, G. Meredith)
- Logical Methods for Concurrency (S. Prasad, S. J. Gay, R. Nagarajan, D. Pavlović)
- Game Semantics, Linear Logic (F. Lamarche, M. Huth)

Significant advances have been made within the last year in this area — several of the points mentioned as future work in last year's report have been completed successfully.

We have been able to develop type systems for processes, based on principles derived from the work on Interaction Categories, to address issues in synchronous and asynchronous concurrent computation, verification of concurrent systems and mobility. These are non-trivial applications of the Interaction Category paradigm. There has been much work based on Linear Logic, Girard's **LU** and Categorical Logic as a basis for typed frameworks of processes. Game Semantics and Linear Logic have been used to advance the state-of-the-art in the theory of functional computation.

A summary of the work done appears below. A list of reports and publications appears at the end of this document, along with other references.

5.3.2 Work Done

5.3.2.1 A Typed Calculus of Synchronous Processes

As part of the programme of developing applications of Abramsky's interaction categories [Abr94a, Abr93] we have investigated a typed process calculus based on the structure present in a general interaction category. A pillar of the theory of sequential computation is the use of the simply typed λ-calculus as a canonical functional programming language, and the definition of its semantics in terms of cartesian closed categories. In this investigation, we have aimed to extend this theory to concurrency; the main difference being that we have started with the semantic categories rather than the calculus. Since interaction categories have a linear type structure, the standard sequent presentation of classical linear logic also provides input to the design of the calculus. Abramsky [Abr94b] has already described how classical linear logic sequents can be used as process interface specifications, and the type system of our calculus follows his proposal.

Specifically, we define a calculus of synchronous processes, with a syntax based on Abramsky's linear realisability algebras but extended with constructions for prefixing, non-determinism and recursion. The calculus has an operational semantics defined as a labelled transition system, and a categorical semantics defined in terms of a collection of axioms for synchronous interaction categories. One such category is Abramsky's category $\mathcal{SP}roc$ of synchronous processes, but it is intended that the calculus can also be interpreted in other suitably-structured categories. The choice of category affects the semantic interpretation of types; at the simplest level, a type consists of a sort and a safety specification.

The operational semantics allows strong bisimulation to be defined as the natural equivalence on processes; the categorical semantics yields a notion of denotational equality, and we prove that this is sound with respect to strong bisimulation. The existence of types and an

operational semantics raises the question of subject reduction, i.e. preservation of types by transitions. In our calculus, the presence of safety specifications in the categorical semantics, and the corresponding properties of syntactic types, mean that the operational semantics does not preserve types. However, there are two results which we call static and dynamic subject reduction. Static subject reduction states that certain aspects of the type of a process (essentially, the number of ports) are preserved by transitions. Dynamic subject reduction states the relationship between the way in which transitions change types, and the syntactic form of the *prefix judgements* which are used to introduce prefixing actions in the calculus.

Another aspect of the established theory of the simply typed λ-calculus is the very strong connection between the calculus and cartesian closed categories, formed by the construction of syntactic categories. The process calculus which we have defined offers the prospect of extending this theory to concurrency as well. Although we have not yet pursued this idea very far for the full calculus, we have established the correspondence for a slightly restricted version of the calculus and a suitably modified notion of interaction category. This work is described briefly in a later section.

`Associated Report: [Gay94]`

5.3.2.2 Types for Asynchronous Deadlock-Freedom

Abramsky's work on interaction categories [Abr94a, Abr93, Abr94a] has established a theory of typed concurrency in which types can encode complex behavioural properties of processes. The laws of typed process combination then become compositional proof rules for these properties. In the most elementary interaction categories such as $\mathcal{SP}roc$ [Abr94a, Abr93] and $\mathcal{ASP}roc$ [Abr94a], only safety properties can be analysed; however, the idea of *specification structures* [Abr94a] allows more complex properties to be systematically added to the types. One such property is *deadlock-freedom*, which we take to mean non-termination. A specification structure for deadlock-freedom of synchronous processes has already been described [Abr94a]; we have now established a similar specification structure for asynchronous deadlock-freedom.

The fundamental operation on processes in an interaction category is restricted composition, but it is easy to see that this operation does not preserve deadlock-freedom. Two processes may each be able to run for ever in isolation, but if forced to communicate they can deadlock each other by being unable to agree on which action should be performed next. Thus a type must impose a greater constraint on a process than just requiring that it never terminates. Following the approach used for synchronous processes, we use the idea of *ready sets* and *ready pairs*. A ready set is a set of actions which a process is prepared to do next; a ready pair is a state (represented as the sequence of actions which have been performed so far) and the ready set corresponding to that state. A notion of *orthogonality* of ready pairs—for two ready pairs to be orthogonal, their ready sets must intersect non-emptily when their traces are the same—allows a condition to be formulated which ensures that composition does not introduce deadlocking behaviours. A type contains a set of ready pairs, and a process satisfies a type only when its ready pairs are all in the set specified by the type.

These types have a linear structure, in which negation comes from orthogonality, and so can be fitted into the general interaction category scheme. In the synchronous case, the ideas described above lead naturally to a category of deadlock-free processes. However, when dealing with asynchronous processes there are a number of complications. The first is the possibility of divergent behaviour. Two processes, when connected together, may be able to communicate forever, but if they never do any actions outside the port on which they are communicating, the result is an infinite sequence of unobservable actions. Under observation equivalence, this is indistinguishable from deadlock. The solution is to introduce a notion of fairness, again by

augmenting the types with additional information. Adding a set of fair infinite behaviours to each type, and making the type constructors combine these sets in appropriate ways, results in a category (which we call $\mathcal{FP}roc$) in which every process behaves fairly between all of its ports, in the sense that an infinite behaviour visits each port infinitely often. The specification structure for deadlock-freedom is then constructed over $\mathcal{FP}roc$. The second problem is more technical: the deadlock-free category does not have a full $*$-autonomous structure, since there is no tensor unit. This is a potentially serious difficulty, as the $*$-autonomous structure of interaction categories is used extensively when working with typed processes. The solution is to use another type, with care, as a tensor unit; a port of this type is simply a place in which time can be observed to pass.

We have applied these ideas to the analysis of two classic concurrency examples: the cyclic scheduler [Mil89] and the dining philosophers [Hoa85]. Both of these examples rely crucially on the ability to construct cyclic process configurations; however, such constructions are not available in the deadlock-free category. In general, it is not possible to deduce deadlock-freedom of cyclic processes just by considering the ready pairs of their components. The method of working is to set up types which enable components to be assembled up to the point where a cyclic connection is needed. Then, a traditional verification argument is used to show that the cycle can be completed without introducing deadlocks. Once this has been done, the result is a typed process which can then be connected to other typed processes without introducing deadlocks. The extra verification step only has to be done once: after that, the types take over again. It is to be expected that this methodology will also apply to the use of types which capture different properties.

We conclude that Abramsky's interaction category paradigm, via the concept of specification structures, can effectively extend the use of types to cover interesting properties of concurrent processes.

`Associated Report: [Gay94]`

5.3.2.3 An Internal Language for Interaction Categories

There is a well-established and elegant body of theory connecting the simply typed λ-calculus and cartesian closed categories. A theory over the simply typed λ-calculus can be soundly modelled in a cartesian closed category; furthermore, any λ-theory has a *classifying category* (the smallest category in which it can be modelled), characterised by a universal property, and this can be constructed from the syntax of the theory. Conversely, given a cartesian closed category, a λ-theory can be extracted from it in such a way that the classifying category of this theory is equivalent to the original category. This theory goes by the name of *categorical logic*, and because of the close correspondence between syntax and semantics, the λ-calculus is said to be an *internal language* for cartesian closed categories. Similar correspondences exist for a number of kinds of λ-theory, ranging from the simply typed case described above [LamSco86, Cro94] to the case of higher-order polymorphism [Cro94], and also for other theories such as linear λ-calculi [MacRomAbr93].

In this paper we establish a similar theory for a variety of Abramsky's interaction categories [Abr94a, Abr93, Abr94a] and a suitable language. We take an interaction category to be a $*$-autonomous category with a $*$-autonomous endofunctor (denoted by $\circ$) such that $\circ$ has the *unique fixed point property* [Abr94a]. The language we use is essentially a fragment of the typed process calculus studied by Gay and Nagarajan [Gay94], without prefixing constructions. In contrast to that calculus, our language does not make any commitment to synchronous or asynchronous interaction categories. We define the notion of *process theory* and give a model of a process theory in an interaction category. We then show how to construct the classifying

category of a process theory, by taking objects to be types and morphisms to be equivalence classes of typed terms under provable equality. We also work in the other direction, extracting a process theory from an interaction category and proving the appropriate correspondence theorem.

`Associated Report: [CroGayNag94]`

5.3.2.4 Categorical logic of concurrency and interaction

This work tries to explain in logical terms the basic operations arising from concurrency. A better understanding of the unifying setting of Interaction Categories [AbrGayNag94a], should result.

The starting point was a logical interpretation of Winskel and Nielsen's account [WinNie] of the basic models of concurrency. Upon the obtained logical structures, one can build a calculus of relations which yields Abramsky's interaction category of synchronous processes. This has been worked out in [Pav94].

Furthermore, a dual setting yields the category of asynchronous processes. In the course of the above logical analysis, we were led to some technical results about (bi)simulations. We argue for a canonical representation of bisimilarity classes (both strong and weak) by irredundant trees. The bisimilarity classes of simulations between them boil down to tree morphisms. These results will be described in forthcoming papers.

`Associated Report: [Pav94]`

5.3.2.5 Toward a Model of the π-calculus

Following a suggestion by Abramsky, we apply the techniques developed in [PitSta93] [O'HeTen92] to handle local variable declaration to develop a model of the π-calculus that accounts for restriction. The main idea is that expressions in the π-calculus denote natural transformations between functors from the category of natural numbers and injective maps to various toposes. There is a monad on the associated functor category which captures the notion of acquiring a fresh name. Most of the work is focused on the category of non-well-founded sets as a suitable target. While there is still work to be done, it appears that there is a very tight correspondence between semantic equality and operational equality with respect to the commitment style operational semantics described in [Mil91], and the associated notion of early bisimulation for a replication free fragment of the calculus. Initial work has been done to investigate whether replication can be modelled by replacing the target category with a suitable category of domains.

The approach differs significantly in form from Hennessy's acceptance tree model and has none of the problems that model has with respect to restriction [Hen]. Additionally, the approach is relatively generic in the sense that it is parametric in the target category.

Following [O'HeTen92], we use a possible-world semantics to parameterise interpretations to take account of local aspects of meanings. The possible worlds are represented as natural numbers. Then for each world we associate a computational domain to each phrase type. In the π-calculus there are only two phrase types: names and processes. For purposes of discussion both types can be given meanings as (non-well-founded) sets. These associations can be extended to functors.

More formally, then, the meaning of the phrase type of names is the functor $\mathcal{E}$, defined by

$$\mathcal{E}(n) = [\mathbf{N} \to \mathbf{N}] \tag{5.1}$$

on objects and

$$\mathcal{E}(f)(g) = f \circ ()(g) \tag{5.2}$$

where $f : m \to n$ is an injective map and $g : \mathbf{N} \to \mathbf{N}$.
The meaning of the phrase type process can be given either as

$$\mathcal{ST}(n) = \wp(\wp(n) \times \mathcal{ST}(n)) \tag{5.3}$$

or as

$$\mathcal{ST}(n) = \wp(\mathcal{ST}(w) + (w \times \mathcal{ST}(w)) + [w \to \mathcal{ST}(w)]) \tag{5.4}$$

The former allows objects that are not denotable by the syntax, but makes connections with PIC [Mil93]. The latter makes a cleaner semantics.

Normally, the meaning of a term will be a natural transformation between these two functors. But, to take account of the restriction operator and its ability to generate fresh names we follow [PitSta93] and utilise a monad over the functor category, defined by

$$\mathbf{T}(\mathcal{F})(n) = \amalg_{w \in \mathbf{N}} \mathcal{F}(n + w). \tag{5.5}$$

Utilising the first definition of the $\mathcal{ST}$ functor, we can produce meanings of the form

$$\lceil x.\, A \rceil(w)(e) = \{(\{e(x)\}, \lceil A \rceil(w)(e))\} \tag{5.6}$$

$$\lceil (\nu x) A \rceil(w)(e) = \lceil A \rceil(w+1)((e \mid x \mapsto w)) \tag{5.7}$$

$$\lceil (\lambda x) F \rceil(w)(e) = \{(\{n\}, \lceil F \rceil(w+n)((e \mid x \mapsto n))) \mid n \in \mathbf{N}\} \tag{5.8}$$

$$\lceil [x] C \rceil(w)(w) = \{(\{e(x)\}, \lceil C \rceil(w)(e))\} \tag{5.9}$$

$$\lceil M + N \rceil(w)(e) = \lceil M \rceil(w)(e) \cup \lceil N \rceil we \tag{5.10}$$

$$\lceil P \mid Q \rceil(w)(e) = \lceil P \rceil(w)(e) \| \lceil Q \rceil(w)(e) \tag{5.11}$$

where

$$\begin{array}{rcl} p \| q & = & \{(\{\}, p' \| q') \mid \exists m, n.(\{m\}, (\{n\}, p')) \in p, (\{-m\}, (\{n\}, q')) \in q\} \\ & & \cup \{(r, (s, p' \| q)) \mid \exists r, s.(r, (s, p')) \in p\} \\ & & \cup \{(r, (s, p \| q')) \mid \exists r, s.(r, (s, q')) \in q\} \end{array}$$

A paper detailing the results is under construction.

`Associated Report: [Mer94b]`

5.3.2.6 The Positive Intuitionistic Fragment of LU

Girard's Unified Logic **LU** provides a single sequent calculus common to classical, intuitionistic and linear logic, in which these logics appear as fragments, *i.e.*, as particular classes of formulae and sequents. We present a fragment of **LU** based on the idea of staying within the class of formulae of *positive polarity*. The positive intuitionistic fragment introduces two new *chimeric connectives*: positive implication $P \to Q$ encoded as $!(P \multimap Q)$ and positive universal quantification $\nabla x P$ encoded as $!(\bigwedge x P)$, when P, Q are positive formulae.

Our main results are showing that the cut elimination result for **LU** may be extended to cover the new chimerae, and that if a positive intuitionistic sequent is provable in **LU**, it is provable within the fragment.

We also provide an informal discussion on the computational content of a subfragment of the positive intuitionistic fragment of **LU**, by providing a "formulae-as-types" view that allows us to see the correspondence between the subfragment and a simply-typed λ-calculus.

`Associated Report: [Pra94a]`

5.3.2.7 Cut Elimination for LU with Mingle

We extend Girard's Unified Logic with a rule called (Mingle), which is the counterpart of the (Mix) rule of linear logic. (Mingle) allows us to combine two independent deductions into a single deduction. We show that **LU**+(Mingle) also enjoys cut elimination under minimal hypotheses. The main connection is to Vauzeilles's cut elimination result, which we extend. This is a technical note, to check that one can use a "mix"-like rule to represent concurrency (as juxtaposition) in **LU**.

`Associated Report: [Pra94b]`

5.3.2.8 Towards a Formulae-as-Types View of Communicating Applicative Programs

We study a simple typed calculus of concurrent applicative processes from a "formulae as types" perspective. The calculus extends a simply-typed λ-calculus with constructs for concurrency and inter-process communication, and lies at the core of a class of typed, concurrent applicative languages, *e.g.*, Concurrent ML and Facile.

We present a formal system within the framework of Girard's *unified logic* **LU**, a sequent calculus common to classical, intuitionistic and linear logic. Our formal system is a small fragment of **LU** extended with a new chimera for positive intuitionistic implication, an additional assumption $\bot = \mathbf{1}$ and with the rule (Mingle), which corresponds to the (Mix) rule of linear logic. We show that the formal system may be considered a closed *quasi-intuitionistic* fragment of **LU**+(Mingle). This result relies on the fact that **LU**+(Mingle) enjoys cut elimination under minimal hypotheses, which we show by adapting Vauzeilles's proof for cut elimination in **LU**.

A term assignment shows how programs in the calculus may be thought of as deductions of sequents in the formal system. The novelty of the treatment is in relating the sending and receiving of typed values to the two symmetric "classical" cut rules of **LU**. Not all deductions in the formal system correspond to programs, but we show that by commuting inference rules, every deduction can be converted to one that has a term assignment. Of particular interest are the transformations on terms induced by commuting inference rules. These seem closely related to certain transformations of terms, such as "structural equivalences" and "reversible ionisations", employed in the *chemical abstract machines*.

We then show that the operational semantics of the calculus, presented both as a Plotkin-style (unlabelled) reduction as well as a Milner-Plotkin style labelled transition system, preserve types assigned by the static semantics (subject reduction). In fact, we show that the operational semantics may be considered as a particular cut-elimination strategy, with communication corresponding to the elimination of the special kinds of cuts.

`Associated Report: [Pra94b]`

5.3.2.9 Proof nets for intuitionistic linear logic

Most of the year was spent in trying to solve the proof net problem for intuitionistic linear logic. After many computations we had come to the conclusion that the logic that was naturally expressed by games semantics was intuitionistic linear logic, with the connectives $\otimes, \multimap, \&$ and $!$. It seemed to us that this was the fragment of LL that games semantics could teach us something about, and that the interpretation of any proper extension of that fragment in games semantics was an ad hoc contrivance, not true to the spirit of the semantics. We were aware that the Danos-Régnier system of polarities allows for a presentation of intuitionistic linear logic that uses one-sided sequents, just like classical LL, and that its interpretation of polarities in the realm of games is "who plays first: Player or Opponent". So, using games as a guide, we tried to construct a theory of proof objects for the logic described above. We first got a result for the multiplicative-additive fragment (presented at the CONFER Workshop in Edinburgh in May 93), but the real breakthrough came in January 94, when we succeeded in adding the exponentials. The paper is [Lam94a]. Therein it is shown how to translate every proof of a sequent in the fragment above into a *proof net*, a special kind of oriented graph which generalises those presented in [Gir87]. The point is that this translation is reversible: there is a correctness criterion on proof nets which singles out those that have been constructed from proofs. The proof of this correctness theorem differs radically from all those that have been presented before in Linear Logic. The reason is that the logic is intuitionistic, and this fact is used in an essential way. In particular the Danos-Régnier polarities are an important invariant, and there is a new notion of *path*, specially tailored for an intuitionistic context, which replaces the *trips* of [Gir87].

`Associated Report: [Lam94a]`

5.3.2.10 A denotational semantics based on Chu spaces

After this syntactical result was written up, we decided to see if it could be applied to semantics; we now had a powerful tool that made computations much easier than before. In the original denotational semantics of LL, that is, coherent domains, there is a natural notion of *web*; given a coherent domain X its *web* $|X|$ is a set such that every element of X is a subset of X. An important property of webs is how they interact with the connectives; in particular we have that

$$\begin{array}{rclcl} |X \otimes Y| & = & |X| \times |Y| & = & |X \parr Y|\,, \\ |X \,\&\, Y| & = & |X| + |Y| & = & |X + Y|\,, \end{array}$$

which is indeed how the additives and the multiplicatives got their names. Recently new models of linear logic have appeared that are also equipped with webs, these webs acting exactly as above with respect to the additives and multiplicatives; for example Plotkin and Winskel's bidomains [PloWin93] and Ehrhard's hypercoherences [Erh93]. It became natural to ask if the notion of web were not directly definable from the syntax, and that maybe direct observation of syntax would yield better models. Here it was obvious that syntax was full linear logic where the multiplicative units are identified (thus we have the Mix rule), and also the additive units are identical. After many computations using suitably modified proof nets, we came up with an abstract definition of the web of a formula X, based on the interaction of the proofs of X with those of $X^\perp$ (because the terminal object is also the initial one, in this logic a formula has both proofs and counterproofs). This web is in general a poset. This allowed us to construct a new model of linear logic, based on Chu spaces [Bar79, Pra93], which has many interesting properties. One of these interesting properties is that the Kleisli category for $!$ (in other words, the intuitionistic category) is a well known one in denotational semantics: it is the category of bifinite domains and continuous functions. This came as a surprise, since experience had

taught us that there seemed to be no model of full linear logic compatible with Scott continuity, outside of the limited world of complete lattices.

`Associated Report: [Lam94b]`

5.3.2.11 Interaction Orders as Games

We enriched the $*$-autonomous category SUP of complete lattices and maps preserving all suprema with the concept of *approximation* by specifying the greatest $*$-autonomous subcategory LFS of linked bicontinuous lattices. In the algebraic case, we arrived at aLFS. Its objects are those complete lattices A such that $\mathrm{id}_A = \sqcup\!\uparrow\mathcal{D}$ for some directed set $\mathcal{D}$ of *idempotent deflations* (idempotent functions d with finite image and $d \leq \mathrm{id}_A$) in $A \multimap A$; the space $A \multimap A$ is the complete lattice of all functions $f\colon A \to A$ preserving all suprema, ordered pointwise. We came up with an order-theoretic description of these lattices via forbidden substructures à la Plotkin. Further, we showed that the distributive lattices in LFS are exactly the completely distributive lattices, which gives rise to two $*$-autonomous categories CD and aCD. In this paper we attempt to reformulate aLFS (and aCD) as a category of *information systems* or *games*. Recall that the information systems for Scott-domains are asymmetrical notions. A process (= function) is gaining information using an inference mechanism and the environment has no active rôle in all of this. The symmetry inherent in the definition of objects in aLFS strongly suggests that process and environment should be explicitly mentioned and interchangeable in such a formal setup. Therefore, it is suitable to consider every complete lattice as a two person game where the poset of sup-, respectively inf-irreducible, elements is the set of moves. Algebraic linear FS-lattices are those bialgebraic lattices where each component of the corresponding game forest is finite. We represent all the type constructors of the $*$-autonomous category of algebraic linear FS-lattices as compound games. The biproduct game models internal non-determinism. The tensor product reflects the *switching conditions* in the work of Abramsky et al. but for certain non-distributive lattices A or B when opponent, unlike player, has even more moves to choose from.

`Associated Report: [Hut94b]`

5.3.3 Interrelations between sites and to other areas

We have already mentioned the work of Sanjiva Prasad, ECRC as basically in the same realm as our research on types and logics for concurrency. Our investigations on typed concurrent programming uses some of the techniques used in other areas such as Calculi and Foundational Models; it is expected that our work can be successfully adapted to those settings as well. The work on a model of the π-calculus is directly relevant to the Calculi area. Research on Linear Logic and more pertinently, the Geometry of Implementation, is clearly connected with work on Foundational Models and Abstract Machines, especially the work of the INRIA Rocquencourt group. This work can be used as a basis for implementation issues in the Programming Languages area.

5.3.4 Future Work

In the logics and types for concurrency area, future plans include:

- A fully-developed typed calculus for asynchronous processes similar to the one worked out for synchronous processes.

- A typed calculus for deadlock-free processes.
- Addressing other verification issues, such as fairness and liveness.
- Categorical Logic of true concurrency: noninterleaving leads to some interesting extensions of regular logic, bordering with the realm of Hopf algebras.

In the logics for functional computation topic, we plan to:

- Complete the work on "executable nets", which are a version of proof nets that allows for parallel execution; they give a procedure for parallel execution of the lambda-calculus which is different from the ones that have been proposed so far.
- Find the best method of doing polymorphism with dialectics; find more about the relationship between Chu spaces and denotational semantics.
- Answer the question as to whether Linear Scott-domains are the greatest symmetric monoidal closed category of Scott-domains where the linear function space is that of maps preserving all existing suprema.

5.3.5 List of Reports

Imperial College, London

[AbrGayNag94a] S. Abramsky, S. J. Gay, and R. Nagarajan. Interaction Categories and Foundations of Typed Concurrent Programming. *Deductive Program Design: Proceedings of the Marktoberdorf International Summer School*, 1994. NATO ASI Series F: Computer and Systems Sciences, Springer Verlag. To appear.

[AbrGayNag94b] S. Abramsky, S. J. Gay, and R. Nagarajan. A Specification Structure for Deadlock-Free Processes. Abstract of talk given at the Tenth Workshop on Mathematical Foundations of Programming Semantics, Manhattan, USA.

[AbrGayNag94c] S. Abramsky, S. J. Gay, and R. Nagarajan. Compositional Verification of Deadlock-Freedom. Abstract of talk given at the CONFER Workshop, ECRC, Munich, Germany, April 1994.

[CroGayNag94] R. L. Crole, S. J. Gay, and R. Nagarajan. An Internal Language for Interaction Categories. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

[Gay94] S. J. Gay. *Linear Types for Communicating Processes.* PhD thesis, University of London, 1994. To appear.

[GayNag94a] S. J. Gay, and R. Nagarajan. A Typed Calculus of Synchronous Processes. Abstract of talk given at the CONFER Workshop, ECRC, Munich, Germany, April 1994.

[GayNag94b] S. J. Gay, and R. Nagarajan. A Typed Process Calculus. Abstract of talk given at the Mini Workshop on Semantics and Topology, Tulane University, New Orleans, USA, March 1994.

[Hut94a] M. Huth. Interaction Orders as Games. Abstract of talk given at the CONFER Workshop, ECRC, Munich, Germany, April 1994.

[Hut94b] M. Huth. Interaction Orders as Games. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

[Lam94a] F. Lamarche. Proof Nets for Intuitionistic Linear Logic I: Essential nets. Abstract of talk given at the CONFER Workshop, ECRC, Munich, Germany, April 1994. Report available by anonymous ftp from theory.doc.ic.ac.uk.

[Lam94b] F. Lamarche. Dialectics: a model of linear logic and PCF. Report available by anonymous ftp from theory.doc.ic.ac.uk.

[Lam94c] F. Lamarche Hypercoherences are Chu Spaces. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

[Mer94a] G. Meredith. Actors and Interaction Categories: Preliminary Investigations. Abstract of talk given at the CONFER Workshop, ECRC, Munich, Germany, April 1994.

[Mer94b] G. Meredith. Notes Toward a Model of the π-calculus. Paper in Preparation.

[Pav94] D. Pavlović Categorical logic of concurrency and interaction, I: Synchronous processes. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

ECRC, Munich

[Pra94a] S. Prasad. The Positive Intuitionistic Fragment of **LU**. Technical report ECRC-94-11, 1994.

[Pra94b] S. Prasad. Cut Elimination for **LU** with Mingle. Technical report ECRC-94-10, 1994.

[Pra94b] S. Prasad. Towards a Formulae-as-Types View of Communicating Applicative Programs (Extended Summary). ECRC Technical report ECRC-94-32, 1994.

Other References

[Abr93] S. Abramsky. Interaction Categories (Extended Abstract). In G. L. Burn, Simon J. Gay, and M. D. Ryan, editors, *Theory and Formal Methods 1993: Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods*, pages 57–70. Springer-Verlag Workshops in Computer Science, 1993.

[Abr94a] S. Abramsky. Interaction Categories and communicating sequential processes. In A. W. Roscoe, editor, *A Classical Mind: Essays in Honour of C. A. R. Hoare*. Prentice Hall International, 1994.

[Abr94a] S. Abramsky. Interaction Categories I: Synchronous processes. Paper in preparation, 1994.

[Abr94b] S. Abramsky. Proofs as processes. *Theoretical Computer Science*, 1994. To appear.

[Bar79] M. Barr. $*$-autonomous categories. *Lecture Notes in Mathematics* 752, Springer Verlag, 1979.

[Cro94] R. L. Crole. *Categories for Types.* Cambridge University Press, 1994.

[Erh93] T. Ehrhard. Hypercoherences, a strongly stable model of linear logic. *Math. Struc. in Comp. Sci.*, 3(1993)365-385.

[Gir87] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[O'HeTen92] P. O'Hearn and B. Tennent. Semantics of Local Variables. In M.P. Fourman, P.T. Johnstone, and A.M. Pitts, editors, *Applications of Categories in Computer Science: Proceedings of the LMS Symposium, Durham 1991*, pages 217–238. Cambridge University Press London Mathematical Society Lecture Note Series 177, Cambridge, 1992.

[Hen] M. Hennessy. *A Model of the π-calculus.* Announced on Concurrency mailing list and available via anonymous ftp at University of Sussex site.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[LamSco86] Joachim Lambek and Philip J. Scott. *Introduction to Higher Order Categorical Logic.* Cambridge Studies in Advanced Mathematics Vol. 7. Cambridge University Press, 1986.

[MacRomAbr93] Ian Mackie, Leopoldo Román, and Samson Abramsky. An internal language for autonomous categories. *Journal of Applied Categorical Structures*, 1(3):311–343, 1993.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mil93] R. Milner. Action Structures. Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh. Technical report, number 92-249, 1993.

[Mil91] The Polyadic π-calculus: a Tutorial. Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh. Technical report, number 91-180, 1991.

[PitSta93] A. M. Pitts and I. D. B. Stark. *On the Observable Properties of Higher Order Functions That Dynamically Create Local Names (Preliminary Report).* In *Workshop on State in Programming Languages, Copenhagen*, 1993, pages 31–45. ACM SIGPLAN Proceedings, 1993.

[PloWin93] G. Plotkin and G. Winskel. Bistructures, bidomains and linear logic. *Extended abstract*, 1993.

[Pra93] V. Pratt. The second calculus of binary relations. *Proceedings of MFCS 1993, Gdansk.*

[WinNie] G. Winskel and M. Nielsen. Models for Concurrency. In S. Abramsky and D. Gabbay and T. S. E. Maibaum, editors. *Handbook of Logic in Computer Science*. Oxford University Press. To appear.

5.4 Programming Languages

This section provides a brief summary of the work pursued in the context of the CONFER BRA in the area of programming language design, implementation and experience with applications.

In the second year of the action concrete results have started to emerge. There is now an advanced implementation of Pict and some demonstration programs have been developed.

The semantic foundation of important techniques on compile time optimisations via unfolding of programs has been studied. This work continues previous activities regarding termination properties of unfolding established last year. The latest results are expected to be mature enough to be applied in prototype compilers.

A programme of research – the Geometry of Implementations – aimed at developing efficient implementations of functional programs based on Girard's Linear Logic and the Geometry of Interaction semantics has been initiated. This work is derived from previous CONFER results of Gonthier et. al. on optimal reduction implementations and the work of Danos and Regnier on local and asynchronous β-reduction.

The Calumet demonstrator written in Facile presented at the first CONFER review has been developed into a robust application running on wide area networks. Calumet is now used frequently for presentations at ECRC.

The first release of Facile – the Facile Antigua Release – has been announced on the news net and made available for research and educational purposes. Technology transfer to the ECRC shareholders has also taken place [1].

The report includes short summaries of the following efforts:

- Facile programming language
  ECRC — A. Giacalone, F. Knabe, A. Kramer, T.M. Kuo, L. Leth, S. Prasad, B. Thomsen. Also with contributions of P. Cregut, P.-Y. Chevalier, J.P. Talpin and C. Crampton.
- Calumet
  ECRC — A. Kramer, J.P. Talpin. Also with contributions of K. Ahlers, P. Marchal.
- Pict programming Language
  University of Edinburgh — B. Pierce, D. Turner.
- Correct unfolding of nondeterministic expressions
  SICS — B. Lisper.
- The Geometry of Implementation
  Imperial College – Ian Mackie.

5.4.1 The Facile programming language

5.4.1.1 Summary

Facile is a higher-order, functional/concurrent programming language that supports the implementation of distributed applications.

Facile is conceived to support the development of systems exhibiting a high degree of mobility, that is systems that may evolve dynamically in terms of structure, communication and computation capabilities. Both processes and communication channels may be dynamically created and are treated as first-class values in the same way as functions. In particular, they can be

[1] It should be noted that the Facile project was already in progress before the beginning of CONFER and also that the dimensions and scope of the project are wider than what can be considered as strictly relevant to the CONFER BRA.

communicated over channels, also between processes executing at different physical locations. The notion of process in Facile is quite powerful, since each process has its own environment and runs a full ML program. However, its implementation is very light-weight, so that large numbers of processes can potentially be executed concurrently. Facile is well-suited for running on loosely connected, physically distributed systems with distributed memory. It is possible to execute Facile programs on both local area networks (LANs) and wide area networks (WANs).

In the 2nd Year of CONFER the group at ECRC has completed a first release of the Facile programming language – the Facile Antigua Release. The Facile Antigua Release is an industrial strength implementation of a distributed higher order concurrent functional programming language based on the theoretical models (such as the λ-calculus, CCS, the π-calculus and CHOCS) related to the CONFER action. The Facile Antigua Release is implemented by modifying and extending the Standard ML of New Jersey implementation. Facile enriches Standard ML with primitives for distribution, concurrency and communication over typed channels. The additional data types provided in the language include node identifiers, process scripts and communication channels. All of these are first-class values that can be manipulated in an applicative style and, in particular, be communicated. New nodes and channels can be created dynamically and processes executing a given script can be spawned dynamically on a given node. The software is documented through a user guide [TLPKKKG93], as well as numerous papers on its semantics and abstract implementations. The current implementation allows the development of applications that operate on a network of SPARC and Sun-4 workstations running UNIX (SunOS or Solaris). Facile Antigua supports distributed programming on both LANs and WANs.

Since January 1994 the Facile Antigua Release has been in alpha test at sites in Europe and the US. Feedback from the alpha test sites has led to minor debugging and has resulted in a rather stable implementation.

Technology transfer to the ECRC share holder companies ICL, Bull and Siemens took place in July 1994. The software is currently under evaluation in several departments.

In July 1994 the availability of Facile Antigua Release has been publicly announced to the research community via the news network and announcements to all ESPRIT partners via email.

5.4.1.2 Work in progress

Future plans for the development of the Facile programming language include porting the software to other hardware platforms (such as Intel x86 processors). Related to this is work in progress on interoperability between Facile systems on different platforms. We currently use a system called the contract mechanism for connecting systems written in Facile with systems written in different languages. We expect to address questions such as interoperability with systems implemented in different programming languages (such as C, C++, prolog) via industrial standards (possibly the emerging CORBA standard). This may replace the contract mechanism.

References

[TLPKKKG93] Thomsen, B., Leth, L., Prasad, S., Kuo, T.-S., Kramer, A., Knabe, F., Giacalone, A.: "Facile Antigua Release – Programming Guide", Technical report ECRC-93-20, 1993.

5.4.1.3 Future work

We plan to address issues related to mobile computing, in particular running Facile on portable devices and interacting with Facile systems running on stationary networks.

Related to the question of running Facile on portable devices a new direction for applications has just been initiated. The model of higher order mobile processes supported by Facile seems a good candidate for structuring software which has to operate in a mobile environment. We are currently conducting experiments to verify this hypothesis.

5.4.2 Calumet

5.4.2.1 Summary

Calumet is a desktop conferencing tool that supports meetings based on a slide presentation metaphor among users in different physical locations. The distributed part of the system, which handles all communications, has been built entirely with Facile and consists of less than two thousand lines of source code [TLPKKKG93, Tal94b].

The Calumet cooperative application for teleconferencing, demonstrated at the 1st Year review at CWI, has been restructured and rendered more fault tolerant. A new user interface programmed in C++ has been developed. The new user interface can display LaTeX style slides with graphics in TIFF format. The Calumet system is now used frequently at ECRC for internal presentations and it has been demonstrated on wide area networks at AT&T (Murray Hills, New-Jersey) and between ECRC, CWI (Amsterdam) and CMU (Pittsburgh). A user manual for the Calumet system can be found in [TalMarAhl94].

5.4.2.2 Work in progress

It is expected that the Calumet system will be released to the scientific community and that technology transfer to the ECRC share holder companies will take place in the near future.

References

[Tal94a] Jean-Pierre Talpin. “The Calumet Experiment - Part I: An Implementation of Group-Communication Protocols in Facile”, Technical report ECRC-94-4, 1994.

[Tal94b] Jean-Pierre Talpin: “The Calumet Experiment in Facile - A Model for Group Communication and Interaction Control in Cooperative Applications”, Technical report ECRC-94-26, 1994.

[TalMarAhl94] Talpin, J.-P., Marchal, P., and Ahlers, K.: “Calumet - A Reference Manual”, Technical report ECRC-94-30, 1994.

5.4.3 Pict Programming Language

5.4.3.1 Summary

Pict is an experimental concurrent programming language based on Milner’s pi-calculus. The project began in 1992 with the following goals:

- To test the suitability of the pi-calculus as a foundation for practical programming.
- To investigate natural idioms arising in this context for programming with concurrent objects.

By the end of August 1993, we had achieved a stable design for the core language, a prototype implementation, and some small demonstration programs. Our work this year has proceeded on all of these fronts.

- The largest change to the language has been the addition of a sophisticated static type system based on a higher-order polymorphic lambda-calculus with subtyping [PieTur92] with channel types based on earlier theoretical work by Pierce and Sangiorgi [PieSan92].

- This type system has been extended with a flexible notion of *extensible record types*, which are heavily used for constructing and manipulating objects; their theoretical study is underway.

- The current Pict implementation also provides a powerful partial type inference algorithm that allows most type information to be omitted from Pict programs; formal underpinnings for this algorithm are being investigated by Dilip Sequeira as part of his thesis work (co-supervised by Pierce).

- The old byte-code interpreter has been replaced by a back-end that compiles Pict programs directly to C. We can now run sizeable interactive graphics applications written entirely in Pict with acceptable performance.

- Our largest programming project in Pict has been a graphical widget toolkit. Our runtime system provides access to the low-level Xlib facilities; on top of this, we have designed a concurrent, object-based library providing a variety of useful widgets (text, buttons, horizontal and vertical stacks of widgets, etc.) maintaining a fairly sophisticated internal protocol for redrawing, resizing, and event handling.

- Several demonstration programs have been written on top of this toolkit, including a "minesweeper" game, a graphical anthill simulation, and a color bitmap editor.

We have been concentrating on design and implementation this year rather than writing, but a few papers have already appeared. Pierce and Sangiorgi finished and submitted a journal version of [PieSan92]; Steffen and Pierce made a careful theoretical study of type checking algorithms for higher-order subtyping [StePie94].

Pierce presented Pict in a week-long tutorial in Erlangen, Germany in September 1993, and in a postgraduate course at the LFCS, Edinburgh, in Winter, 1994. A number of research seminars on Pict and related topics were given at various sites.

5.4.3.2 Future work

Our principal goals for the coming year are:

- Polishing the implementation and bringing the documentation and programming tutorial up to date for a first public release of the system.

- Writing one or more retrospective papers concerning the language design, implementation, and theoretical underpinnings.

- Continuing the investigation of programming styles based on concurrent objects via one or more new major applications, perhaps involving remote communication over a network.

References

[PieSan92] Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. In *1993 IEEE Symposium on Logic in Computer Science*, June 1993.

[PieTur92] Benjamin C. Pierce and David N. Turner. Simple type-theoretic foundations for object-oriented programming. *Journal of Functional Programming*, 4(2):207–247, April 1994. A preliminary version appeared in Principles of Programming Languages, 1993, and as University of Edinburgh technical report ECS-LFCS-92-225, under the title "Object-Oriented Programming Without Recursive Types".

[StePie94] Martin Steffen and Benjamin Pierce. Higher-order subtyping. In *IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET)*, June 1994. An earlier version appeared as University of Edinburgh technical report ECS-LFCS-94-280 and Universität Erlangen-Nürnberg Interner Bericht IMMD7-01/94, January 1994.

5.4.4 Correct unfolding of nondeterministic expressions

5.4.4.1 Summary

This work is related to the work on various reduction systems in the area "Foundational models and abstract machines", but its direction towards practical issues like optimising program transformations motivates its placement in "Programming languages". It continues the previous activities regarding termination properties of unfolding.

Symbolic unfolding and compile-time execution of programs are important techniques for improving the performance of programs. They are best understood for pure functional languages but are also being employed for sequential programs with destructive assignments. When it comes to concurrent languages, however, the techniques are hardly being used, even though their benefits certainly would be desirable also in this context. A major reason is that the nondeterminism of concurrent language constructs can cause the unfolded program to have a different semantics than the original one. Basically, executing a nondeterministic construct at compile-time means that a possible outcome is discarded and will never occur at runtime. This change in semantics can be acceptable, if the nondeterminism simply specifies a number of acceptable outcomes. But very often nondeterminism is used to model an unpredictable environment in which the program executes (like, say, a communication channel where several processes can write). Then, such a change in semantics violates the description of the environment.

For pure functional languages, correctness of symbolic execution follows more or less directly from confluence. Thus, one may think that it is semantically correct to restrict symbolic execution of nondeterministic programs to a purely functional subset. But this is sometimes not enough. Consider, for instance, a simple language with nondeterministic choice "+" and "parallel if", defined by the following TRS: $x + y \to x$, $x + y \to y$, $\mathit{if}(x, y, y) \to y$. Here, the last rule taken by itself defines a confluent sub-TRS (i.e. "purely functional subset"): yet, symbolic execution of the term $\mathit{if}(x, y + z, y + z)$ w.r.t. this rule yields $y + z$ and in the process the normal forms $\mathit{if}(x, y, z)$ and $\mathit{if}(x, z, y)$ are lost. A conclusion is that more refined criteria are needed to ensure correctness of symbolic execution w.r.t. some sublanguage. Some steps towards such criteria are reviewed here.

5.4.4.2 Work in progress

Computations can be modeled by reduction sequences in abstract reduction systems. If the mapping s from an ARS $\langle A, \rightarrow\rangle$ to a cpo $\langle C, \sqsubseteq\rangle$ is monotone w.r.t. "$\rightarrow$" and "$\sqsubseteq$", then any reduction sequence $a_0 \rightarrow \cdots \rightarrow a_i \rightarrow \cdots$ yields a l.u.b. $\bigsqcup_{i=0}^{\infty} s(a_i)$ in C. As "normal form semantics" for $a \in A$ we can then take the minimal set $S(a) \subseteq C$ such that (1) any s in $S(a)$ is a l.u.b. resulting from some reduction sequence starting in a, and (2) $s, s' \in S(a) \Longrightarrow s \not\sqsubseteq s' \wedge s' \not\sqsubseteq s$. One can now prove the following:

Theorem 5.4.1 *if $a \rightarrow^* a'$, and if for any reduction sequence $\{a_i\}_{i=0}^{\infty}$ starting in a there is a reduction sequence $\{a'_i\}_{i=0}^{\infty}$ starting in a' such that $\bigsqcup_{i=0}^{\infty} s(a_i) \sqsubseteq \bigsqcup_{i=0}^{\infty} s(a'_i)$, then $S(a') = S(a)$.*

If $a \rightarrow^* a'$ is seen as a symbolic computation of a into a', then this result gives an abstract criterion that ensures that the semantics is preserved.

Of particular interest are computations over terms since these can be seen directly as symbolic executions of programs. As semantic domain can be taken the cpo of (finite and infinite) "partial" terms t, where possibly $t/p = \bot$ for some positions p, with the obvious partial ordering. For terms $a \in A$, we define $s(a)/p = a/p$ iff $a'/p = a/p$ for all a' such that $a \rightarrow^* a'$, and $s(a)/p = \bot$ otherwise. In some special cases, Th. 5.4.1 above can now be directly applied:

- If $\rightarrow$ is strongly normalising, then any element in $S(a)$ will be a finite, total term. If $\rightarrow$ can be factored as $\rightarrow_D \cup \rightarrow_N$ where $\rightarrow_D$ is confluent and $\rightarrow_D$, $\rightarrow_N$ commute, then Th. 5.4.1 is applicable whenever $a \rightarrow_D^* a'$. A case where $\rightarrow_D$ and $\rightarrow_N$ commute is when they are given by TRS:es D and N, respectively, which are orthogonal to each other (left-linear and mutually non-overlapping). (In the counterexample above, the rule $\mathit{if}(x, y, y) \rightarrow y$ is not left-linear, which destroys commutation.)
- This result can be extended to the case when $\rightarrow$ is ω-*converging* in the sense of Dershowitz, Kaplan and Plaisted [DerKapPla91]. Then, all elements in $S(a)$ are total (but possibly infinite) terms. If $\rightarrow_D$, $\rightarrow_N$ "semi-ω-commute" (cf. semi-ω-confluence in [DerKapPla91]), then the result again holds whenever $a \rightarrow_D^* a'$. Semi-ω-commutation follows when $\rightarrow_D$ and $\rightarrow_N$ are given by TRS:es which are orthogonal to each other.

5.4.4.3 Extensions and future work

It should be investigated in more detail to what extent Th. 5.4.1 can be applied to other reduction systems than such given by plain TRS:es. There are several reasons for this: for instance, realistic evaluation strategies for nondeterministic programs are not adequately described by such. Neither can constructs for concurrency be described in a pure TRS setting. Here, interaction with research in the other areas of the project could be helpful. Once it is understood how Th. 5.4.1 applies to concurrent languages, experiments should be undertaken with some real concurrent language.

[DerKapPla91] Nachum Dershowitz and Stéphane Kaplan and David A. Plaisted: *Rewrite, Rewrite, Rewrite, Rewrite, Rewrite,...*, Theoret. Comput. Sci vol 83, no 1, pp. 71–96, 1991.

5.4.5 The Geometry of Implementation

5.4.5.1 Summary

The Geometry of Implementation [Mac94] is a programme of research aimed at developing efficient, correct implementation of functional programs based on a foundation of Girard's Linear Logic and the Geometry of Interaction semantics.

There are a number of established implementation techniques for functional programming languages. Broadly speaking these are based on graph rewriting; stack manipulation, for example the SECD machine; and buffered data-flow. The first step in our program is the *geometry of interaction machine*—a completely new implementation technique for a simple functional programming language (PCF) based on sequential data-flow, without buffering. The significant features of this work include:

- Sound semantic foundation arising from linear logic and the geometry of interaction.
- A direct compilation of functional programs into assembly language, thus giving a computational interpretation of Girard's geometry of interaction.
- Finally, it requires only a very small run-time system in which functional programs can be executed.

Although incredibly simple, this compilation scheme leads to quite an inefficient implementation. Starting from this basic machine, we analyse properties of the semantics which give rise to a sequence of optimisations which again can be very simply implemented.

The first substantial optimisation is the *question and answer* discipline for PCF programs. This leads to reducing the computation by half as the expense of increased space usage.

A second optimisation has arisen using ideas from memoisation from standard functional programming technology.

5.4.5.2 Work in progress

Further work in this programme include the investigation of a notion of optimal reduction for this kind of implementation.

These implementation techniques are reported in [Mac94].

This work is derived from previous CONFER research of Gonthier et al on optimal reduction implementations and Danos and Regnier's work on local and asynchronous β-reduction.

Presentation

'The Geometry of Interaction Machine' was presented at the CONFER workshop held at ECRC Munich in April 1994. The geometry of implementation will be presented at the Second Theory and Formal Methods workshop, to be held in Cambridge, UK in September 1994, and is the core of Ian Mackie's PhD thesis [Mac94].

[Mac94] Ian Mackie *The Geometry of Implementation (Applications of the Geometry of Interaction to language implementation).* PhD thesis, University of London, 1994. To appear.

5.4.6 Interrelations between sites and other areas

Work on the Facile programming language continues to use results produced in the areas of Calculi, Foundational Models and Abstract Machines as well as Logics for the λ-calculus and Concurrency.

A collaboration between ECRC (B. Thomsen, L. Leth and S. Prasad) and ENS (R. Amadio, who visited ECRC for six months) has resulted in joint work on the mathematical foundations of the Facile programming language. This work is also relevant to the area of Foundational Models and Abstract Machines, as is the work on Chemical Abstract Machines for Facile.

Constructs found in the Facile programming language have stimulated interesting research from a more theoretical point of view. Here it is worth mentioning the joint work of S. Prasad and R. Amadio on localities and failures which shows a connection with previous work done at Sophia and Pisa. It is expected that the work on semantic foundations of distributed programming in the Facile language will be further developed. A collaboration between ECRC and Pisa has been initiated on this subject, and the cooperation between ECRC and ENS will be maintained.

A major aspect of this year's work on Pict has been the design, implementation, and theoretical study of the type system, extending earlier work by Pierce and Sangiorgi on type systems for the pi-calculus. The Pict group at Edinburgh (Pierce/Turner) have continued discussions with Didier Remy at INRIA-Roquencourt on the Pict language design, particularly the design of the typing system for extensible records.

We have begun discussions with the group at SICS (Parrow/Lisper) on an interface between Pict and the Mobility Workbench.

Sangiorgi, Amadio, and the group in Amsterdam (Klop, van Raamsdonk and van Oostrum) have been valuable sources of expertise in an ongoing investigation of the semantic foundations of Pict by Uwe Nestmann (a student at the Univ. of Erlangen) in collaboration with Pierce.

An implementation, called GOI-Tools, has been developed at Imperial College. This is a collection of tools to investigate the possibility of using the geometry of interaction to implement programming languages and obtain global intuitions about this semantic paradigm. This work is strongly related to the theoretical work on Logics for Concurrency and the λ–calculus. Part of the work for the Geometry of Implementation was done jointly with Vincent Danos (Paris VII) whilst Ian Mackie was visiting INRIA October-November 1993. The work relates to the works of Danos and Regnier, and Gonthier et al. which were reported under Foundational Models in last years PPR.

Chapter 6

Appendices

Below is a list of reports and publications mentioned in this document.

CONFER-52	J.A. Bergstra P. Klint	*The Toolbus: a Component Interconnection Architecture* Technical Report P9408, March 1994, Programming Research Group University of Amsterdam
CONFER-53	V. van Oostrom	*Confluence for Abstract and Higher-order Rewriting* Ph.D. thesis, March 1994, Free University of Amsterdam
CONFER-54	J.W. Klop, V. van Oostrom F.van Raamsdonk	*Combinatory Reduction Systems: introduction and survey* TCS, Vol.121, Nrs.1-2, Dec. 1993, guest eds. M. Dezani-Ciancaglini, S. Ronchi Della Rocca, M. Venturini-Zilli, A Collection of Contributions in Honour of Corrado Boehm on the Occasion of his 70th Birthday, p.279-308
CONFER-55	J.A. Bergstra I. Bethke A. Ponse	*Process Algebra with Combinators* In: Proc. 7th Workshop CSL '93 (Computer Science Logic), Swansea 1993, eds.: E. Borger, Y. Gurevich, K. Meinke, Springer LNCS 832, pp.36-65.
CONFER-56	J.A. Bergstra I. Bethke A. Ponse	*Process algebra with iteration and nesting.* Technical Report P9314b, Programming Research Group, University of Amsterdam, 1994.
CONFER-57	Z. Ariola J.W. Klop	*Cyclic Lambda Graph Rewriting* In: Proc. 9th Annual IEEE Symposium on Logic in Computer Science (LICS '94), Paris, July 1993, pp. 416-425.
CONFER-58	V. van Oostrom F.van Raamsdonk	*Comparing Combinatory Reduction Systems and* *Higher-order Rewrite Systems.* In: J. Heering, K. Meinke, B. Möller, T. Nipkow (Eds.) Higher-Order Algebra, Logic and Term Rewriting (HOA '93) Lecture Notes in Computer Science, Vol. 816, pp. 276-304
CONFER-59	V. van Oostrom F.van Raamsdonk	*Weak Orthogonality Implies Confluence: The Higher-Order Case* In: A. Nerode, Yu.V. Matiyasevich (Eds.), Logical Foundations of Computer Science (LFCS '94), Lecture Notes in Computer Science, Vol. 813. pp. 379-392

CONFER-60	Robin Milner	*An action structure for synchronous π-calculus*, Proc. FCT Conference, Szeged, Hungary, LNCS, Vol 710, August 1993, 87–105.
CONFER-61	Robin Milner	*Action calculi, or concrete action structures*, Proc. MFCS Conference, Gdansk, Poland, LNCS, Vol 711, September 1993, 105–121.
CONFER-62	Robin Milner	*Higher-order action calculi*, to appear in Proc. CSL conference, Swansea, October 1993.
CONFER-63	Robin Milner	*Pi-nets: a graphical form of pi-calculus*, Proc. ESOP'94, LNCS Vol788, Springer-Verlag, April 1994, 26–42.
CONFER-64	Robin Milner	*Bisimulation is not finitely (first-order) equationally axiomatisable*, in Proc. LICS '94.
CONFER-65	Martin Steffen Benjamin Pierce	*Higher-Order Subtyping*, Proc. IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET), S. Miniato, Italy, June 94, to appear.
CONFER-66	Davide Sangiorgi	*Bisimulation in higher-order calculi* Proc. IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET), S. Miniato, Italy, June 94, to appear.
CONFER-67	Davide Sangiorgi	*Locality and Non-interleaving Semantics in Calculi for Mobile Processes*, in Proc. International Symposium on Theoretical Aspects of Computer Science (TACS '94), LNCS 789, Springer Verlag.
CONFER-68	David Walker	*Algebraic proofs of properties of objects*, Proc. 5th European Symposium on Programming, Edinburgh, April 1994, D. Sannella (ed.), Springer-Verlag LNCS vol. 788, 501-516.
CONFER-69	David Walker	*On bisimulation in the pi-calculus*, to appear in Proc. 5th International Conference on Concurrency Theory, Uppsala, August 1994, Springer-Verlag LNCS.

CONFER-70	Robin Milner	*Action calculi IV: molecular forms*, November 1993.
CONFER-71	Robin Milner	*Action calculi V: reflexive molecular forms*, June 1994. (with Appendix by Ole Jensen)
CONFER-72	Robin Milner	*Action calculi VI: strong normalisation at higher-order*, December 1993.
CONFER-73	Robin Milner Alex Mifsud John Power	*Control Structures*, June 1994.
CONFER-74	Philippa Gardner	*Closed Action Calculi*, July 1994.
CONFER-75	Benjamin Pierce Didier Rémy David N. Turner,	*A Typed Higher-Order Programming Language Based on the Pi-Calculus*, 1994.
CONFER-76	Benjamin Pierce	*Programming in the Pi-Calculus: An Experiment in Programming Language Design*, February 1994.
CONFER-77	Michele Boreale Davide Sangiorgi	*A study of causality in the π-calculus*, Submitted for publication.
CONFER-78	Davide Sangiorgi	*On the bisimulation proof method*, July 1994.
CONFER-79	Benjamin Pierce Didier Rémy David Turner	*Pict: A typed, higher-order concurrent programming language based on the π-calculus*, 1994.

CONFER-80	Thomsen, B. et al	*Facile Antigua Release – Programming Guide*, Technical report ECRC-93-20, 1993.
CONFER-81	Roberto Amadio O. Ait-Mohamed	*An Analysis of π-calculus Bisimulations*, Technical report ECRC-94-2, 1994.
CONFER-82	Roberto Amadio.	*Translating Core Facile*, Technical report ECRC-94-3, 1994.
CONFER-83	Jean-Pierre Talpin.	*The Calumet Experiment - Part I: An Implementation of Group-Communication Protocols in Facile*, Technical report ECRC-94-4, 1994.
CONFER-84	Sanjiva Prasad.	*Cut Elimination for LU with Mingle*, Technical report ECRC-94-10, 1994.
CONFER-85	Sanjiva Prasad.	*The Positive Intuitionistic Fragment of LU*, Technical report ECRC-94-11, 1994.
CONFER-86	Roberto Amadio Lone Leth Bent Thomsen	*From Concurrent Functional Programs to Mobile Processes*, submitted for publication.
CONFER-87	Leth, L. Thomsen, B.	*Facile Chemistry Revised*, Technical report ECRC-94-36, 1994.
CONFER-88	Roberto Amadio Sanjiva Prasad	*Localities and Failures*, Technical report ECRC-94-18, 1994. FST&TCS'14 Madras, India, December 1994.
CONFER-89	Sanjiva Prasad	*Towards a Formulae-as-Types View of Communicating Applicative Programs (Extended Summary)*, Technical report ECRC-94-32, 1994.
CONFER-90	J.-P. Talpin	*The Calumet Experiment in Facile - A Model for Group Communication and Interaction Control in Cooperative Applications*, Technical report ECRC-94-26, 1994.
CONFER-91	Talpin, J.-P. Marchal, P. Ahlers, K.	*Calumet - A Reference Manual*, Technical report ECRC-94-30, 1994.

CONFER-92	R. Amadio O. Ait-Mohamed	*An analysis of π-calculus bisimulations* Technical Report 94-2, European Computer-Industry Research Center, 1994.
CONFER-93	R. Amadio	*Translating core Facile* Technical Report 94-3, European Computer-Industry Research Center, 1994.
CONFER-94	R. Amadio S. Prasad	*Localities and failures* Technical Report 94-18, European Computer-Industry Research Center, 1994.
CONFER-95	A. Asperti V. Danos C. Laneve L. Regnier	*Paths in lambda-calculus* Proceedings IEEE-LICS 94, Paris

CONFER-96	S. Abramsky S. J. Gay R. Nagarajan.	Interaction Categories and Typed Concurrent Programming *Deductive Program Design: Proceedings of the Marktoberdorf International Summer School*, 1994. NATO ASI Series F: Computer and Systems Sciences, Springer Verlag. To appear.
CONFER-97	R. L. Crole S. J. Gay R. Nagarajan.	*An Internal Language for Interaction Categories* In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.
CONFER-98	M. Huth.	*Interaction Orders as Games* In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.
CONFER-99	F. Lamarche	*Hypercoherences are Chu Spaces* In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.
CONFER-100	F. Lamarche.	*Proof Nets for Intuitionistic Linear Logic I: Essential nets*, Report available by anonymous ftp from `theory.doc.ic.ac.uk`.
CONFER-101	F. Lamarche.	*Dialectics: a model of linear logic and PCF*, Report available by anonymous ftp from `theory.doc.ic.ac.uk`.
CONFER-102	I. C. Mackie.	*The Geometry of Interaction Machine* In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.
CONFER-103	P. Malacaria.	*Studying equivalences of transition systems with algebraic tools* Theoretical Computer Science. To appear.
CONFER-104	D. Pavlović	*Categorical logic of concurrency and interaction, I: Synchronous processes.* In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, 1994. To appear.

CONFER-105	A. Asperti	*Linear logic, comonads, and Optimal Reductions* Fundamenta Informaticae, Special Issue devoted to Categories in Computer Science (invited paper). V.22, n.1, 1994.
CONFER-106	A. Asperti C. Laneve	*Interaction systems I: The theory of optimal reductions* Mathematical Structures in Computer Science. To appear.
CONFER-107	A. Asperti C. Laneve	*Paths, computations and labels in the λ-calculus* Theoretical Computer Science, Special Issue devoted to RTA '93 (Montreal)
CONFER-108	A. Asperti C. Laneve	*The family relation in Interaction Systems* Proc. of the International Symposium on Theoretical Aspects of Computer Science (TACS'94), Sendai, Japan. April 1994.
CONFER-109	A. Asperti V. Danos C. Laneve L. Régnier	*Paths in the λ-calculus. Three years* *of communications without understandings* Proc. of the International Symposium on Logic in Computer Science (LICS'94), Paris, France. 1994.
CONFER-110	A. Asperti	$\delta \circ !\epsilon = 1$ Internal Report of the Dipartimento di Matematica, Univeristà di Bologna. 1994.
CONFER-111	Damien Doligez Georges Gonthier	*Portable, Unobtrusive Garbage Collection for* *Multiprocessor Systems*, Twenty-First Annual ACM Symposium on Principles of Programming Languages, Portland.
CONFER-112	Gilles Dowek Thérèse Hardin Claude Kirchner	*Higher-order unification via explicit substitutions* INRIA report
CONFER-113	P.-A. Melliès	*Typed λ-calculi with explicit substitutions* *may not terminate*. Submitted to TLCA'95.

CONFER-114	A. Asperti C. Laneve	*The family relation in interaction systems* TACS Symposium, Sendai, April 1994.
CONFER-115	A. Asperti V. Danos C. Laneve L. Régnier	*Paths in the lambda-calculus* 9th LICS, Paris, July 1994.
CONFER-116	A. Asperti C. Laneve	*Interaction Systems 1: The theory of optimal reductions* to appear in Mathematical Structures in Computer Science.
CONFER-117	A. Asperti C. Laneve	*Interaction Systems 2: The practice of optimal reductions* revised version, submitted to Theoretical Computer Science.
CONFER-118	G. Boudol,	*Some chemical abstract machines* in "A Decade of Concurrency, Reflections and Perspectives", LNCS 803, 1994.
CONFER-119	Ferrari, G. Montanari, U. Quaglia, P.	*A π-calculus with Explicit Substitution: the Late Semantics* In Proc, MFCS'94, LNCS 841, 1994.
CONFER-120	Ferrari, G. Montanari, U. Quaglia, P.	*A π-calculus with Explicit Substitution* Full version submitted for publication, 1994.
CONFER-121	Ferrari, G. Montanari, U.	*Typed Additive Concurrency*, Submitted for Pubblication, August 1994.
CONFER-122	Ferrari, G. Montanari, U.	*Turning SOS Rules into Operations*, August 1994.
CONFER-123	Corradini Gadducci F. Montanari, U.	*Prime event structures and categorical models of term rewriting*, August 1994.
CONFER-124	Montanari, U. Pistore, M.	*Concurrent Semantics for the π-calculus*, July 1994.

CONFER-125	J. Parrow.	*Interaction Diagrams.* In de Bakker, de Roever and Rozenberg (Eds.): *A Decade of Concurrency*, REX school/Symposium, The Netherlands June 1993, Pages 477–508.
CONFER-126	J. Parrow D. Sangiorgi	*Algebraic Theories for Name-Passing Calculi* In de Bakker, de Roever and Rozenberg (Eds.): *A Decade of Concurrency*, REX school/Symposium, The Netherlands June 1993, Pages 509–529. Published as Springer Verlag LNCS 803 (1994). Extended version accepted for publication in *Information and Computation.*
CONFER-127	M. Dam.	*Model Checking Mobile Processes.* In Best (Ed.): *Proceedings of CONCUR'93*, pages 22–36, Published as Springer Verlag LNCS 715 (1993). Extended version accepted for publication in *Information and Computation.*
CONFER-128	B. Victor F. Moller	*The Mobility Workbench — A Tool for the π-Calculus.* In Dill (Ed.): *Proceedings of CAV'94*, pages 428–440, Published as Springer Verlag LNCS 818 (1994).
CONFER-129	B. Victor.	*A Verification Tool for the Polyadic π-Calculus.* Licentiat thesis, Department of Computer Systems, Uppsala University, May 1994.