CONFER

CONcurrency and Functions:

Evaluation and Reduction

Basic Research Action

Project Number: 6454

Periodic Progress Report
October 13, 1995

Contents

Chapter 1

Overview

This report contains the Third Periodic Progress Report for ESPRIT BRA Nr. 6454 (CONFER). It is the final report of this action.

The report contains 4 main parts: management at consortium level and at each site, deliverables (programs of each CONFER workshops 5 and 6, software deliverables), a progress report describing the technical work achieved during the third 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. Davide Sangiorgi from INRIA Sophia-Antipolis helped in preparing the area report on Foundational Models and Abstract Machines, Gérard Boudol from INRIA Sophia-Antipolis and Ugo Montanari from University of Pisa prepared the area report on Calculi, Rajagopal Nagarajan from Imperial College did the area report on Logics for Concurrency and the λ-calculus, Benjamin Pierce from Edinburgh University (and Cambridge) 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 organised around four main areas:

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

The objectives for the third year of CONFER put forth at the end of Year 2 of CONFER have been achieved at the end of Year 3. Significant results beyond these objectives have also been obtained, and results obtained in the CONFER action are still having a considerable impact in many areas beyond the objectives of CONFER. The overall impression of the third year of CONFER is that is has been as successful as the previous years.

In the area of Foundational Models and Abstract Machines, progress has been done in three areas. Firstly, the abstraction of interaction, i.e. Milner's action structures, has been studied. Secondly, calculi with bound variables have been treated in many ways. Little is known in the literature on these calculi, much less than for standard algebraic rules. This second part corresponds to work on optimal reductions, abstract reduction systems, explicit substitutions, CRS and cyclic CRS. Thirdly, 2 models for operational and denotational semantics of CHOCS and Facile have been presented.

In the area of Calculi work has been done at several levels. Some work done during the last years in this area has been finalised, resulting in conference papers and articles accepted for publication in technical journals. The work done this year still mainly deals with the semantics of the π-calculus and other related formalisms. As in previous years we have distinguished two main topics: *bisimulations*, and related methods for proving and verifying properties of processes, and *relating calculi*, which mainly means relating various calculi to the π-calculus. The topic of bisimulation is maturing, and this year's work is more oriented towards implementation questions than "fundamental" ones,

regarding the very concept of bisimulation. The π-calculus occupies a pivotal position in the area of relating calculi. A main theme in the reported work is to try to identify subsystems of the π-calculus, aiming at a particular expressive or discriminating power.

In the area of Logics for Concurrency and the λ-calculus, work has proceeded on the development of specification and verification techniques for concurrency, using the syntactic and semantic typed frameworks based on Interaction Categories. Typed calculi for synchrony and asynchrony have been developed, based on the structure of Interaction Categories. Related work has resulted in the development of a typed process calculus with modal μ-types. Progress has also been made in understanding the connections between logical structures and Interaction Categories; and in the denotational modelling of mobility. There has also been continued work on Game Semantics and Linear Logic as foundations for semantics of functional computation. Significant advances have been made in this area within the last year, and more generally, over the past three years.

In the area of Programming Languages several efforts have been completed. This year a major effort has been put into a release of PICT, and new developments for the type system of PICT to include linear types have been completed. Work on a distributed version of PICT suitable for use on the WWW has also begun. To demonstrate the industrial viability of programming distributed systems based on the integration of functional and concurrent programming several larger applications have been developed using the Facile language. All applications have been shown to an audience of industrialist, EU officials and members of the press at an event in Brussels. A major effort is currently being devoted to preparing a second release of Facile. Facile is now licenced by more than 25 sites in Europe, the US and Asia. A prototype implementation of (a variant of) Lamping's optimal graph reduction technique: the Bologna Optimal Higher-order Machine has been developed. New techniques on compile time optimisations via unfolding of programs useful for languages combining functional and concurrent computations have been developed. The mechanical verification of the concurrent garbage collector has been completed. A weak λ-calculus, $\lambda\sigma_w$, has been defined. This calculus has been used to prove the correctness of many skeleton compilers and runtime systems modeled as abstract machines.

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

During Year 3 of CONFER two workshops have been held. The first was organised by Rajagopal Nagarajan and Simon Gay and took place in October 1994 at Imperial. The workshop had 19 presentations and 35 participants. The second Year review took place on October 7th 1994 at Imperial. The second workshop in Year 3 was organised by Pierre-Louis Curien and took place in April 1995 at ENS. The workshop had 21

presentations and 37 participants.

The seventh and final CONFER workshop will take place October 11-13, 1995 at ECRC, organised by Lone Leth and Bent Thomsen. 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 not only the researchers strictly supported by the CONFER funding. Furthermore, a number of researchers outside the consortium are contributing to the effort.

The community of researchers in Europe exploring the area of computer science which, mainly due to the CONFER basic research action, has become known as integration of concurrency and functions. The integration of concurrency and functions has proved to be very useful in the construction of large scale highly reliable and dependable distributed systems. It furthermore shows promising for construction of mobile systems.

The work done in the CONFER BRA has been of very high quality on the frontiers of research. In many cases results from CONFER have defined the state of the art and stimulated research far beyond the project in places such as the US, Australia and Japan. Results obtained in the CONFER BRA have already proved to be of high industrial relevance and are currently being exploited by ECRC and its shareholder companies Bull, ICL and Siemens, as well as by small and medium sized companies. Results obtained in the CONFER BRA are also influencing work on the international LOTOS standard which most likely will encompass mobility inspired by the π-calculus and CHOCS.

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 third year of the CONFER project:

Two workshops have been held. The first took place in October 1994 at Imperial College with 19 presentations and 35 participants. The second workshop took place in April 1995 at ENS with 21 presentations and 37 participants. The second annual review took place on October 7th, 1994 at Imperial College.

At the workshop at ENS a management meeting was held to plan for the third annual review, and renewal of our project. We concluded in planning a proposal for a Framework-4 Working Group, named CONFER-2, which has been sent to the European Union in June 1995. We also decided on asking a 3-month extension of CONFER, which has been accepted by EU at end of August.

The seventh and final CONFER workshop will take place October 11-13, 1995 at ECRC.

The ftp site at Imperial College remains operational. All documents produced in CONFER can be found at this site at

`ftp://theory.doc.ic.ac.uk/theory/CONFER`.

A World Wide Web page describing the CONFER project has been developed by ECRC. It is placed as a sub page of the ECRC World Wide Web server

(`http://www.ecrc.de/research/collaborations/confer`)

To disseminate information about the progress of CONFER the (technical) coordinators produced an overview of the second year results. This report was published in Bulletin of EATCS, Number 55, 1995, pp. 68-87.

3.2 CWI

3.2.1 Research Directions

In the past year CWI has continued to work in the following directions.

3.2.1.1 Higher-order Rewriting

In several calculi featuring in the CONFER project the property of termination (strong normalisation) is important. Van Raamsdonk has made a study of normalisation in the framework of classical λ-calculus in cooperation with P. Severi (Technical University Eindhoven), arriving at new proofs of fundamental lemma's such as the finite developments lemma, the strong normalisation theorem of simply typed lambda calculus, and related results. Current work is concerned with obtaining the normalisation property for outermost reduction in (weakly) orthogonal higher-order rewriting systems. For instance, for λ-calculus with β- and η-reduction it does not seem to be established yet that parallel-outermost evaluation is normalising.

3.2.1.2 Term Rewriting and λ-calculus with Explicit Recursion

This subject, pursued by J.W. Klop in cooperation with Z.M. Ariola (University of Oregon), is concerned with term graph rewriting both for first-order rewrite systems and for λ-calculus. We consider cyclic term graphs and cyclic λ-graphs. The aim is to design suitable transformation (rewrite) rules for such graphs, such that confluence holds. Especially for the lambda calculus case it is a delicate matter how to treat free and bound variables. The search for a fully satisfactory system has not yet been completed. Present concern is with systems of nested recursion equations; an interesting feature is that there is a certain similarity with λ-calculi with explicit substitution.

To serve as semantics for λ-calculus with explicit recursion, J.W. Klop has worked in infinitary λ-calculus in cooperation with J.R. Kennaway and M.R. Sleep (both University of East-Anglia) and F.J. de Vries (Hitachi, Japan). This work is concerned with λ-terms that may be infinitely deep (like the well-known Böhm trees), while reduction may have transfinite length.

3.2.1.3 Process Algebra

Previous work by J.A. Bergstra in cooperation with P. Klint (University of Amsterdam) on the ToolBus is extended with features like conditionals and simple operations on the built-in data type of terms. Also discrete time is introduced.

3.2.2 Persons and Exchanges

CWI participants are: Jan Willem Klop, Femke van Raamsdonk.

F. van Raamsdonk is a PhD-student under supervision of J.W. Klop; her thesis about higher-order rewriting is relevant to the CONFER project and is expected to be completed early next year (1996).

Part of the work is subcontracted to the University of Amsterdam: Jan Bergstra, Piet Rodenburg.

During the past year CWI has participated in the following exchanges in the framework of CONFER:

1. Ian Mackie (Imperial College/Ecole Polytechnique) visited CWI January 9-12, 1995, and gave a talk on his work on (near) optimal implementations of λ-calculus.

2. J.W. Klop (CWI) visited Università di Pisa 3-29 October, 1994, to give a graduate course on term rewriting and to give a seminar on work done in CONFER concerning lambda calculus with explicit recursion.

3. Paul-André Melliès (INRIA Rocquencourt) works in the group of Jan Willem Klop at the Vrije Universiteit in Amsterdam from November 1994 until the end of 1995 on issues in abstract rewriting.

3.2.3 Publications, Technical Reports

1. Z.M. Ariola, J.W. Klop
   Equational term graph rewriting
   CWI Report CS-R9552, July 1995
   (short version also to appear in Fundamenta Informaticae 1995)

2. J.A. Bergstra, P. Klint
   The discrete Time ToolBus
   Technical Report P9502, March 1995, Programming Research Group, University of Amsterdam

3. Jan A. Bergstra, Inge Bethke, Alban Ponse
   Process Algebra with Iteration and Nesting
   The Computer Journal,
   Vol. 37, No. 4, 1994

4. J.R. Kennaway, J.W. Klop, M.R. Sleep, F.-J. de Vries
   Infinitary lambda calculus
   (published in RTA-95 proceedings as 'Infinitary lambda calculi and Böhm models', Kaiserslautern, April 1995, Springer LNCS 914, p.257-270, ed. Jieh Hsiang)
   CWI Report CS-R9535, May 1995

5. V. van Oostrom, F. van Raamsdonk
   Weak orthogonality implies confluence: the higher-order case
   CWI Report CS-R9501, January 1995
   (short version already appeared in proceedings of LFCS '94, Springer LNCS 813, p.379-392)

6. F. van Raamsdonk, P. Severi
   On normalisation
   CWI Report CS-R9545, June 1995

3.3 University of Cambridge and University of Edinburgh

3.3.1 Research directions

Work on action structures continues in Cambridge and Edinburgh. In 1993-4, the main advance was in defining action calculi, which are action structures of a concrete nature, with added structure, and which represent a wide variety of calculi of computation – functional, sequential, and concurrent. In 1994-5 the emphasis has been on

1. refining the presentation of action calculi;
2. investigating a natural variant: closed or name-free action calculi;
3. defining the category of their models.

The study of the meaning of bisimulation in higher-order calculi, reported last year, was continued this year.

The Pict language and compiler have received a great deal of polishing and documentation. It is being distributed to a number of external sites and has been used to teach several courses. Some experimental extensions are now under consideration, including a "linear" type system with use-once channel types and a library supporting a physically distributed channel abstraction.

3.3.2 Perspectives, work in progress

Action structures. The refinement of the presentation of action calculi is in [1]; the paper presents several examples and uses a new notion, action graphs, in doing so. The analysis of closed action calculi appears in [2] and [unpublished report 1]. The model theory of action calculi, so-called control structures, appears in [3] and [4].

Future work will be in classifying action calculi according to their dynamics; it appears that the model theory will help considerably here. Work is also in progress to find ways of defining a bisimilarity congruence uniformly over action calculi. For this and other purposes, the graphical presentation of action calculi is being examined in considerable detail.

We have recently given an Action Calculus for the Join Calculus of Fournet and Gonthier, and are considering extension of this to the Join Calculus with explicit distribution and site failure. This work was started during the visit of Fournet to Cambridge in August and September, 1995.

The definition of control structure is broadly category theoretic. However, categorical formulations of similar phenomena typically do not have a "set of names" explicit, but have a category with finite products to represent a category of arities, the names being an internal language for the category. So control structures, modulo two mild conditions, have been characterised in terms of fibrations with structure, yielding *fibrational control structures*, and these have been further shown equivalent to the simpler but less malleable *elementary control structures*.

Pict programming language. The Pict compiler is being distributed to a gradually increasing number of sites; a general announcement of its availability is expected this fall. A number of experimental extensions are planned, including support for separate compilation (based on a recent rewrite of the code generator), a simple higher-order module system, and a fully typed intermediate language that will enable more aggressive use of static type information (including linearity) in code optimisation. Our largest current programming experiment is a library providing a physically distributed channel abstraction with the same semantics as — but immiscible with — ordinary pi-calculus channels. Simple client-server demonstration programs are running now; more sophisticated applications are planned.

Work is in progress on the abstract semantics of Pict, attempting to give principled definitions of observational equivalence of Pict programs and of correctness of Pict abstract machines. In a related effort, we are considering the structure required for an abstract "π-calculus labelled transition system."

3.3.3 Person and exchanges

On the appointment of Milner at Cambridge (on 1 Jan 1995), several of the group (Pierce, Sewell, Compagnoni, Jensen) moved from Edinburgh to Cambridge. Power and Gardner continue to work on these topics at Edinburgh.

Sewell was part-time employed within the project, from October 94 to March 95. His PhD thesis, which is relevant to the CONFER project, was completed in March 95.

In the spring of 1995, Pierce and Turner spent several days with the group at Inria-Rocquencourt for discussions of their Distributed Pict project. Later in the summer, we hosted return visits by Remy (during the Newton Institute workshop on Advances in Type Systems) and Fournet (a two-month "summer-student" visit to work on Pict and Distributed Pict with Pierce.) Earlier in the spring, Pierce visited Warwick University to teach a two-day unit on Pict programming in Walker's graduate course on concurrency theory.

3.3.4 Publications

1. Robin Milner, "Calculi for Interaction," To appear in Acta Informatica, 1996.

2. Philippa Gardner, "A Name-free Account of Action Calculi," Proceedings of Mathematical Foundations of Programming Semantics, New Orleans, 1995.

3. Alex Mifsud, Robin Milner and John Power, "Control Structures," Proceedings of IEEE Conference on Logics in Computer Science, San Diego, 1995.

4. Claudio Hermida and John Power, "Fibrational control structures," Proceedings of CONFER Conference, 1995.

5. Ghelli, G., Pierce, B.: "Bounded Existentials and Minimal Typing". Submitted to Information and Computation, 1995.

6. Hofmann, M., Pierce, B.: "Positive Subtyping". Proceedings of Twenty-Second Annual ACM Symposium on Principles of Programming Languages, January 1995. Full version submitted for publication.

7. Kobayashi, N., Pierce, B.C., Turner, D.N.: "Linearity and the Pi-Calculus". Submitted to POPL '96.

8. Pierce, B.C., Turner, D.N.: "Concurrent Objects in a Process Calculus". Theory and Practice of Parallel Programming (TPPP), Sendai, Japan, Nov. 1994. (Invited talk.) Springer-Verlag LNCS 907.

9. Pierce, B.C.: "Foundational Calculi for Programming Languages". CRC Press Handbook of Computer Science and Engineering. To appear, 1996.

3.3.5 Unpublished research reports

1. Philippa Gardner and Robin Milner, *A short note on Closed Action Calculi,* DRAFT, 1995.

2. John Power *Elementary control structures* January 1995

3. John Power *Control structures III: arity monoids and their associated naming monoids* August 1994

4. Pierce, B.C.: "Programming in the Pi-Calculus: An Experiment in Programming Language Design". Tutorial notes on the Pict language. Available with the Pict distribution, 1995.

5. Pierce, B.C., Turner, D.N.: "Pict: A Programming Language Based on the Pi-Calculus". Invited journal article, in preparation, 1995.

3.3.6 Software

1. Benjamin Pierce and David Turner, *Pict: A typed, higher-order concurrent programming language based on the π-calculus.* Version 3.6b, 1995.

3.4 ECRC

3.4.1 Research directions

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.

As mentioned in last years report, the availability of the first release of Facile – the Facile Antigua Release – was publicly announced in July 1994. At the same time technology transfer to the ECRC shareholder companies took place. Currently, there are more than 25 licenced sites, including sites in Europe, Asia and the US.

In the 3rd Year of CONFER the group at ECRC has mainly concentrated on the development of major demonstrators and applications. Three major applications have been developed: Mobile Service Agents, Einrichten and Cooperative Mechanical Repair.

These demonstrators were first shown on November 30, 1994 at the Demonstration Day at the Concert Noble in Brussels, celebrating ECRC's 10th Anniversary. This event had participation of representatives of the ECRC shareholder companies and their customers in the European IT industry, representatives of the European Union Commission and members of the press (e.g. the event was described in an article by Information Technology correspondent, David Tebbutt, The European – 22-28 December 1994, p. 25).

Mobile Service Agents

The last few years have seen a growing interest in mobile computing. The use of portable computers in conjunction with wireless communication networks provides new means of interaction between users and computers. As the hardware infrastructure begins to materialise, there are more possibilities (and greater need!) for software applications that are adapted to this highly dynamic setting. Facile's basis on formal process models for mobility (e.g., the π-calculus and CHOCS) and on functional programming languages has allowed it to inherit features and a basic approach that provide an edge in developing a significant class of mobile applications. We call this class Mobile Service Agent (MSA). MSAs are self-contained pieces of software that can move between computers on a network. Agents can serve as local representatives for remote services, provide interactive access to data they accompany, and carry out tasks for a mobile user temporarily disconnected from the network. Agents also provide a means for the set of software available to a user to change dynamically according to the user's needs and interests. In the MSA architecture, the Global Information Infrastructure (GII) is seen as consisting of sites each supporting various services based on agents. Upon connecting to a site, a "directory agent" is first dispatched to the user's machine. This directory agent serves simultaneously as an informational guide to the site and as a conduit through which other agents from the site can be introduced to the user's computer.

As proof of concept we have constructed a demonstration of the MSA architecture that is centered on the scenario of a person from ECRC in Munich going to a meeting in

Brussels. The person is carrying a portable computer and is temporarily disconnected from the GII while in transit.

The two sites, ECRC and Brussels, both provide various services that the user may access at various points in the trip. For example, prior to departure from ECRC, the user can connect to Brussels via the GII and obtain such services (implemented as agents) as an interactive map of the conference site and an agenda for the conference. While in transit the user can continue to use these services even though the connection to Brussels is unavailable. Upon reconnection to the GII in Brussels or elsewhere, the agenda agent automatically takes advantage of the opportunity to communicate with its home location and updates itself with the latest information and bulletins for the user.

This demonstrator was mainly developed by Pierre-Yves Chevalier, Fritz Knabe, Lone Leth and Bent Thomsen. A paper on the subject of mobile agents (ECRC/M3/R9) has been published in the July issue of Communications International and a technical report giving more details of the demonstration has been written (ECRC/M3/R8). (Some further description of the technology and the demonstration, including photographs, given at the ECRC 10th anniversary may be retrieved through the www: http://www.ecrc.de/research/ds/msa/)

The MSA demonstration has been accepted for the European IT Conference '95 exhibition in Brussels in November 1995.

Einrichten

Einrichten is a collaborative design system developed at ECRC. It is a demonstration of interactive graphics and real-time video for the purpose of interior design. The system combines the use of a heterogeneous database system of graphical models, an augmented reality system, and the distribution of 3D graphics events over a computer network. This application shows how improvements in computing and communication hardware can be combined with sophisticated software platforms to produce powerful results for end users.

The scenario for this application consists of an office manager who is working with an interior designer on the layout of a room. The office manager intends to order furniture for the room. On a computer monitor the pair see a picture of the room from the viewpoint of the camera. By interacting with various manufacturers over a network, they select furniture by querying databases using a graphical paradigm. The system provides descriptions and pictures of furniture that is available from the various manufacturers who have made models available in their databases. Pieces or groups of furniture that meet certain requirements such as colour, manufacturer, or price may be requested. The users choose pieces from this "electronic catalogue", and 3D renderings of this furniture appear on the monitor along with the view of the room. The furniture is positioned using a 3D mouse. Furniture can be deleted, added and rearranged until the users are satisfied with the result; they view these pieces on the monitor as they would appear in the actual room. As they move the camera they can see the furnished room from different points of view.

The users can consult with colleagues at remote sites who are running the same system. Users at remote sites manipulate the same set of furniture using a static picture of the room that is being designed. Changes by one user are seen instantaneously by all of the others, and a distributed locking mechanism ensures that a piece of furniture is moved only by one user at a time. In this way groups of users at different sites can work together on the layout of the room. The group can record a list of furniture and the layout of that furniture in the room for future reference.

The 3D graphics and augmented vision components of the system are built with GSP, a software platform which combines interactive 3D graphics and computer vision technology to calibrate, align, and display 3D models and real-time video.

The system also provides a means of distributing graphics events in a transparent manner. The communication of these events to multiple GSP applications, the locking, and the management of users joining and leaving the group is achieved through Facile, which provides the tools for reliable connections, disconnections and multicasts.

This application was demonstrated at the 1995 G7 technology summit in Brussels.

This application has mainly been developed by Andre Kramer and Klaus Ahlers. A paper (ECRC/M3/R4) on the application will be presented at Eurographics '95, Maastricht, the Netherlands. A technical report (ECRC/M3/R5) giving more details of the application and its implementation has been produced.

Cooperative Mechanical Repair

The main goal of this application is to assist a field engineer in maintaining and repairing a complex artifact (e.g., an aircraft engine) using augmented reality (AR) technology.

Using a head-mounted or portable display with a see-through visor, the engineer is able to work on the engine while "seeing" additional computer generated information (e.g., identification of a component). Annotations may identify the names of parts, describe their function, or present other important information like maintenance or manufacturing records. AR may lead the mechanic through a specific task by highlighting parts that must be sequentially removed and showing the path of extraction. The system may also provide safety information. Parts that are hot or electrified can be flagged as a constant reminder. The mechanic may also be assisted by a remote expert who can control what information is displayed on the mechanic's AV system. The application supports remote mechanical inspection and allow the field engineer and a remote expert to interact and complete the repair task together. Access to maintenance and supplier databases (e.g., history, prices, physical characteristics) will be available. Diagnostic information (generated by a remote expert system) may be presented.

On the technical side and apart from the AR, the main features of this application are:

- Video transmission
- Event broadcast (remote interaction and tracking)
- Fault-tolerance (replicated data and dynamic joining)

- Interoperability C++/Facile (contract)

This application has mainly been developed by Jean-Yves Litzler, Andre Kramer and Pierre-Yves Chevalier.

Other applications

The Calumet cooperative application for teleconferencing, demonstrated at the 1st Year review at CWI, was further refined during the 2nd Year of CONFER. In the 3rd Year of CONFER Calumet has been used regularly for presentations at ECRC. Calumet has also been integrated into the Mobile Service Agents demonstration and is used as an example of services that can be offered. Furthermore it is a good example of how high level programming facilitates code reuse. Only 50 lines of code was needed to integrate the Calumet system into the MSA architecture. The Calumet system was mainly developed by Jean-Pierre Talpin.

Furthermore, Marc Bourgois at ECRC is using the Facile system to implement high level models of interaction and coordination based on generative communication and Linear Logic (LO). This work is taking place in the context of the COORDINATION BRA 9102.

Facile

Facile as it stands now is mature enough to be used for sizeable applications. However, Facile is not in a frozen state. In fact we hope that Facile will continue to play the role of a research vehicle where theoretical results will be applied and transferred into real usage. There are several extensions to the Facile system under way or planned. Furthermore, we anticipate adjustments and new features being added as the "real" needs from users start to provide feed back to the further development.

In the Autumn of 1995 we expect to make a second release of Facile – the Facile Barbados Release. Apart from tuning and bug fixes, the major novel features of the Facile Barbados Release will be support for multi-platform environments (SPARC and Intel 486 processors), reliable group communication and support for interoperability with systems implemented in different programming languages (such as C, C++, prolog) via the ONC-PRC/XDR industrial standard.

The support for multi-platform environments is based on a novel approach to transmission of code in heterogeneous environments, developed by Fritz Knabe for his forthcoming Ph. D. thesis. This is in fact one of the key novelties enabling the development of applications based on the mobile agent principle.

A critical issue when operating in a multi-platform environment is the efficient transmission of code (functions and processes) across different processors. In essence, machine code cannot generally be communicated, and source code would have to be recompiled and further suffers from not preserving static binding. So a natural approach is to send closures in a machine independent representation. Functions and processes that may be sent to a foreign environment must be kept in an intermediate representation. Since it is not desirable to keep all functions and processes in machine

independent representation it is necessary to specify which are kept and which are not. The initial approach will be based on program annotations, where the programmer explicitly specifies which objects are transmissible and which are not. It is hoped that at some point in the future these annotations can be eliminated and that program analysis can detect which objects should be kept and which need not be kept in machine independent representation.

The communication paradigm in the Facile Antigua Release is based on hand shake (synchronous) communication over typed channels. On top of this primitive one may build more elaborate communication protocols (i.e. group communication (ECRC/M2/R4)). However, in some applications it is more natural to think directly in terms of point-to-point streams (video and audio are good examples) or multi casts or transactions. It is clear that such other paradigms can be "simulated" by implementing them using synchronous channels. However, it may also be advantageous to have these communication paradigms as primitives since some may be implemented directly in hardware or supported directly by communication sublayers. Based on the interface described in (ECRC/M2/R4) reliable group communication has been added as a communication primitive in Facile. The implementation takes advantage of multi-casting support from the underlying network protocols (such as MBONE on the Internet) when such support is available. Regarding the semantics of reliable group communication we have, at first, taken the "SML approach" by giving an implementation in terms of channels and define this as the semantics, but have the implementation use the actual hardware or communication sublayer. However, this should be considered a temporary solution. A challenging long term goal is to integrate such paradigms and have them coexist, both in terms of semantics and in terms of implementation. The new group communication mechanism has been developed by Andre Kramer.

An extension of the Facile environment for interoperability between processes in different address spaces, using the ONC-RPC protocol and the XDR data representation are being developed by Ioannis Nikolaidis. This extension will permit the support of traditional RPC-based client-server applications. The extension will include, (a) a stub generator for the generation of Facile (b) modifications to the runtime environment for support of the necessary communication primitives and (c) accompanying Facile library for support of the compound data structure operations and for the connection management operations. The ONC-RPC extensions will allow the direct interoperation with well-known legacy applications, such as NFS, and enhance the systems programming potential of the Facile environment.

It is expected that the ONC-RPC extensions will replace the contract mechanism used in current applications to interface with non-Facile systems.

Theory

Work on Theory at ECRC has in Year 3 mainly concentrated on promulgation of results (the lack of which was criticised at the 2nd Year review).

In Year 2 of CONFER Roberto Amadio, Lone Leth and Bent Thomsen have studied the connection between the concurrent functional core of Facile and the π-calculus.

This work has been further pursued in Year 3 of CONFER and has led to a paper (ECRC/M3/R7), presented at FCT'95 in Dresden, Germany. The full version of this paper is presented in (ECRC/M3/R6).

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). The results of the two papers have been merged and accepted for publication in the Journal of Formal Aspects of Computing, 1995. A short version (ECRC/M3/R10) will be considered a joint CONFER/LOMAPS deliverable and the long version (ECRC/M3/R11) will be considered a CONFER deliverable.

The work by Bent Thomsen on "Polymorphic Sorts and Types for Concurrent Functional Programs" (ECRC/M1/R4) was presented at the 6th International Workshop on the Implementation of Functional Languages, UEA Norwich, UK, 1994. The corresponding paper (ECRC/M3/R1) is published in the proceedings of the workshop. It is worth mentioning that this work is being further pursued, mainly by Tsung-Min Kuo, in the LOMAPS BRA 8130 where a full implementation of a polymorphic type system for Facile based on effect analysis is being developed.

A paper (ECRC/M3/R3) by Bent Thomsen entitled "A Theory of Higher Order Communicating Systems", has been published in *Information and Computation* Vol. 116, No 1, pp. 38-57, January 1995. This paper describes the theory of CHOCS and is considered a CONFER deliverable. The paper was written during Year 1 of CONFER and revised during Year 2.

A paper (ECRC/M3/R12) by Bent Thomsen and Samson Abramsky on the denotational semantics for CHOCS has been accepted for publication in the Journal of Theoretical Computer Science (TCS). The paper is currently under revision.

Bent Thomsen gave an invited talk: "Logic, Domains and Higher-Order Processes" at the Workshop on Logic, Domains, and Programming Languages (LDPL'95), Department of Mathematics, Technische Hochschule Darmstadt, Germany, May 24–27, 1995. A paper (ECRC/M3/R13) is being written for the proceeding of this workshop. This paper will be considered a CONFER deliverable.

During Year 3 of CONFER a lively interaction with researchers from Inria-Sophia, CNET, University of Sussex, Imperial College, University of Edinburgh, University of Cambridge, SICS, University of Pisa and ECRC has taken place. Several important topics for further theoretical development have been identified and are expected to be pursued in the LOMAPS BRA 8130 action and the proposed CONFER-2 working group.

3.4.2 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 2nd Year report on the

CONFER project (ECRC/M3/R2).

Lone Leth and Bent Thomsen have developed a World Wide Web page describing the CONFER project. It is placed as a sub page of the ECRC World Wide Web server (http://www.ecrc.de/research/collaborations/confer).

The 7th CONFER workshop is being arranged by Lone Leth and Bent Thomsen and will be held at ECRC (October 11-13, 1995).

3.4.3 Persons and exchanges

The following ECRC personnel has been engaged in the action in Year 3 of CONFER: Andre Kramer, Tsung-Min Kuo, Lone Leth, Jean-Pierre Talpin and Bent Thomsen.

Klaus Ahlers, Marc Bourgois, Pierre-Yves Chevalier, Alessandro Giacalone, Fritz Knabe, Jean-Yves Litzler and Ioannis Nikolaidis have also contributed to the development of Facile and the applications developed with the system.

Members of the group have taken part in the two CONFER workshops (no. 5 and 6) held in Year 3. Fritz Knabe, Lone Leth and Bent Thomsen took part in the 5th workshop. Fritz Knabe gave a talk entitled: "Efficient function transmission in Facile". Alessandro Giacalone, Lone Leth and Bent Thomsen took part in the 2nd Year review. Alessandro Giacalone gave the area report on programming languages.

Lone Leth and Bent Thomsen took part in the 6th workshop.

Prof. Ugo Montanari visited ECRC on March 8th.

3.4.4 Perspectives, work in progress

As the CONFER project approaches its ending date it is worthwhile to consider the achievements of CONFER. In the view of ECRC the CONFER BRA has achieved very important results in all the areas of main interest to ECRC:

Connection between calculi and programming languages
Abstract machines and efficient implementations
Types and concurrency
Primitive constructs
Dynamically evolving systems
Exceptions
Applications

and we feel that the collaboration has been extremely beneficial, and we hope (know) that our developments on Facile and end-user applications, such as Calumet, Einrichten and MSA, have stimulated further studies.

The Facile system is drawing considerable interest from several departments in the ECRC shareholder companies, as well as some of their customers and other European companies. This includes companies developing technology for electronic financial markets and organisations responsible for air traffic control.

Our main interest for the future is the further development of the Facile programming language and pursuing experimental medium to large scale applications, as well as, with partners, developing industrial applications using Facile.

We hope that Facile will continue to play the role of a research vehicle where theoretical results will be applied and transferred into real usage. There are several extensions to the Facile system under way or planned. Furthermore, we anticipate adjustments and new features being added as the "real" needs from users start to provide feed back to the further development.

Beyond the Facile Barbados Release, we expect future releases of Facile to include a much lighter weight implementation (the current implementation requires 10MB per Facile node) and integration of a set of sophisticated "middleware" services, such as security, authentication and persistence. As already mentioned, a new type system for Facile based on the principle of effects analysis is being developed in the context of the LOMAPS BRA 8130 action.

To promulgate our experience about the design, implementation and use of Facile we are currently writing a book chapter on these topics:

Bent Thomsen, Lone Leth and Tsung-Min Kuo: "FACILE – from Toy to Tool", To appear in: *ML with Concurrency: Design, Analysis, Implementation, and Application*, Flemming Nielson (Editor), Springer-Verlag, 1996.
This book chapter will be considered a deliverable for CONFER. (ECRC/M3/R14).

As already mentioned, during Year 3 of CONFER a lively interaction with researchers from Inria-Sophia, CNET, University of Sussex, Imperial College, University of Edinburgh, University of Cambridge, SICS, University of Pisa and ECRC has taken place. Several important topics for further theoretical development have been identified and is expected to be pursued in the LOMAPS BRA 8130 action and the proposed CONFER-2 working group.

3.4.5 Publications, technical reports

A set of 9 reports and published papers has been produced as deliverables during Year 3 of CONFER:

ECRC/M3/R1: Thomsen, B.: "Polymorphic Sorts and Types for Concurrent Functional Programs", Proceedings of the 6th International Workshop on the Implementation of Functional Languages (Ed. J. Glauert), UEA Norwich, UK, 1994.

ECRC/M3/R2: Levy, J.-J., Thomsen, B., Leth, L., Giacalone, A.: "Second Year Report for Esprit Basic Research Action 6454-CONFER CONcurrency and Functions: Evaluation and Reduction", Bulletin of EATCS, Number 55, 1995, pp. 68-87.

ECRC/M3/R3: Thomsen, B.: "A Theory of Higher Order Communicating Systems", *Information and Computation* Vol. 116, No 1, pp. 38-57, January 1995.

ECRC/M3/R4: Ahlers, K. et al.: "Distributed Augmented Reality for Collaborative Design Applications", Technical report, ECRC-95-03, 1995.

ECRC/M3/R5: Ahlers, K. et al.: "Distributed Augmented Reality for Collaborative Design Applications", To appear in Proceedings of Eurographics '95, Maastricht, the Netherlands (August 28 to 1 September 1995).

ECRC/M3/R6: Amadio, R., Leth, L. and Thomsen, B.: "From a Concurrent λ-calculus to the π-calculus" Technical report, ECRC-95-18, 1995.

ECRC/M3/R7: Amadio, R., Leth, L. and Thomsen, B.: "From a Concurrent λ-calculus to the π-calculus" To appear in Proceedings of Fundamentals of Computation Theory, FCT'95.

ECRC/M3/R8: Thomsen, B., Leth, L., Knabe, F. and Chevalier, P.-Y.: "Mobile Agents", Technical report, ECRC-95-21, 1995.

ECRC/M3/R9: Thomsen, B., Knabe, F., Leth, L. and Chevalier, P.-Y.: "Mobile Agents Set to Work", Communications International, July, 1995.

The papers ECRC/M3/R2, R4, R6 and R8 have been placed at the CONFER ftp site at Imperial College.

A set of 5 papers is in the process of being written or published and will be considered CONFER deliverables.

ECRC/M3/R10: Leth, L. and Thomsen, B.: "Some Facile Chemistry (Summary)", Accepted for publication in Formal Aspects of Computing, 1995.

ECRC/M3/R11: Leth, L. and Thomsen, B.: "Some Facile Chemistry", Accepted for electronic publication in Formal Aspects of Computing, 1995.

ECRC/M3/R12: Thomsen, B. and Abramsky, S.: "A Fully Abstract Denotational Semantics for the Calculus of Higher Order Communicating Systems", Accepted for publication in TCS, 1994.

ECRC/M3/R13: Thomsen, B.: "Logic, Domains and Higher-Order Processes", To appear in the Proceedings of Workshop on Logic, Domains, and Programming Languages (LDPL'95), Technische Hochschule Darmstadt, Germany, May 24–27, 1995.

ECRC/M3/R14: Thomsen, B., Leth, L. and Kuo, T.-M.: "FACILE – from Toy to Tool", To appear in: *ML with Concurrency: Design, Analysis, Implementation, and Application*, Flemming Nielson (Editor), Springer-Verlag, 1996.

3.4.6 Software

The main thrust of work at ECRC during Year 3 of CONFER has been on the software systems:

Facile As mentioned above and in last years report, the availability of the first release of Facile – the Facile Antigua Release – was publicly announced in July 1994. At the same time technology transfer to the ECRC shareholder companies took place. Currently, there are more than 25 licenced sites, including sites in Europe, Asia and the US. A second release – the Facile Barbados Release – is in preparation.

Calumet The Calumet cooperative application for teleconferencing, demonstrated at the 1st Year review at CWI, was further refined during the 2nd Year of CONFER. In the 3rd Year of CONFER Calumet has been used regularly for presentations at ECRC. Calumet has also been integrated into the Mobile Service Agents demonstration and is used as an example of services that can be offered. Furthermore it is a good example of how high level programming facilitates code reuse. Only 50 lines of code was need to integrate the Calumet system into the MSA architecture.

MSA A new technology, called Mobile Service Agents, is based on the Facile programming environment. The purpose of this technology is to ease the development of new distributed system infrastructures by enabling systems to communicate not just data but also programs. Transmissions can contain chunks of program code that can travel around a network and execute at different nodes. We call these travelling programs "mobile agents". By communicating agents applications can essentially program the network to suit their particular needs.

Einrichten Einrichten is an application that melds distribution, sophisticated graphics, and live video to permit collaborative interior design work for widely-separated participants. Facile is used in this system for distributing graphics events in a transparent manner and for locking of graphical objects, and the management of users joining and leaving the group

Cooperative Mechanical Repair The main goal of this application is to assist a field engineer in maintaining and repairing a complex artifact (e.g., an aircraft engine) using augmented reality (AR) technology. The Facile implementation of management and distribution of graphical event done for the Einrichten system has been reused in this application with minor modifications.

LO Facile is used in the COORDINATION BRA 9102 for implementing high level models of interaction and coordination based on generative communication and Linear Logic (LO).

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 distributed systems. 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, Paris, and Sophia-Antipolis has continued its mainly theoretical work in the following research areas:

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

This year we have pursued our collaboration with ECRC (on calculi) and with SICS (on logic) and we have tightened our connections with INRIA-Sophia: C. Lavatelli is finishing her thesis under the joint direction of P.-L. Curien and G. Boudol, and R. Amadio has moved from Nancy to Sophia-Antipolis to join Boudol's group.

3.5.2 Persons

The following researchers are engaged in the action: Roberto Amadio (CNRS-INRIA Sophia-Antipolis, previously CNRS-INRIA, Nancy), Pierre-Louis Curien (CNRS, LIENS), Vincent Danos (CNRS, Paris VII), Carolina Lavatelli (PhD Student, LIENS Paris), and Laurent Regnier (CNRS, Marseille).

3.5.3 Reports and Publications

A set of 4 reports has been produced as deliverables during Year 3 of CONFER:

1. R. Amadio and M.Dam. Reasoning about higher-order processes. In *CAAP 95, Aarhus*, SLNCS 915. Also appeared as SICS Research Report 94-18, 1994. This deliverable is shared with the LOMAPS project.

2. R. Amadio and L. Leth and B. Thomsen. ¿From a concurrent λ-calculus to the π-calculus. In *Foundations of Computation Theory 95, Dresden*, SLNCS to appear. Expanded version appeared as ECRC-95-18. Available at ftp.ecrc.de.

3. V. Danos and L. Regnier. Reversible and Irreversible Computations (GOI and Environment Machines), available as revirrev.[dvi, ps].[Z, gz] at http://boole.logique.jussieu.fr/www.danos/experience.html

4. C. Lavatelli. Full abstraction for an extended resource language, Technical Report LIENS 95-18.

Deliverable 2 and 4 relate to the *Calculi* area. Deliverable 1 relates to the *Logic* area. Deliverable 3 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, and B. Thomsen from ECRC, has continued the investigation of the concurrent semantics of the Facile programming language. We have determined a simple parallel concurrent λ-calculus that can be regarded as the kernel of concurrent-functional languages such as LCS, CML and Facile, and we have proven the adequacy of a compact translation of this calculus into the π-calculus.

   A second collaboration with M. Dam (SICS) concerns the specification and verification problem for process calculi such as Chocs, CML and Facile where processes or functions are transmissible values. Our work takes place in the context of a static treatment of restriction and of a bisimulation-based semantics. As a paradigmatic and simple case we have concentrated on (Plain) Chocs. We have shown that Chocs bisimulation can be characterised by an extension of Hennessy-Milner logic including a constructive implication, or function space constructor. This result is a non-trivial extension of the classical characterisation result for labelled transition systems.

   We have also addressed the problem of developing a proof system for the verification of process specifications. Building on previous work for CCS we have presented a sound proof system for a Chocs sub-calculus not including restriction. We have obtained two completeness results: one for the full specification language using an infinitary system, and one for a special class of so-called *well-described* specifications using a finitary system.

2. V. Danos and L. Regnier have designed an evaluation scheme based on the *legality condition* of Asperti and Laneve which computes the *execution path* of a lambda-term.

   That 'B-algorithm' is shown to be yet another variation on the geometry of interaction theme, giving a simple proof of the equivalence between the geometry of interaction and the legality condition.

   Furthermore, the formulation of the B-algorithm suggests some natural optimisations (by *natural* we mean that these optimisations are easily proved correct). It appears that the optimised B-algorithm is isomorphical to a standard one for executing lambda-terms: *Krivine's machine* which is an environment based machine that performs weak head reduction. In other words, the link between an abstract theory of computation (the geometry of interaction) and a practical implementation of lambda-calculus (Krivine's machine) has been completely elucidated. This work is detailed in the paper "Reversible and irreversible computations" (`ftp`-accessible on `lmd.univ-mrs.fr`, in `/pub/regnier/revirrev.ps.[Z, gz]`).

   At the moment, V. Danos and L. Regnier work on further optimisations of the B-algorithm (following a suggestion from Asperti). Since the B-algorithm basically computes a path in a term, by moving in the graphical structure of the term, it is natural to memorize some particular shared subpaths (called *jumps*)

so that when the execution comes back on them, they are completed in one step. If jumps are chosen to be the so-called virtual redexes (or balanced legal paths in Asperti/Laneve terminology) this storing mechanism yields a new optimal implementation of lambda-calculus. This implementation seems quite different from Lamping's one and remains to be practically tested.

3. C. Lavatelli, in collaboration with G. Boudol and P.-L. Curien, has pursued the study of the semantics of lazy lambda calculus with resources λ_r, a non-deterministic language which provides explicit syntactical means to control the availability of arguments defined by Boudol.

   Two domain equations have been proposed to produce models of λ_r, both of which can be written $D = (\mathcal{M}(D) \to D)_\perp$. They differ by the construction $\mathcal{M}$ used; roughly, the points of $\mathcal{M}(D)$ are multisets of points of D. In the first construction $\mathcal{M}(D)$ is a powerdomain on the multisets of D. The second one defines $\mathcal{M}(D)$ as the domain-analogue of multisets. Accordingly, two interpretations of terms arise; the first one treats bags (the arguments of the calculus) as first-class entities, i.e. defines the meaning of parallel composition; the second one defines the denotation of terms only.

   Both equations have solutions in the category of prime algebraic lattices which constitute adequate models of the calculus. The canonical solution of the first one is a filter model (characterised by an intersection type system à la Coppo extended with a non-idempotent conjunction). As for the second equation, the type presentation of the concrete model is an intersection system where conjunction is non-idempotent.

   Full abstraction fails for the two models considered; the calculus of resources lacks a convergence testing facility. It is enough to add this facility as far as the second model is concerned; the proof follows the definability approach based on the existence of characteristic terms for each type. Instead, it seems that the first model requires also a kind of "duplicator" in the language to allow the definition of characteristic terms for idempotent conjunction types.

3.6 Imperial College, London

Research directions

The main focus of research done at Imperial College has been on Logics for Concurrency and λ-calculus.

The main goal of our work in the Logics for Concurrency area has been the transfer of ideas from the theory of functional computation to concurrency. Over the last three years, we have made considerable progress in our endeavour. The discovery of Interaction Categories was the starting point of our research in this area, and this has developed through the work on typed calculi of processes. We have been able to address specification and verification issues of typed concurrent programs using the syntax of the process calculi and using the semantic structure of various Interaction Categories. Work is also being done on modelling other formalisms for concurrency such as the π-calculus and Action Structures within the same setting. Continued understanding of the rôle of logical structures to model concurrent computation has also been a priority.

In the λ-calculi area, much work has been done in understanding the connections between traditional denotational semantics and game semantics. Work is in an advanced state in the area of the geometry of implementation, which uses ideas from Girard's Geometry of Interaction to develop efficient implementations of functional programs.

Personnel and exchanges

Personnel involved in CONFER at this site are Samson Abramsky, Simon Gay, Greg Meredith, Rajagopal Nagarajan and Duško Pavlović. François Lamarche and Ian Mackie who have left the college within the past year, have continued to participate actively in the project.

Two PhDs have been completed within the last year; and a third is nearing completion. The titles are as follows

- Linear Types for Communicating Processes
- The Geometry of Implementation
- Typed Concurrent Programs: Specification & Verification

All these topics are closely tied to the aims of the CONFER project and the dissertations were directed by Samson Abramsky. Greg Meredith has continued his PhD work under Abramsky's supervision and his work is also in the same realm.

Rajagopal Nagarajan visited Sophia-Antipolis for three weeks, under an exchange scheme between Imperial College and Ecole de Mines. He gave two talks during his visit and received useful feedback from the CONFER participants at the Sophia-Antipolis site.

Ian Mackie has moved and taken up a research position in the Semantics, Proof and Abstract Interpretation group at the Laboratoire d'Informatique (LIX) de l'École Polytechnique, but has continued to participate in the project. François Lamarche has left Imperial and has been an academic visitor at Genoa and Marseille.

We have had several interesting discussions with researchers at the University of Cambridge and University of Edinburgh regarding common areas of work. Our site members have actively participated in the CONFER workshops held in the last year, giving presentations, and have had fruitful exchange of ideas with other CONFER participants.

Publications

[1] S. Abramsky, S. J. Gay, and R. Nagarajan. Specification Structures and Propositions-as-Types for Concurrency. *Logics for Concurrency: Structure vs. Automata—Proceedings of the VIIIth Banff Higher Order Workshop* (G. Birtwistle and F. Moller, ed.). Springer-Verlag Lecture Notes in Computer Science, 1995.

[2] S. J. Gay and R. Nagarajan. A typed calculus of synchronous processes. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995.

[3] S. J. Gay. Linear types for communicating processes. PhD thesis. University of London, 1995.

[4] C. L. Hankin, I. C. Mackie, and R. Nagarajan, (ed). *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Theory and Formal Methods Section Workshop*, Cambridge, September 1994. IC Press, 1995.

[5] F. Lamarche. Generalizing coherence spaces and hypercoherences. In *Proceedings, Eleventh Conference on the Mathematical Foundations of Programming Semantics*. 1995.

[6] F. Lamarche. From Chu spaces to cpos. 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. IC Press, 1995.

[7] F. Lamarche. Games Semantics for Full Propositional Linear Logic. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995.

[8] I. C. Mackie. The geometry of implementation. PhD thesis. University of London, 1995.

[9] I. C. Mackie. The Geometry of Interaction Machine. In *Proceedings, 22nd ACM Symposium on Principles of Programming Languages (POPL)*. ACM Press, 1995.

[10] R. Nagarajan, (ed). *Proceedings of the Fifth CONFER Workshop*, Imperial College, London, October 1994. Imperial College Department of Computing Technical Report No. 95-6, February 1995.

[11] R. Nagarajan. Typed Concurrent Programs: Specification & Verification. PhD thesis. University of London. To appear.

[12] D. Pavlović. Convenient Categories of Processes and Simulations I: Modulo Strong Bisimilarity. In *Proceedings, Sixth Conference on Category Theory and Computer Science*. 1995.

3.7 INRIA-Rocquencourt

3.7.1 Research directions

INRIA Rocquencourt has worked in the following directions.

- Distributed Pict-like languages.

  To get a full distributed implementation of mobile processes, a first step is to add primitives for explicit control over distribution and for failure recovery to the Pict design. Fournet and Gonthier proposed a new process calculus, the *join-calculus*, as the formal basis for the distributed Pict language, which will be presented at POPL'96. A corresponding implementation is underway by Florent Guillaume and Luc Maranget. The final aim is to have a language more integrated to communication than ECRC Facile.

- Optimal reductions. Andrea Asperti and Cosimo Laneve mostly pursued their work on optimal reduction techniques. In [ACa], Lamping's graph reduction algorithm is extended from λ-calculus to arbitrary Interaction Systems. These higher order Rewriting Systems cover all usual syntactical constructs of declarative languages; in particular, [ACa] provided the theoretical foundation for a prototype implementation of a functional language based on Lamping's idea: the Bologna Optimal Higher-order Machine (BOHM).
  Since Lamping's work, many different encodings of λ-terms into sharing graphs have been proposed in the literature; the relations between all these translations are investigated in [ACb].
  The problem of accumulation of control operators and a first, partial solution based on the use of simplification rules for "safe" operators is addressed in [A].
  Finally, [L] contains a first attempt to generalise the theory and practice of optimal reduction to concurrent systems.

- Abstract reductions systems. Confluence, Finite developments, Standardisation, Stability have been studied and are presented in the dissertation of Melliès which should be defended at the end of 1995.

- Explicit substitutions. Thérèse Hardin with Gilles Dowek and Claude Kirchner worked on Higher-order unification via explicit substitutions. They reduce higher-order unification to first-order equational unification by replacing substitution by grafting. Unification in such a calculus can be performed by well-known algorithms such as narrowing. Huet's algorithm can be seen as a specific strategy in the new algorithm with explicit substitutions, each of its steps is decomposed in elementary ones, giving a more atomic description of the unification process. This work has been presented at LICS'95 [DKH95].

- Explicit substitutions and Functional Back-Ends of Compilers. Thérèse Hardin, Luc Maranget and Bruno Pagano defined a weak λ-calculus, $\lambda\sigma_w$, as a subsystem of the full λ-calculus with explicit substitutions $\lambda\sigma$. They study the connection with traditional back-ends.

- Damien Doligez and Georges Gonthier finished the correctness proof of a realistic and original concurrent garbage collector in Lamport's temporal logic of actions (TLA). This 8-month long work consisted in writing 20k lines of TLP, TLA interface for the Larch Prover LP, which was the only equational system available at that time at INRIA [Dol95]. Several bugs in the program and formal proof were found along this correctness proof.

3.7.2 Persons and exchanges

INRIA Rocquencourt participants are: Jean-Jacques Lévy, Damien Doligez, Cédric Fournet, Georges Gonthier, Florent Guillaume, Thérèse Hardin, Luc Maranget, Paul-André Melliès, Bruno Pagano and Didier Rémy. Part of the work is subcontracted to the University of Bologna: Andrea Asperti and Cosimo Laneve.

Benjamin Pierce and David Turner visited INRIA-Rocquencourt 4 days in April. They worked with the Rocquencourt group on issues related to a Pict distributed implementation. Roberto Amadio gave a seminar at our working group on the semantics of failures in a distributed setting with mobile processes.

Paul-André Melliès has moved to the Free University in Amsterdam for 16 months. He works with Klop and Van Raamsdonk. Cédric Fournet (a new PhD student) spent 2 months (August-September) in Cambridge with Benjamin Pierce. Cosimo Laneve also worked in collaboration with Gerard Boudol on the discriminating power of the λ-calculus with multiplicities [BL].

Damien Doligez defended his PhD, on the design, implementation and correction proofs of Concurrent Garbage Collector on May 5, 1995 [Dol95].

3.7.3 Perspectives, work in progress

The group in Rocquencourt is mainly working towards a distributed implementation for Pict-like languages.

In the background, work on lambda-calculus and term rewritings with bound variables is continued.

3.7.4 Publications, technical reports

[ACa] A. Asperti, C. Laneve. *Interaction Systems II: the practice of optimal reductions.* Theoretical Computer Science, to appear (expected in Vol.160, June 1996).

[ACb] A. Asperti, C. Laneve. *Relating λ-calculus translations in sharing graphs.* Proc. of the Second International Conference on Typed Lambda Calculi and Applications, TLCA'95, Edinburgh, Scotland. 1995.

[A] A. Asperti. *$\delta \circ !\epsilon = 1$: optimizing optimal λ-calculus implementations.* Proc. of the Sixth International Conference on Rewriting Techniques and Applications, RTA'95, Kaiserslautern, Germany. 1995.

[BL] G. Boudol, C. Laneve, *Termination, deadlock and divergence in the λ-calculus with multiplicities*. Proc. of the Eleventh Conference on the Mathematical Foundations of Programming Semantics, MFPS'95, Electronic Notes in Computer Science n. 1, pp. 63 – 76, Springer-Verlag, 1995.

[L] C. Laneve. *An Investigation on the Optimal Implementation of Processes*. Atti del V Convegno Italiano di Informatica Teorica, Ravello, 9-11 Novembre 1995.

[FG] C. Fournet, G. Gonthier, *The reflexive CHAM and the join-calculus*. To be presented at POPL'96, Jan 96, St Petersburgh.

[Dol95] Damien Doligez, *Conception, réalisation et certification d'un glaneur de cellules concurrent*, PhD thesis (in French), Université Paris 7, May 1995.

[DKH95] Gilles Dowek, Thérèse Hardin, Claude Kirchner, Higher-order unification via explicit substitutions, LICS'95, Jul 95, San Diego.

[M95] Paul-André Melliès, Typed λ-calculi with explicit substitutions may not terminate, TLCA'95, Apr 95, Edinburgh.

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. This year our work focused mainly on the π-calculus, the λ-calculus and their relationships. More specifically, the following achievements were obtained:

- regarding the λ-calculus with multiplicities, we have studied various observation scenarios, depending on the way divergence, convergence and deadlock are dealt with. The resulting observational semantics are characterised, using the model of Lévy-Longo trees. In particular, we establish that the observational semantics where divergence, convergence and deadlock are distinguished coincides with the semantics induced by the π-calculus encoding of Milner [1,2,3]. Progress has been made also on the denotational semantics of the λ-calculus with resources – a non-deterministic version of the λ-calculus with multiplicities. This is reported on in the LIENS Chapter, and in the Calculi area report

- a refinement of the π-calculus, where only private names are sent, has been defined and studied [6,7,9]. It is shown that the mathematical theory of this "internal mobility" is much simpler than the one of the full π-calculus. Nevertheless, the expressive power remains very high. In particular, agent passing can still be encoded using internal mobility.

The work done in Sophia is reported on in more details in the section on the Calculi area.

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 is now a researcher at the University of Bologna. He is still a member of CONFER, affiliated to the INRIA Rocquencourt site. Since October 1994, Ch. Retoré has a research position at INRIA Nancy Lorraine. He is no longer involved in the CONFER project.

R. Amadio moved from Nancy to Sophia Antipolis. He is now (since November 1994) working with the MEIJE project at INRIA, although, as far as CONFER is concerned, he is still attached to the ENS Paris site. D. Sangiorgi has a research position at INRIA Sophia Antipolis (since December 1994). To avoid modifications in the contract, his travelling for CONFER is still funded by the Edinburgh site.

We maintained our collaboration with the ENS group in Paris. C. Lavatelli is finishing her PhD Thesis under the joint supervision of P.-L. Curien and G. Boudol. She visited Sophia for one month.

3.8.3 Publications and research reports

[1] G. Boudol and C. Laneve: *The discriminating power of multiplicities in the λ-calculus*, INRIA Research Report 2441, December 1994, submitted for publication.
[2] G. Boudol and C. Laneve: *Termination, deadlock and divergence in the λ-calculus with multiplicities*, Proceedings of the 11th Conference on the Mathematical Foundations of Programming Semantics, Electronic Notes in Computer Science 1, 1995.
[3] G. Boudol and C. Laneve: *λ-Calculus, multiplicities and the π-calculus*, INRIA Research Report 2581, June 1995, submitted for publication.
[4] D. Sangiorgi: *Bisimulation in higher-order calculi*, INRIA Research Report 2508, 1995. Revised version of a paper appeared in the proceedings of the IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94), North Holland, 1994.
[5] D. Sangiorgi: *Lazy functions and mobile processes*, INRIA Research Report 2515, 1995, submitted for publication.
[6] D. Sangiorgi: πI: *A symmetric calculus based on internal mobility*, TAPSOFT'95, LNCS, 1995.
[7] D. Sangiorgi: *Internal mobility and agent passing calculi*, ICALP'95, LNCS, 1995.
[8] D. Sangiorgi: *On proof method for bisimulation*, Proceedings of the Conference on the Mathematical Foundations of Computer Science, LNCS, 1995. A revised version of the technical report ECS-LFCS-94-299 has been submitted for publication.
[9] D. Sangiorgi: *π-Calculus, internal mobility and agent-passing calculi*, INRIA Research Report 2539, 1995, submitted for publication.

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*, *Foundational Models and Abstract Machines* and *Logics for Concurrency*.

- *Calculi*

  Research has continued on developing the π-calculus with explicit substitutions, the $\pi\xi$-calculus. Weak bisimulation semantics for the π-calculus have been analysed (and axiomatised) in terms of ordinary *Observation* equivalence over the transition systems of the $\pi\xi$-calculus. A prototype version of a semantic-based verification environment for the π-calculus, based on the semantics model of the $\pi\xi$-calculus, has been also developed.

  A novel and efficient technique has been developed for checking strong (early and late) bisimulation equivalence of the π-calculus without matching. The proposed procedure improves the complexity bound for checking bisimilarity of finitary agents (an agent is finitary if the degree of parallelism of all its reachable agents is bound).

- *Foundational Models and Abstract Machines*

  A generalisation of Larsen and Xinxin *context systems* in terms of *double-categories* has been proposed. The adequateness properties of context systems naturally follow from the categorical models, and moreover interesting SOS specifications which are not expressible as context systems, can be described in the categorical formulation.

- *Logics for Concurrency*

  A new paradigm for typing concurrent processes, called *behaviours-as-types*, has been proposed. In this paradigm, types are used to convey information about the *behaviour* of processes: terms correspond to processes, types correspond to behaviours. Moreover, two terms having the same type denote two processes which behave in the same way, that is, they are *bisimilar*. A sound and complete compositional typing system for a simple process calculus has been also developed.

3.9.2 Perspective

We are interested in further developing theoretical models for mobile process calculi and the verification techniques based on these models.

Bisimulations for the π-calculus are instances of a larger class of bisimulation semantics for processes calculi, called *history dependent*. This concept can be found in various semantics for process calculi, e.g. locality or causality. One aspect that we intend to study is the development of verification methods for checking history dependent bisimulation semantics.

Moreover, we intend to investigate notions of types which ensure only authorised accesses to information. The idea is to use notions of types which give information on the behaviours of processes together with their interface with other processes: *behaviours-as-types* (e.g. the modal μ-types).

3.9.3 Persons

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

Two Phd-Theses are nearing completion, that of Fabio Gadducci and Paola Quaglia, both under the supervision of Ugo Montanari. The topics of the theses are closely related to the aims of the CONFER project.

3.9.4 Interactions with other CONFER Sites

Marco Pistore visited INRIA Sophia-Antipolis (May 95) to discuss the issues related to the efficient verification of open bisimulation.

3.9.5 Publications, Technical Reports

1. Ferrari, G., Montanari, U., Quaglia, P., The Weak Late π-calculus Semantics as Observation Equivalence, to appear Proc. CONCUR'95, LNCS, 1995.

2. Ferrari, G., Modoni, G., Quaglia, P., Towards a Semantic-based Verification Environment for the π-calculus, To appear Fifth Italian Conference on Theoretical Computer Science, 1995.

3. Gadducci, F., Montanari, U., SOS Contexts as Cells in Double-Categories, submitted for publication, 1995.

4. Gadducci, F., Miculan, M., Modal μ-types for Processes, Proc. LICS'95, 1995.

5. Montanari, U., Pistore, M., Checking Bisimilarity for Finitary π-calculus, to appear Proc. CONCUR'95, LNCS, 1995.

Deliverables 1, 2 and 5 relate to the *Calculi* area. Deliverable 3 relates to the *Foundational Models and Abstract Machines* area and deliverable 4 relates the *Logic for Concurrency* area.

3.9.6 Description of Technical Contributions

1. We show that the Weak Late π-calculus semantics can be characterised as ordinary Observation congruence over a specialised transition system where both the instantiation of input placeholders and the name substitutions, due e.g. to communication, are explicitly handled via suitable constructors. The approach presented here allows to axiomatise the Weak Late semantics by simply adding

Milner's τ-laws to the proof system for the Strong equivalence. Resorting to Observation equivalence provides a framework which is general enough to allow to recover, in straightforward ways, other bisimulation semantics (Early, both Strong and Weak, and Dynamic and Branching, both Early and Late).

2. A prototype version of a semantic-based verification environment for manipulating and analysing mobile systems specified in the π-calculus is presented. In the current version, the π-environment provides two main facilities: a π-calculus interpreter equipped with a graphical interface, and a verification tool which is used to decide (strong and weak) *early* and *late* bisimulation equivalences for finite π-calculus processes. The π-environment is built on top of existing verification systems, and the re-use of software modules, based on semantical consideration, is the key feature of our proposal.

3. *Context systems* (briefly, CS's) were introduced by Kim Larsen and Liu Xinxin as a framework for developing a verification methodology for concurrent systems. At the same time, they represent an alternative presentation of the operational semantics of process algebras: they do not follow the inductive nature of the standard *structural operational semantics* approach, but they are proved to be "adequate" (i.e., behaviour-preserving) whenever suitable restrictions on the format of the SOS rules are satisfied. A generalisation of CS's, relying on *double-categories*, is presented. We illustrate how to recast all the context constructions in categorical terms, preserving the adequateness properties of CS's, and showing furthermore that there are interesting SOS specifications not expressible as context systems, which can be described in our framework.

4. We introduce a new paradigm for concurrency, called *behaviours-as-types*. In this paradigm, types are used to convey information about the *behaviour* of processes: terms correspond to processes, types correspond to behaviours.

   We apply this paradigm to Winskel's Process Algebra. Its types are similar to Kozen's modal μ-calculus; hence, they are called *modal μ-types*. We prove that two terms having the same type denote two processes which behave in the same way, that is, they are *bisimilar*. We give a sound and complete compositional typing system for this language. Such a system naturally recovers the notion of bisimulation also on open terms, allowing us to deal in a compositional manner with processes with undefined parts.

5. We associate to every π-calculus agent an *irredundant unfolding*, i.e., a labeled transition system equipped with the ordinary notion of strong bisimilarity, so that agents are mapped into strongly bisimilar unfoldings if and only if they are early strongly bisimilar. For a class of finitary agents (that strictly contains the finite control agents) without matching, the corresponding unfoldings are finite and can be built efficiently. The main consequence of the results presented in the paper is that the irredundant unfolding can be constructed also for a single

agent, and then a minimal realisation can be derived from it employing the ordinary partition refinement algorithm. Instead, according to previous results only *pairs* of π-calculus agents could be unfolded and tested for bisimilarity, and no minimisation of a single agent was possible. Another consequence is the improvement of the complexity bound for checking bisimilarity of finitary agents without matching.

3.9.7 Updates of Deliverables PPR2

1. Ferrari, G., Montanari, U., Quaglia, P., A π-calculus with Explicit Substitution. To appear in Theoretical Computer Science.

2. Ferrari, G., Montanari, U., Type Additive Concurrency, (A revised version entitled "Dynamic Matrices and the Cost Analysis of Concurrent Programs" appeared in Proc. AMAST'95, LNCS 936, 1995).

3. Corradini, A., Gadducci F., Montanari, U., Relating two categorical models of term rewriting, in Proc. Rewriting Techniques and Applications, (RTA'95), LNCS 914, 1995.

4. Montanari, U., Pistore, M., Concurrent Semantics for the π-calculus, in Proc. MFPS'95, ENTCS 1, 1995.

3.10 SICS

3.10.1 Research directions

During the third year of CONFER, work at SICS has mainly been conducted on Programming Languages and on Calculi. In the former, Björn Lisper has extended his work on correct unfolding of nondeterministic programs into a general theory for reduction strategies and program transformations of programs executing in nondeterministic environments [L95]. In the latter, Björn Victor and Lars-Henrik Eriksson have developed the Mobility Workbench by integrating model checking code by Mads Dam. A sort inference algorithm based on [Gay:sortinf] has also been implemented.

Within the areas Foundations and Calculi, Joachim Parrow has updated his contributions to the second year deliverable (on axiomatisations and interaction diagrams) for journal publication [P95,PS95].

3.10.2 Persons and exchanges

During the third year at SICS, Björn Lisper, Faron Moller, Joachim Parrow, Björn Victor, Lars-Henrik Eriksson and Torkel Franzén have been active on CONFER. The continued collaboration between Joachim Parrow and Davide Sangiorgi (Edinburgh) on axiomatisations has resulted in a publication [PS95]. Another connection has been to CWI, where Björn Lisper has benefited from contacts with the CWI expertise on higher-order rewriting. We have participated in the CONFER workshops.

3.10.3 Perspectives, work in progress

Future work on the Mobility Workbench will include new algorithms for equivalence checking, based on traditional partitioning algorithms; checking of early and late equivalences; adaption of model checking to characterise open and weak equivalences.

The theory for programs executing in nondeterministic environments can support the design of nondeterministic programming languages with particularly appealing semantical properties. An interesting possibility is to design and implement lazy languages with process communication primitives.

3.10.4 Publications, technical reports

L95 B. Lisper. Computing in Unpredictable Environments: Semantics, Reduction Strategies, and Program Transformations. Research Report, Swedish Institute of Computer Science (SICS), Kista (1995). (submitted)

P95 J. Parrow. Interaction diagrams. Accepted for publication in *Nordic Journal of Computing* (1995).

PS95 J. Parrow and D. Sangiorgi: Algebraic Theories for Name-Passing Calculi. *Information and Computation* **120**(2):174–197 (1995).

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 model checking based on a variant of the modal logic of [Dam93].

[Dam93] M. Dam. *Model Checking Mobile Processes.* CONCUR'93, LNCS, pp22-36. Full version in Research Report R94:01, SICS.

[Gay:sortinf] S. J. Gay. *A Sort Inference Algorithm for the Polyadic π-Calculus*, Proceedings, 20th ACM Symposium on Principles of Programming Languages, 1993.

Chapter 4

Deliverables

4.1 Workshop 5

At Imperial College in London, with 35 participants.

Tuesday, October 4

12.00-12.45	Registration
12.45-14.15	Lunch
14.14-14.30	Welcome and Opening Remarks, Rajagopal Nagarajan, Jean-Jacques Lévy
14.30-15.30	Concurrent Semantics for the $\pi-$calculus, Ugo Montanari
15.300-16.00	Efficient Function Transmission in FACILE Fritz Knabe
16.00-16.30	Tea
16.30-17.00	Algebraic Tools for Equivalence fo Transition Systems, Pasquale Malacaria

Wednesday, October 5

09.30-10.15	The Discriminating Power of Multiplicities, Cosimo Laneve
10.15-11.00	On the Semantics for the λ-calculus with Resources, Carolina Lavatelli
11.00-11.30	Coffee
11.30-12.00	Action Semantics for Concurrency, Peter Mosses
12.00-13.00	Internal Mobility and the $\pi-$calculus, Davide Sangiorgi
13.00-14.30	Lunch
14.30-15.15	Interaction Combinators, Yves Lafont
15.15-16.00	The Geometry of Interaction Machine II: Optimisations, Ian Mackie
16.00-16.30	Tea
16.30-17.15	A Denotational Semantics for the π-calculus, Greg Meredith
17.15-18.00	From CHU Spaces to CPO's, François Lamarche
19.30	Dinner

Thursday, October 6

09.15-09.45	A Hennessy-Milner Logic for Higher-Order Bisimulation, Roberto Amadio
09.45-10.30	Relating Two Categorical Models of Term Rewritings Fabio Gaducci
10.30-11.00	Optimal Rewriting of Proof Nets Representing Lambda Terms Femke van Raamsdonk
11.00-11.30	Coffee
11.30-12.00	Closed Action Calculi Philippa Gardner
12.00-13.00	Concurrent Objects, Benjamin Pierce
13.00-14.30	Lunch
14.30-16.00	Preparing for the Review & Future Prospects (Part II)
16.00-16.30	Tea
16.30-17.15	Interaction Diagrams Joachim Parrow
17.15-17.45	An Abstract Strong Normalisation Theorem, Paul-André Melliès

Friday, October 7
REVIEW DAY

4.2 Workshop 6

At ENS in Paris with 35 participants.

Monday, April 10

08.45-09.15	Registration
09.00-10.00	On the bisimulation proof method, Davide Sangiorgi
10.00-10.30	Operational Equivalence in Action Calculi – A Progress Report, Ole Jensen
10.30-11.00	Coffee Break
12.45-14.00	Lunch
14.00-15.00	Specification structures and Propositions-as-types for Concurrency, Samson Abramsky
15.00-15.30	Using TLP for proving concurrent programs Georges Gonthier
15.30-16.00	Tea Break
16.00-16.45	Modal μ-Types for Processes, Fabio Gadducci
16.45-17.30	Elementary and fibrational control structures, John Power
17.30-18.00	Operational Equivalence in Action Calculi – A Progress Report, Ole Jensen

Tuesday, April 11

09.00-10.00	GOI and environment machines, Vincent Danos
10.00-10.30	Some empirical results in optimal beta-reducers, Andrea Asperti
10.30-11.00	Coffee Break
11.00-11.45	Eliminating spurious control nodes in optimal lambda-graph reduction, Stefano Guerrini
11.45-12.30	Convenient categories of processes and simulations, Dusko Pavlovic
12.30-14.30	Lunch
14.30-15.30	Towards a distributed implementation for Pict, Cedric Fournet
15.30-16.30	Business meeting

Wednesday, April 12

09.00-09.45	A Synchronous Interaction Category with Value Passing, Lindsay Errington
09.45-10.30	Checking Bisimilarity for Finitary π-calculus, Marco Pistore
10.30-11.00	Coffee Break
11.00-11.45	Towards a Model of the π-calculus, L. Greg Meredith
11.45-12.30	The Weak Late π-calculus Semantics as Observation Equivalence, Paola Quaglia
12.30-14.00	Lunch

4.3 Software deliverables

Several pieces of software have been constructed during the three years 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 three 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. Crégut, 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 analysing 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.

4.3.1 Facile Antigua Release

The Facile Antigua Release is subject to a written licence agreement. The software is free of charge for research and educational purposes. To obtain the Facile Antigua Release ftp a copy of the licence agreement from ftp.ecrc.de (141.1.1.1). Log in as "anonymous" and copy the file "/pub/magic/license.ps.Z". Return two signed copies and an email address to:

Facile Project – Antigua
ECRC GmbH
Arabellastrasse 17
81925 Munich, Germany

When we receive the signed licence agreements we will instruct you how to obtain the software via ftp. If you are interested in learning more about the Facile Antigua Release we suggest you start by reading:

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.

This report and other ECRC technical reports can be obtained via ftp from ftp.ecrc.de (141.1.1.1). Log in as "anonymous". The report ECRC-93-20 is stored under the directory "pub/ECRC_tech_reports/reports" as ECRC-93.20.ps.Z. There is a README file (in directory "pub/ECRC_tech_reports") containing some further information.

Further information on Facile may be found using the World Wide Web:
`http://www.ecrc.de/facile/facile_home.html`

4.3.2 BOHM

BOHM is available at the CONFER ftp site
`ftp://theory.doc.ic.ac.uk/theory/CONFER/impl/Asperti/`

See the Programming Languages section of this report for more information.

4.3.3 Lilac

Lilac is available at the CONFER ftp site
`ftp://theory.doc.ic.ac.uk/theory/CONFER/impl/Mackie/lilac/`

See the Programming Languages section of PPR-2 for more information.

4.3.4 Pict

Pict is available at the CONFER ftp site
`ftp://theory.doc.ic.ac.uk/theory/CONFER/impl/Turner/`

See the Programming Languages section of this report for more information.

4.3.5 Portable, unobtrusive garbage collection for multiprocessor systems

The mono processor version of the garbage collector is available inside the Caml-light system `http://pauillac.inria.fr/caml/`. The multi processor, existing for Encore, Sequent and KSR implementations of Caml-light, will be released soon inside the Concurrent Caml-light system.

4.3.6 The Mobility Workbench

The Mobility Workbench (MWB) is a tool for manipulating and analysing mobile concurrent systems described in the π-calculus, developed in cooperation with BRA Concur2. The two basic functionalities of the MWB are to decide the open bisimulation equivalence of [**?**], and to model check formulas in a variant of the modal logic of [Dam93].

The equivalence checking algorithm has proven to be efficient enough to make the MWB useful in postgraduate education, while the model checking algorithm is still very inefficient. Work is currently being done on improving this algorithm, as well as improving the data representation to make the overall performance of the MWB better. Sort checking and sort inference, based on [Gay:sortinf], is also currently being implemented.

Future work will include new algorithms for equivalence checking, based on traditional partitioning algorithms; checking of early and late equivalences; adaption of model checking to characterise open and weak equivalences.

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 has 3 parts. First, we study the abstraction of interaction, ie Milner's action structures. It is also work by Gadducci and Montanari. Secondly, we treat calculi with bound variables in many ways. Little is known in the 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. Thirdly, 2 models are presented for operational and denotational semantics of CHOCS and Facile.

5.1.1 The Theory of Action structures

5.1.1.1 Action structures

Action structures have previously been proposed as an algebra for both the syntax and the semantics of interactive computation. Work on action structures has continued in Cambridge and Edinburgh.

In 1993-4, the main advance was in defining action calculi, which are action structures of a concrete nature, with added structure, and which represent a wide variety of calculi of computation – functional, sequential, and concurrent. Each action in an action calculus is represented as an assembly of molecules; the syntactic binding of names is the means by which molecules are bound together. One action calculus differs from another only in its generators, called controls.

In 1994-5 the emphasis has been on

a) refining the presentation of action calculi;

b) investigating a natural variant: closed or name-free action calculi;

c) defining the category of their models.

The refinement of their presentation is in [Mil96]; the paper presents several examples (e.g., typed lambda calculus, and Petri nets parametrised on their places and transitions) and uses a new notion, action graphs, in doing so. This presentation is significant because it makes possible links with graph reduction for the lambda calculus, and also with graph-rewriting which is a highly developed research topic.

Moroever, in [Mil96] an equational characterisation of action calculi is given: each action calculus A is the quotient of a term algebra by certain equations. The terms are generated by a set of operators, including those basic to all action structures as well as the controls specific to A; the equations are the basic axioms of action structures together with four additional axiom schemata.

The analysis of closed action calculi, name-free account of action calculi, appears in [Gar95] and [GarMil95]. The analysis shows that although names play an important presentational role in actions calculi, they are in fact not essential. This development promises to establish connections with the mainstream of semantic theory, in which name-free objects are found more tractable. Closed action calculi are introduced in [Gar95]. The work in [GarMil95] establishes an exact correspondence between the equational theories of action calculi [Mil96] and name-free action calculi [Gar95].

5.1.1.2 Control structures

The model theory of action calculi, so-called *control structures*, appears in [MiMiPo95] and [HePo95]. The first advance was to show that the equational theory of action calculi can be purified; the equations become the axioms which characterise the category of control structures, as action structures with added structure. Importantly, each action calculus is initial in the appropriate subcategory of control structures; it is this which entitles us to say that control structures are the models of action calculi. The paper contains the technical formulation of the category of control structures and their homomorphisms, and the proof of initiality. 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.

In [HePo95], it is shown that control structures have a very succinct categorical formulation which is in some ways more convincing that their equational characterisation. This category theoretic reformulation of control structures avoids explicit reference to names. The basis of the formulation is *a binding structure*, which accounts for naming and the associated operation of binding in isolation, i.e., without reference to extra features. Upon adding structure to such a binding structure a fibrational control structures is obtained, which (with a mild extra condition) is showed equivalent to locally finite control structures, those in which every action has a finite surface.

5.1.1.3 Contexts as Cells in Double Categories

A line of research with analogies with that in control structures is being pursued in Pisa. The basic idea is to define the syntax and semantics of process calculi using *double*

categories. The dynamics (i.e. the operational semantics) of the calculi is described through the double-category notion of *cell*. This makes available a certain number of powerful semantic tools. As a first example, in [GaMo95], Kim Larsen and Liu Xinxin's *Context Systems* are studied. At present, work is being carried on in applying this categorical approach to the π-calculus.

Context systems were introduced by Larsen and Liu as a framework for developing a verification methodology for concurrent systems. At the same time, they provide an alternative mechanism to describe the operational semantics of process calculi : they do not follow the inductive nature of the standard *structural operational semantics* approach, but they are proved to be "adequate" (i.e., behaviour-preserving) whenever suitable restrictions on the format of the SOS rules are satisfied. The paper [GaMo95] relies on cells to propose a generalisation of context systems. It is proved that all the context constructions naturally derive from the categorical formulation, i.e. the axioms of double-categories. Moreover, there are interesting SOS specifications not expressible as context systems, which can be described in the categorical formulation.

5.1.1.4 References

[Mil96] Robin Milner, Calculi for Interaction, To appear in Acta Informatica, 1996.

[Gar95] Philippa Gardner, A Name-free Account of Action Calculi, Proceedings of Mathematical Foundations of Programming Semantics, New Orleans, 1995.

[GarMil95] Philippa Gardner and Robin Milner, A short note on Closed Action Calculi, DRAFT.

[MiMiPo95] Alex Mifsud, Robin Milner and John Power, Control Structures, Proceedings of IEEE Conference on Logics in Computer Science, San Diego, 1995.

[HePo95] Claudio Hermida and John Power, Fibrational control structures, Proceedings of CONFER Conference, 1995.

[GaMo95] Gadducci, F., Montanari, U., SOS Contexts as Cells in Double-Categories, submitted for publication, 1995.

5.1.2 Calculi with bound variables

5.1.2.1 Basic Properties

Calculi with bound variables still provide many interesting problems with their basic properties: confluence, normalisation and termination.

Higher-order Rewriting

In several calculi featuring in the CONFER project the property of termination (strong normalisation) is important. Van Raamsdonk has made a study of normalisation in the framework of classical λ-calculus in cooperation with P. Severi (Technical University

Eindhoven), arriving to new proofs of fundamental lemma's such as the finite developments lemma, the strong normalisation theorem of simply typed lambda calculus, and related results. Current work is concerned with obtaining the normalisation property for outermost reduction in (weakly) orthogonal higher-order rewriting systems. For instance, for λ-calculus with β- and η-reduction it does not seem to be established yet that parallel-outermost evaluation is normalising.

Term Rewriting and λ-calculus with Explicit Recursion

This subject, pursued by J.W. Klop in cooperation with Z.M. Ariola (University of Oregon), is concerned with term graph rewriting both for first-order rewrite systems and for λ-calculus. We consider cyclic term graphs and cyclic λ-graphs. The aim is to design suitable transformation (rewrite) rules for such graphs, such that confluence holds. Especially for the lambda calculus case it is a delicate matter how to treat free and bound variables. The search for a fully satisfactory system has not yet ended. Present concern is with systems of nested recursion equations; an interesting feature is that there is a certain similarity with λ-calculi with explicit substitution.

To serve as semantics for λ-calculus with explicit recursion, J.W. Klop has worked in infinitary λ-calculus in cooperation with J.R. Kennaway and M.R. Sleep (both University of East-Anglia) and F.J. de Vries (Hitachi, Japan). This work is concerned with λ-terms that may be infinitely deep (like the well-known Böhm trees), while reduction may have transfinite length.

Abstract Rewriting Systems

For 3 years, Paul-André Melliès has studied *Abstract Rewriting Systems* which are an axiomatic presentation of rewriting systems with bound variables. This important work suggests the elementary properties needed for confluency and termination. Results on standardisation, finite developments and strong normalisation are collected in his dissertation [Mb95]. A thorough treatment of termination led to the discovery of a counterexample for termination of first-order typed explicit substitution [M95], reported in last year report. Axiomatic versions for the standardisation and finite development theorems for λ-calculus-like languages may be found in his dissertation. These results were presented at LICS'92 in San Diego, at the Amsterdam Confer workshop 94 and at TLCA'95 in Edinburgh.

5.1.2.2 Optimal Reductions

Andrea Asperti and Cosimo Laneve mostly pursued their work on optimal reduction techniques. In [ACa], Lamping's graph reduction algorithm is extended from λ-calculus to arbitrary Interaction Systems. These higher order Rewriting Systems cover all usual syntactical constructs of declarative languages; in particular, [ACa] provided the theoretical foundation for a prototype implementation of a functional language based on Lamping's idea: the Bologna Optimal Higher-order Machine (BOHM).
Since Lamping's work, many different encodings of λ-terms into sharing graphs have

been proposed in the literature; the relations between all these translations are investigated in [ACb].
The problem of accumulation of control operators and a first, partial solution based on the use of simplification rules for "safe" operators is addressed in [A].
Finally, [L] contains a first attempt to generalise the theory and practice of optimal reduction to concurrent systems.
Cosimo Laneve also worked in collaboration with Gerard Boudol on the discriminating power of the λ-calculus with multiplicities [BL].

5.1.2.3 Explicit Substitutions

Explicit substitutions is an alternative way of looking at calculi with bound variables. Often, it uses de Bruijn's notation to avoid binders. Inside CONFER, explicit substitutions have been reported at several places: termination, model for functional back-end of compilers in the programming languages area, and higher order unification.

Higher-order unification via explicit substitutions

This work is by Thérèse Hardin with Gilles Dowek (INRIA) and Claude Kirchner (INRIA Lorraine & CRIN).

Higher-order unification is equational unification for $\beta\eta$-conversion but it is not a first-order process as substitution has to avoid capture. Thus the methods for equational unification (such as narrowing) built upon grafting (i.e. substitution without renaming), cannot be used for higher-order unification, which needs specific algorithms. We reduce higher-order unification to first-order equational unification by replacing substitution by grafting. This has to be done carefully as this replacement raises two major problems. First, some unification problems have solutions with grafting but no solution with substitution. Then equational unification algorithms rest upon the fact that grafting and reduction commute. But grafting and $\beta\eta$-reduction do not commute in λ-calculus and reducing an equation may change the set of its solutions. This difficulty comes from the interaction between the $\beta\eta$-conversion and the instantiation by the unification process, which have to share the same set of variables. But, this set is in fact partitioned into unification variables and reduction ones. So, we have to separate these two kinds of variables and to make reduction and grafting commuting, which needs to delay substitutions induced by a reduction. So we use the $\lambda\sigma$-calculus of explicit substitutions.

Unification in such a calculus can be performed by well-known algorithms such as narrowing, but we present a specialised algorithm for a greater efficiency. Then, we show how to relate unification in λ-calculus and equational unification in $\lambda\sigma$. Thus we describe a new higher-order unification algorithm which eliminates some burdens of the previous ones, particularly the functional handling of scopes. Huet's algorithm can be seen as a specific strategy for our algorithm, each of its steps is decomposed in elementary ones, giving a more atomic description of the unification process.

This work has been presented at LICS'95 [DHK95]

5.1.2.4 References

[AK] Z.M. Ariola, J.W. Klop, *Equational term graph rewriting.* CWI Report CS-R9552, July 1995, (short version also to appear in Fundamenta Informaticae 1995)

[ACa] A. Asperti, C. Laneve. *Interaction Systems II: the practice of optimal reductions.* Theoretical Computer Science, to appear (expected in Vol.160, June 1996).

[ACb] A. Asperti, C. Laneve. *Relating λ-calculus translations in sharing graphs.* Proc. of the Second International Conference on Typed Lambda Calculi and Applications, TLCA'95, Edinburgh, Scotland. 1995.

[A] A. Asperti. *$\delta \circ ! \epsilon = 1$: optimizing optimal λ-calculus implementations.* Proc. of the Sixth International Conference on Rewriting Techniques and Applications, RTA'95, Kaiserslautern, Germany. 1995.

[BL] G. Boudol, C. Laneve, *Termination, deadlock and divergence in the λ-calculus with multiplicities.* Proc. of the Eleventh Conference on the Mathematical Foundations of Programming Semantics, MFPS'95, Electronic Notes in Computer Science n. 1, pp. 63 – 76, Springer-Verlag, 1995.

[DHK95] Gilles Dowek, Thérèse Hardin, Claude Kirchner, *Higher-order unification via explicit substitutions*, LICS'95, Jul 95, San Diego.

[KKSV] J.R. Kennaway, J.W. Klop, M.R. Sleep, F.-J. de Vries, *Infinitary lambda calculus*, (published in RTA-95 proceedings as 'Infinitary lambda calculi and Böhm models', Kaiserslautern, April 1995, Springer LNCS 914, p.257-270, ed. Jieh Hsiang), CWI Report CS-R9535, May 1995

[L] C. Laneve. *An Investigation on the Optimal Implementation of Processes.* Atti del V Convegno Italiano di Informatica Teorica, Ravello, 9-11 Novembre 1995.

[Mb95] P.-A. Melliès, Dissertation, `ftp://pauillac.inria.fr/ mellies/wwwmain.ps.gz` (In French)

[M95] P.-A. Melliès, *Typed λ-calculi with explicit substitutions may not terminate*, TLCA'95, Apr 95, Edinburgh.

[OR] V. van Oostrom, F. van Raamsdonk, *Weak orthogonality implies confluence: the higher-order case*, CWI Report CS-R9501, January 1995, (short version already appeared in proceedings of LFCS '94, Springer LNCS 813, p.379-392)

[RS] F. van Raamsdonk, P. Severi, *On Normalisation*, CWI Report CS-R9545, June 1995.

5.1.3 Models of CHOCS and Facile

5.1.3.1 Chemical Machines for Facile

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). The results of the two papers have been merged and accepted for publication in the Journal of Formal Aspects of Computing, 1995. A short version (ECRC/M3/R10) will be considered a joint CONFER/LOMAPS deliverable and the long version (ECRC/M3/R11) will be considered a CONFER deliverable.

In this paper, we use the chemical abstract machine (CHAM) framework by Berry and Boudol for discussing various semantics for the Facile programming language and for formalising (parts of) its implementations. We use these formal descriptions to argue (informally) about implementability and cost of implementation in terms of low level machinery needed to implement the given semantics.
We take the Facile language as source for discussion, but the results also apply to several other new languages such as CML and Poly/ML. Characteristic for all these languages is that they combine ideas from the λ-calculus and process algebra, such as CCS, to support high level constructs for programming concurrent, parallel and/or distributed systems.

5.1.3.2 Denotational Semantics for CHOCS

A paper (ECRC/M3/R12) by Bent Thomsen and Samson Abramsky on the denotational semantics for CHOCS has been accepted for publication in the Journal of Theoretical Computer Science (TCS). The paper is currently under revision.

In this paper we study the Calculus of Higher Order Communicating Systems (CHOCS) in a denotational setting. We present a construction of a denotational semantics for CHOCS which resides in a domain constructed using the standard constructions of separated sum, Cartesian product, the Plotkin Power Domain constructor and recursively defined domains. We show, under mild restrictions, that the denotational semantics and the operational semantics of CHOCS are fully abstract. We have previously proved using bisimulation arguments that processes as first class objects are powerful enough to simulate recursion. However, the proof is very long and tedious. To demonstrate the power of the denotational approach we use it to obtain a very simple proof of the simulation of recursion result.

5.1.3.3 References

[ECRC/M3/R10] Leth, L. and Thomsen, B.: "Some Facile Chemistry (Summary)", Accepted for publication in Formal Aspects of Computing, 1995.

[ECRC/M3/R11] Leth, L. and Thomsen, B.: "Some Facile Chemistry", Accepted for electronic publication in Formal Aspects of Computing, 1995.

[ECRC/M3/R12] Thomsen, B. and Abramsky, S.: "A Fully Abstract Denotational Semantics for the Calculus of Higher Order Communicating Systems", Accepted for publication in TCS, 1994.

5.2 Calculi

This section reports on the work of the third year of CONFER in the area of Calculi. Some work done during the last years in this area has been finalised, resulting in conference papers and articles accepted for publication in technical journals. These are listed in the references below, but we do not report on this again. The work done this year still mainly deals with the semantics of the π-calculus and other related formalisms. To organise the presentation, we distinguish two main topics: *bisimulations*, and related methods for proving and verifying properties of processes, and *relating calculi*, which mainly means relating various calculi to the π-calculus. We refer to the previous Periodic Progress Report for a general presentation of these topics.

5.2.1 Bisimulations

This topic is maturing, and this year's work here is more oriented towards implementation questions than "fundamental" ones, regarding the very concept of bisimulation.

The proof technique provided by the notion of *open bisimulation*, proposed by Sangiorgi in the first year of CONFER, has been implemented in the Mobility Workbench. This implementation is used in postgraduate education. More details on this may be found in the Software Deliverables section.

In the same vein, a prototype version of a semantic-based verification environment for manipulating and analysing mobile systems specified in the π-calculus has been developed in Pisa [FMdQ]. In the current version, the π-environment provides two main facilities: a π-calculus interpreter equipped with a graphical interface, and a verification tool which is used to decide (strong and weak) *early* and *late* bisimulation equivalences for finite π-calculus processes. The π-environment is built on top of existing verification systems; the reuse of software modules, based on semantical considerations, is a key feature.

The (difficult) problem of effective bisimilarity checking for π-calculus is addressed in [MP]. A specific problem with the π-calculus is that the behavioural model associated with a π-term is usually infinite. Typically, an input prefix is interpreted, in the "early" approach, as an infinitely branching transition system. There is one input action for each name that can be received. In the "late" approach, there is a single transition from an input prefix, but the resulting agent is an abstraction, that is an infinite object. However, not all instantiations are relevant. Most often, considering a finite set of received names should be enough for bisimulation checking.

Montanari and Pistore define in [MP] an interpretation of π-terms by means of labelled transition systems, which they call *irredundant unfoldings*, which have the property that two terms are early strongly bisimilar if and only if their interpretations are strongly bisimilar, in the ordinary sense. Moreover, for a class of finitary agents without matching, which strictly contains the finite control agents (without matching), the corresponding unfoldings are finite and can be built efficiently. Since the irredundant unfolding is constructed for a single agent, a minimal realisation can be built out of it, using the ordinary partition refinement algorithm. This improves upon previous

work, where no minimisation of a single agent was possible, and only "on the fly" comparison of two agents was supported. The complexity bound for checking bisimilarity of finitary agents without matching is also improved by using Montanari and Pistore's technique.

Other theoretical work on bisimulations has also been pursued. In [FMnQ] Ferrari, Montanari and Quaglia consider a refinement of the π-calculus, the $\pi\xi$-calculus, where both the instantiation of input variables and the name substitutions, arising from to communications, are explicitly handled. The result is that the weak late π-calculus semantics can be characterised as ordinary observation congruence over the labelled $\pi\xi$-transition system. The authors also investigate an equational characterisation of the weak late bisimilarity: an equational system is obtained by adding Milner's τ-laws to the proof system of strong late bisimilarity. Other weak bisimulation semantics (e.g. dynamic and branching, both early and late) and their equational characterisations are obtained by considering the corresponding τ-laws.

The PhD Thesis by Sewell also investigates equational theories for bisimilarity, though in a different setting: Sewell's PhD Thesis addresses the problem of (finite) axiomatisability for strong bisimulation over finite state processes, those given by the signature consisting of prefix, summation and recursion. In order to express an interesting class of equational axioms, these simple processes are embedded into a simply typed lambda calculus, allowing equation schemes with metasubstitutions to be expressed by pure equations. This gives a fragment of the higher order π-calculus. Two equivalences over the lambda terms are defined, an extensional equality and a higher order bisimulation. Their restrictions to first order variables are shown to coincide, and therefore one can conclude that no finite equational axiomatisation of strong bisimulation can exist. The Basic Process Algebra with iteration and zero (BPA^*_δ) can be encoded into this lambda calculus. It is then proved that strong bisimulation over this algebra is not finitely equationally axiomatisable. This result sharply contrasts with the extant positive result for the fragment without zero.

Sewell not only investigates the equational theory, but also the implicational theory, showing the existence of finite computable complete sets of unifiers for finite sets of equations between processes (with zero order variables). It follows that the soundness of sequents over these is decidable. Some applications to the theories of higher order process calculi and non-well-founded sets are given.

5.2.2 Relating calculi

The π-calculus occupies a pivotal position here. A main theme in the reported work is to try to identify subsystems of the π-calculus, aiming at a particular expressive or discriminating power.

The paper by Liu and Walker [LW] is an instance of this theme: they introduce a class of processes, satisfying a confluence property, which is wide enough to include the encoding of objects (in the sense of object oriented languages), but restrictive enough that it enjoys a strong theory. More precisely, the authors start developing the theory of "partially confluent" processes, and use it to show the equivalence of two

class definitions, of which one is purely sequential while the other allows concurrent operations. The equivalence holds in arbitrary program contexts, and the main point is that when considering partially confluent processes in context, one may restrict attention to small fragments of their behaviour. The approach makes it possible to reason about the behaviour in contexts which may be very far from confluent.

It is now a well established fact that the name passing discipline of the π-calculus is expressive enough to encode agent passing. On the other hand, as the studies made in CONFER on the bisimulation notion have shown, the mathematical theory of the π-calculus is much more complex than, for instance, that of CCS. Then one may wonder whether there exist sub-calculi of the π-calculus that are "expressive enough" while having a tractable mathematical theory. This is the question addressed by Sangiorgi in [San95c, San95e, San95f].

Sangiorgi identifies a specific form of communication in the π-calculus, which he calls *internal mobility*, consisting in sending *private* names only. This is strictly less powerful than the unrestricted, "external" mobility of the π-calculus. For instance, receiving a name cannot change the communication topology inside the receiving agent. The study of the resulting πI-calculus suggests that internal mobility is responsible for much of the expressiveness of the π-calculus, whereas external mobility is responsible for much of the semantic complications. In particular, the only form of substitution involved in the semantics of πI is α-conversion of bound names: a sender and a receiver just have to agree on the name that is transmitted for a communication to occur. In fact, the bisimulation theory of πI is that of CCS, plus α-conversion.

As Sangiorgi shows, internal mobility is strongly related to agent-passing mobility. Defining a typed version of the calculus, and imposing bounds on the order of the types of πI, a hierarchy of name-passing calculi based on internal mobility is built. Symmetrically, one defines a hierarchy of agent-passing calculi within the Higher-Order π-calculus; the first level is similar to CCS, while the second is like CHOCS. Then Sangiorgi shows that there is an exact correspondence, in terms of expressiveness, between the two hierarchies. This remarkably refines the statement that "name passing is more expressive than agent passing".

The "standard" theory of translating agent passing into the π-calculus has been also further investigated. Amadio, Leth and Thomsen in [ALT] have considered the case of a calculus which may be regarded as the kernel of functional+concurrent languages such as LCS, CML or FACILE. The calculus is a simply typed, call-by-value λ-calculus, enriched with parallel composition, dynamic channel generation and input-output communication primitives. Such a calculus may be taken as a basis for, e.g. the definition of abstract machines, transformation of programs, and reasoning using modal specification languages (see the Logics area report). The authors provide a translation into the π-calculus, and show that it is adequate with respect to barbed bisimulation.

The paper [San95b] by Sangiorgi revisits results that he previously obtained regarding Milner's encoding of the lazy λ-calculus into the π-calculus. In this paper, Sangiorgi uses in fact an "asynchronous" version of the π-calculus for this encoding, and this leads him to study the theory of this calculus. He shows in particular that ground, late, early and open bisimulations all coincide and are congruences in the asynchronous polyadic

π-calculus. This simplifies the study of Milner's encoding.

Milner's encoding is also studied in the paper [BL95b], though from a different perspective. Milner showed that his translation of the lazy λ-calculus into the π-calculus is adequate with respect to the "may testing" semantics, but not fully abstract. The same holds with respect to bisimulation (of any kind, if one uses the asynchronous π-calculus). With the aim of achieving full abstraction, one may play with several parameters of the problem: enrich the source calculus, restrict the target calculus, strengthen the semantics of the source calculus, or weaken that of the target calculus – though there seems to be no room for this last possibility if we are to use may testing. For instance, Sangiorgi shows in [San95b] that full abstraction may be achieved by using a form of bisimulation on λ-terms, which he calls open applicative bisimulation. As a matter of fact, the λ-calculus semantics for which the encoding is fully abstract is the equality of Lévy-Longo trees. This was established in previous work by Sangiorgi, as well as the fact that full abstraction can be obtained by enriching the λ-calculus with non-deterministic choice, still dealing with bisimulation. In [San95f] the target calculus is restricted to be πI – though no investigation is made here on the full abstraction problem.

However, full abstraction still fails for the encoding of the lazy λ-calculus enriched with non-deterministic choice if one uses a may testing semantics for it. Therefore the question remains to understand what makes the π-calculus strictly more discriminating in this case. To tackle this question, Boudol introduced some years ago the λ-calculus with *multiplicities*. This is essentially the λ-calculus, extended with the possibility for (deterministic) evaluation to deadlock, due to a lack of resource for instantiating a variable. That is, the main difference with the usual λ-calculus is that an argument is not necessarily infinitely available. Various observation scenarios are possible in this calculus, depending on how divergence, convergence to a value, and deadlock are respectively observed.

In [BL94, BL95a] several scenarios are investigated, and characterisation results are obtained, establishing the "local structure" of the corresponding may testing semantics. In particular, the "standard" scenario, where deadlock is not observed, corresponds to a preorder on Lévy-Longo trees previously considered by Luke Ong in his PhD Thesis, called the "lazy Plotkin-Scott-Engeler preorder". It is shown also that the "flat" scenario, where both deadlock and convergence are observed (and distinguished), coincides with the inclusion of Lévy-Longo trees, and therefore also with the semantics induced by the π-calculus encoding. The conclusion is that what the π-calculus adds to the lazy λ-calculus is essentially the possibility of deadlocks (in a deterministic evaluation process).

Progress has been made also regarding the denotational semantics of the λ-calculus with resources, a non-deterministic version of the λ-calculus with multiplicities. In [Lav], Lavatelli defines a filter model for this calculus, based on a typing system previously given by Boudol. She shows that this is a fully abstract semantics for the λ-calculus with resources, provided we add a convergence testing facility, as it is standard for weak calculi. In her PhD Thesis to appear, Lavatelli shows that the filter model is the canonical solution of a domain equation, involving a new "bag-domain"

construction, for interpreting the resources.

[ALT] R. Amadio, L. Leth and B. Thomsen: *From a concurrent λ-calculus to the π-calculus*, Proceedings of the Conference on Foundations of Computation Theory, LNCS to appear, 1995. Also available as ECRC-95-18, Technical Report, ECRC München.

[BS] M. Boreale and D. Sangiorgi: *A fully abstract semantics for causality in the π-calculus*, Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science, LNCS, 1995.

[BL94] G. Boudol and C. Laneve: *The discriminating power of multiplicities in the λ-calculus*, INRIA Research Report 2441, December 1994, submitted for publication.

[BL95a] G. Boudol and C. Laneve: *Termination, deadlock and divergence in the λ-calculus with multiplicities*, Proceedings of the 11th Conference on the Mathematical Foundations of Programming Semantics, Electronic Notes in Computer Science 1, 1995.

[BL95b] G. Boudol and C. Laneve: *λ-Calculus, multiplicities and the π-calculus*, INRIA Research Report 2581, June 1995, submitted for publication.

[FMnQ] G. Ferrari, U. Montanari and P. Quaglia: *The weak late π-calculus semantics as observation equivalence*, Proceedings of the CONCUR'95 Conference, LNCS, 1995.

[FMdQ] G. Ferrari, G. Modoni and P. Quaglia: *Towards a semantic-based verification environment for the π-calculus*, to appear in the Fifth Italian Conference on Theoretical Computer Science, 1995.

[Lav] C. Lavatelli: *Full abstraction for an extended resource language*, Research Report, LIENS, 1995.

[LW] X. Liu and D. Walker: *Confluence of processes and systems of objects*, Proceedings of the CAAP'95, LNCS, 1995.

[MP] U. Montanari and M. Pistore: *Checking bisimilarity for finitary π-calculus*, Proceedings of the CONCUR'95 Conference, LNCS, 1995.

[San95a] D. Sangiorgi: *Bisimulation in higher-order calculi*, INRIA Research Report 2508, 1995. Revised version of a paper appeared in the proceedings of the IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94), North Holland, 1994.

[San95b] D. Sangiorgi: *Lazy functions and mobile processes*, INRIA Research Report 2515, 1995, submitted for publication.

[San95c] D. Sangiorgi: *πI: A symmetric calculus based on internal mobility*, TAPSOFT'95, LNCS, 1995.

[San95d] D. Sangiorgi: *Internal mobility and agent passing calculi*, ICALP'95, LNCS, 1995.

[San95e] D. Sangiorgi: *On proof method for bisimulation*, Proceedings of the Conference on the Mathematical Foundations of Computer Science, LNCS, 1995. A revised version of the technical report ECS-LFCS-94-299 has been submitted for publication.

[San95f] D. Sangiorgi: *π-Calculus, internal mobility and agent-passing calculi*, INRIA Research Report 2539, 1995, submitted for publication.

[Sew] P.M. Sewell: *The Algebra of Finite State Processes*, PhD Thesis, University of Edinburgh, CS Technical report CST-118-95, also published as LFCS-95-328, 1995.

5.3 Logics for Concurrency and λ-calculus

This is a summary of research carried out in the Logics for Concurrency and λ-calculus area. Work has been carried out at three sites: Imperial College, London, University of Pisa, and ECRC, Munich. Work done can be divided into the following categories:

- Sorts and Types for Concurrent and Distributed Computation (F. Gadducci, S. J. Gay, M. Miculan, R. Nagarajan, B. Thomsen)
- Mathematical Structures for Concurrency (S. Abramsky, S. J. Gay, G. Meredith, R. Nagarajan, D. Pavlović)
- Semantics of Functional Computation (F. Lamarche)
- Reasoning about higher-order processes (R. Amadio)

We have continued our development of specification and verification techniques for concurrency, using the syntactic and semantic typed frameworks based on Interaction Categories. Typed calculi for synchrony and asynchrony have been developed, based on the structure of Interaction Categories. Related work has resulted in the development of a typed process calculus with modal μ-types. Progress has also been made in understanding the connections between logical structures and Interaction Categories; and in the denotational modelling of mobility. There has also been continued work on Game Semantics and Linear Logic as foundations for semantics of functional computation.

Significant advances have been made in this area within the last year, and more generally, over the past three years. Our work in this area has resulted in a number of quality publications, including three papers in the highly prestigious Logic in Computer Science (LICS) conference in 1995. A summary of the work done appears in Section 5.3. A list of reports are included at the end of the description of each topic.

5.3.1 Sorts and Types for Concurrent and Distributed Computation

We present a sort and type system [ECRC/M3/R1] for a simple variant of Facile where constructs for channel creation, sending and receiving are functions of polymorphic type as opposed to syntactic constructs in the original definition of Facile. Facile is an experimental concurrent functional programming language intended to support applications that require a combination of distribution and complex computation. The language originates from an integration of the typed call-by-value λ-calculus with a model of concurrency derived from Milner's CCS. An experimental implementation currently supports distributed programming over networks of workstations.

The sort and type system is inspired by the type and effect discipline developed by Talpin and Jouvelot. The type system is polymorphic in the style of Damas and Milner, and the sort system is inspired by the type system for Nielson's TPL and it is similar to the sort system for CHOCS developed by Thomsen. The sort system is used to control type generalisation in presence of channel creation and communication. Regions (lacking a better name) are used to abstract sets of possible aliased channels.

The observable communications of an expression range over the regions that are free in its type environment and its type. Thus communications with processes which are "local" to the expression can be discarded during type reconstruction.

Apart from providing information for safe polymorphic generalisation sort and type systems may be of independent interest for concurrent functional programming since sorts and types capture essential static information about a programs dynamic behaviour, in particular its communication potential, and may thus prove useful for compile-time optimisations as well as for run-time placement of processes and channels in a distributed environment. Furthermore, sort and type information may prove valuable in module systems where the sort can be seen as enriching the signature with statements about the dynamic behaviour of encapsulated entities.

The development of typed calculi [GN95, Gay95] has progressed with the work on a calculus of asynchronous processes nearing completion [Nag95]. A calculus of asynchronous processes is defined, with a type system based on the structure of interaction categories; this corresponds to classical linear logic under the Curry-Howard isomorphism. The calculus has a denotational semantics in any category with suitable structure, which is specified abstractly, and an operational semantics which matches the categorical semantics when ASProc is taken as the semantic category. This is a natural extension of our work on the typed calculus of synchronous processes; we describe Milner's scheduler in our calculus as an example.

In related work, studies have been carried out on the development of a new paradigm for concurrency, called *behaviours-as-types* [GM95]. In this paradigm, types are used to convey information about the *behaviour* of processes: while terms corresponds to processes, types correspond to behaviours. We apply this paradigm to a simple process calculus. Its types are similar to Kozen's modal μ-calculus; hence, they are called *modal μ-types*. We prove that two terms having the same type denote two processes which behave in the same way, that is, they are *bisimilar*. We give a sound and complete compositional typing system for this language. Such a system naturally recovers the notion of bisimulation also on open terms, allowing us to deal in a compositional manner with processes with undefined parts.

[GM95] F. Gadducci and M. Miculan. Modal μ-types for processes. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995.

[Gay95] S. J. Gay. *Linear Types for Communicating Processes.* PhD thesis, University of London, 1995. Available as `theory/papers/Gay/thesis.ps.gz` via anonymous ftp to `theory.doc.ic.ac.uk`.

[GN95] S. J. Gay and R. Nagarajan. A typed calculus of synchronous processes. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995.

[Nag95] R. Nagarajan. *Typed Concurrent Programs: Specification & Verification.* PhD thesis, University of London, 1995. To appear.

[Tho94] B. Thomsen. Polymorphic Sorts and Types for Concurrent Functional Programs. In *Proceedings of the 6th International Workshop on the Implementation of Functional Languages*. J. Glauert (ed.). UEA Norwich, UK, 1994.

5.3.2 Mathematical Structures for Concurrency

While working out Abramsky's idea of processes as relations extended in time, through an abstract calculus of relations over a regular fibration [Pav95], we were led to pursue a basic representation theory of processes [Pav95a, Pav]. In spite of obvious structural kinship, the relational operations of process calculi and the operations of abstract regular logic remained distinct in an evasive way — by virtue of the former being considered *modulo bisimilarity*. The situation is comparable to the one that arises in linear algebra, where the operations of tensor calculus deviate from the universal ones by virtue of being considered *modulo bilinearity*. The next papers on *categorical logic of concurrency and interaction*, for which [Pav95] provides groundworks, should clarify the threads from logic to process calculus in a more insightful way. On the other hand, the representation theory of more realistic, more interesting models of concurrency (Petri nets, independence relations, higher dimensional automata) can be taken up only after the basic cases have been thoroughly treated [Pav95a, Pav].

Recently we have also been studying *specifications* viewed as lax functors to the bicategory of spans. A byproduct of this is a correspondence of simulations and lax natural transformations between graph morphisms into the category of relations. The bisimulation morphisms (open maps) appear in this setting as strict natural transformations. A brief report on this work has been given in [Pav95a].

Research has progressed on developing a fully abstract denotational semantics for the π-calculus. The work is highly indebted to the work of Pitts and Stark [Pit93] on the ν-calculus, and uses the same basic monadic apparatus to model the π-calculus ν. The work has been divided into two components, an axiomatic model and a more concrete model. The axiomatic model provides enough constraints on an CCC to obtain a sound model. The concrete model details a concrete category which forms the basis for a fully abstract semantics.

The work on Specification Structures and Propositions-as-types for concurrency has been further developed and applied to another standard example from concurrency—the alternating bit protocol—to illustrate verification of asynchronous deadlock-freedom in a typed framework [Abr95].

We have continued to investigate abstract axioms for interaction categories. This work was begun in [Gay95], where it led to axioms which characterise the key properties of synchronous interaction categories such as **SProc**. The novelty of the axioms consists of the notion of *guarded functor* and the use of the *unique fixed point property* to give a categorical formulation of the fact that guarded recursive definitions have unique solutions. The situation regarding asynchronous interaction categories such as **ASProc** is more subtle, as the general structure is rather weaker, and it becomes harder to express suitable axioms. Nevertheless, progress has been made and will be reported in

[Nag95].

[Abr95] S. Abramsky, S. J. Gay, and R. Nagarajan. Specification Structures and Propositions-as-Types for Concurrency. *Logics for Concurrency: Structure vs. Automata—Proceedings of the VIIIth Banff Higher Order Workshop* (G. Birtwistle and F. Moller, ed.). Springer-Verlag Lecture Notes in Computer Science. 1995.

[Gay95] S. J. Gay. *Linear Types for Communicating Processes.* PhD thesis, University of London, 1995. Available as `theory/papers/Gay/thesis.ps.gz` via anonymous ftp to `theory.doc.ic.ac.uk`.

[Nag95] R. Nagarajan. *Typed Concurrent Programs: Specification & Verification.* PhD thesis, University of London, 1995. To appear.

[Pav95] 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 Theory and Formal Methods Section Workshop*, IC Press, 1995.

[Pav95a] D. Pavlović. Convenient Categories of Processes and Simulations I: Modulo Strong Bisimilarity. In *Proceedings, Sixth Conference on Category Theory and Computer Science.* 1995.

[Pav] D. Pavlović. Convenient Categories of Processes and Simulations II: Asynchronous Cases, submitted for publication.

5.3.3 Semantics of Functional Computation

In the area of denotational semantics, the connection between Chu spaces and ordinary Scott-continuous semantics was first observed in [Lam], in which we gave a model of Classical Linear Logic whose intuitionistic category (the Kleisli category of the "of course" comonad) was a well-known model of computation, namely the category of bifinite domains and Scott-continuous functions [Abr]. This work was then simplified in [Lam95a] in which the intuitionistic universe can now be the much larger category of all cpos and Scott-continuous functions. In these two papers the objects of the linear category are Chu spaces [Bar79, Pra94] that are required to satisfy additional conditions which allow recursion (fixpoints) to happen at the level of terms (and types as well, it seems). The ordinary topological axioms of Scott continuity are then *derived* from the requirements needed to have cocommutative comonoids, they being the result of applying the "of course" operator. One outcome of this is a cartesian closed category of topological spaces suitable for modelling computation which is even larger than that of cpos and Scott-continuous functions.

Another progress in denotational semantics was the realisation that two standard models of Linear Logic, coherence spaces [Gir87] and hypercoherences [Ehr93], were

actually particular cases of a very general construction, quite similar to the well-known process in set and topos theory of constructing Q-valued sets, where Q is a complete Boolean algebra, or more generally a locale (complete Heyting algebra). In our case Q becomes a Girard quantale (the equivalent of a locale in Linear Logic) but the construction does not try to emulate the category of sets and functions, but instead that of sets and *relations*. The work is described in [Lam95] and has already been further generalised by V. de Paiva and A. Schalk.

For some time this author has been advancing the thesis that the natural logic of games semantics is the fragment of Intuitionistic Linear Logic that has only the connectives $\otimes, \multimap, \&$ and ! [Lam94]. Some people have suggested calling this fragment Minimal Linear Logic, or MiLL. This seems to be a fairly weak fragment, so how can more general connectives be captured? In general this probably should be done by translations into MiLL. One first fully documented instance of this is [Lam95b], in which a "double negation" translation, much reminiscent of (and indeed inspired by) the one given in [Gir91] is used to give a games semantics for full propositional classical linear logic. One feature of that semantics is an illustration of the slogan "an opponent's move–player's response pair in a strategy corresponds to an axiom link in MiLL".

[Lam95] F. Lamarche Generalizing coherence spaces and hypercoherences. In *Proceedings, Eleventh Conference on the Mathematical Foundations of Programming Semantics*. 1995.

[Lam95a] F. Lamarche From Chu spaces to cpos. 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 Theory and Formal Methods Section Workshop*, IC Press, 1995.

[Lam95b] F. Lamarche Games Semantics for Full Classical Propositional Linear Logic In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995.

[Lam94] F. Lamarche Proof nets for Intuitionistic Linear Logic I: Essential nets, Preprint, April 1994.

[Lam] F. Lamarche Dialectics: a model of linear logic and PCF, submitted to *Mathematical Structures in Computer Science*.

5.3.4 Reasoning about higher-order processes

Amadio and Dam address the specification and verification problem for process calculi such as Chocs, CML and Facile where processesor functions are transmissible values. This work takes place in the context of a static treatment of restriction and of a bisimulation-based semantics. As a paradigmatic and simple case, they concentrate on (Plain) Chocs. They show that Chocs bisimulation can be characterized by an extension of Hennessy-Milner logic including a constructive implication, or function

space constructor. This result is a non-trivial extension of the classical characterization result for labelled transition systems. A proof system for the verification of process specifications have been developed with a sound proof system for a Chocs sub-calculus not including restriction. There are two completeness results: one for the full specification language using an infinitary system, and one for a special class of so-called well-described specifications using a finitary system.

[AD:caaps95] R. Amadio, M. Dam, *Reasoning about higher-order processes* , In CAAP 95, Aarhus, SLNCS 915. Also appeared as SICS Research Report 94-18, 1994.

5.3.5 Interrelations between sites and to other areas

The work on typed process calculi is directly applicable to the calculi area and draws upon results in that area. One of the aims of our work in the Logics for Concurrency and λ-calculus area is to develop techniques that forms the basis for implementation of programming languages and hence has an impact in that area as well.

There is a close connection between the work done at the University of Pisa and Imperial College, London in the area of typed process calculi. Research on linear types at Imperial could have an impact in the future on the type system of FACILE under development at ECRC. Of course, we are still seeking to establish the common thread that should connect the work on Interaction Categories at Imperial and Action Structures at the University of Cambridge and the University of Edinburgh.

5.3.6 Future Work

We envisage that our work will continue along the following lines:

- A full implementation of a polymorphic type system for Facile based on effect analysis. This is already being developed by Tsung-Min Kuo in the context of LOMAPS BRA.

- Investigation of techniques from linear types and Interaction Categories in order to incorporate them into the FACILE type system.

- Extension of the typed process calculi based on interaction categories to address issues such as deadlock-freedom and object-oriented programming.

- We also plan to investigate how modal μ-types could be extended to convey information about different kind of (truly) concurrent semantics. Moreover, we aim to recover minimal realisations of transition systems for typed process terms.

- Another immediate goal is to describe how the idea of *mobility* can be captured within the framework of *processes-as-relations* (Interaction Categories), while also accommodating the *duality of states and events* (Chu spaces).

- Extension of the work on Chu spaces in the category of sets to the category of posets, which gives a foundation to the theory of bidomains.
- Working out in full the relationship between games semantics and proof nets.

Other References

[Abr] S. Abramsky and A. Jung. Domain theory. In *Handbook of Computer Science*, Oxford University press.

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

[Gir87] J-Y. Girard. Linear Logic. *Theoret. Comp. Sci* 50(1987)1–20.

[Gir91] J-Y. Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science* 1, 3(1991)255–296.

[Ehr93] T. Ehrhard. Hypercoherences, a strongly stable model of linear logic. *Mathematical Structures in Computer Science* 3(1993)365-385.

[Pit93] A. M. Pitts and I. D. B. Stark. In *Proceedings, Eighteenth International Symposium on Mathematical Foundations of Computer Science*. Springer-Verlag Lecture Notes in Computer Science. Vol 711. 1993.

[Pra94] V. Pratt. Chu spaces: complementarity and uncertainty in rational mechanics. Course notes for TEMPUS summer school, Budapest, 1994.

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.

Year 3 of the action has been very successful and results, systems and application experiments beyond the expectations expressed in the Second Year report have been achieved. In Year 3 of CONFER results that looked very theoretical three years ago, such as Lamping's optimal graph reduction technique, the λ-calculus with explicit substitutions and the π-calculus, have been used, further developed and shown to be of practical relevance in the programming language area of CONFER. Furthermore, the principle of functional and concurrent programming is now proving to be industrially viable in the construction of distributed systems. It should be said that many of the results have been achieved in collaboration with researchers outside the project. Thus the project strategy of being organised with participation of people not directly funded by CONFER is paying off.

This year a major effort has been put into a release of PICT, including a PICT-to-C compiler, a reference manual, a language tutorial and libraries. Furthermore, new developments for the type system of PICT to include linear types, capable of ensuring that certain channels are only used once, have been completed. PICT is now used in undergraduate courses in Warwick and Cambridge.

At INRIA-Rocquencourt work on a distributed version of PICT suitable for use on the WWW has begun and is becoming a major research direction for the group.

To demonstrate the industrial viability of programming distributed systems based on the integration of functional and concurrent programming several larger applications have been developed using the Facile language. These applications have been developed at ECRC. All applications have been shown to an audience of industrialist, EU officials and members of the press at an event in Brussels. A major effort is currently being devoted to preparing a second release of Facile. Facile is now licenced by more than 25 sites in Europe, the US and Asia.

A prototype implementation of (a variant of) Lamping's optimal graph reduction technique: the Bologna Optimal Higher-order Machine [BOHM], has been developed. BOHM is a simple interpreter written in C, and it runs on most systems having a C compiler, including Personal Computers. The extension of Lamping's graph reduction techniques to this language is essentially based on Asperti and Laneve's work on Interaction Systems done in Year 1 and 2 of CONFER.

Inspired by previous work by Boudol new techniques on compile time optimisations via unfolding of programs useful for languages combining functional and concurrent computations have been developed. The techniques are based on an abstract setting where the computational part of a systems is given by a confluent abstract reduction system and the descriptive part is a possibly non-deterministic system.

In Year 3 of CONFER Doligez and Gonthier completed the mechanical verification of their concurrent garbage collector. They checked a 22000 line proof script that demonstrated the partial correctness of their algorithm, making it the first realistic,

fully checked garbage collection algorithm.

Based on previous work on the λ-calculus with explicit substitutions a weak λ-calculus, $\lambda\sigma_w$, has been defined by Hardin, Maranget and Bruno Pagano. This calculus has been used to prove the correctness of many skeleton compilers and runtime systems modeled as abstract machines, including the Krivine machine, the CAM, the SECD, the FAM, the TIM, the G-machine.

The report includes short summaries of the following efforts:

- Higher-order languages for distributed applications
  INRIA-Rocquencourt — D. Doligez, C. Fournet, G. Gonthier, F. Guillaume, J.-J. Lévy,
  L. Maranget, D. Rémy

- Pict programming Language
  University of Cambridge — B. Pierce, D. Turner.

- Observational equivalence of Pict programs
  University of Cambridge — P. Sewell.

- The Facile programming language
  ECRC — P.-Y. Chevalier, F. Knabe, A. Kramer, T.M. Kuo, L. Leth, I. Nikolaidis, J.P. Talpin, B. Thomsen.

- Mobile Service Agents
  ECRC — F. Knabe, P.-Y. Chevalier, L. Leth, B. Thomsen.

- Einrichten
  ECRC — K. Ahlers, A. Kramer.

- Cooperative Mechanical Repair
  ECRC — J.-Y. Litzler, A. Kramer, P.-Y. Chevalier.

- Correct Transformations of Nondeterministic Programs
  SICS — B. Lisper

- Implementation of Optimal Reduction
  Bologna — A. Asperti, C. Laneve

- Concurrent program verification
  Inria-Rocquencourt — D. Doligez, G. Gonthier

- Explicit substitutions
  Inria-Rocquencourt — T. Hardin, P.-A. Melliès

5.4.1 Higher-order languages for distributed applications

INRIA Rocquencourt oriented its efforts this year on the development of a higher-order, distributed language supporting mobile execution, suitable for use on the World Wide Web. Although the effort is still in the early design stages, significant progress has been made and a prototype implementation by Guillaume is now under way.

The initial design was based on the π-calculus, and consisted roughly in adding explicit control over distribution to the PICT design. It was found that adequate expressiveness required a combination of lexical scope for defining process groups, and dynamic site assignment to allow for run-time migration.

However, it was also found that the precise distributed behaviour of this language was inordinately complex, especially if one accounts for faults. The problem was traced to the choice of primitives in the π-calculus, which turn out to be almost impossible to implement efficiently in a distributed setting.

This has led Fournet and Gonthier to propose a new process calculus, the *join-calculus*, as the formal basis for our language. The join-calculus eliminates the "ethereal communication" that the π-calculus inherited from CCS, by merging in a single construct the restriction, input guard and iteration constructs of the π-calculus. The resulting construct can essentially be construed as "`let` continuation `in` body", and dually output terms as continuation calls. To compensate for what would otherwise be a loss of expressive power, the join-calculus also allows the definition of *join continuations*, i.e., continuations that need to be called from several entries to proceed. It has been shown that the join-calculus is exactly equivalent to the π-calculus.

Because all communications are located at a continuation, the join-calculus has a clear, efficiently implementable distributed semantics. Furthermore pattern-matching can easily be added to the join-calculus, and this yields a rather expressive language, which smoothly combines ML-style functional programming with Actor-style concurrent object-oriented programming.

[1] Fournet, C., Gonthier, G.: "The reflexive CHAM and the join-calculus". Submitted to POPL '96.

5.4.2 The Pict programming language

Pict is a language in the ML tradition, formed by adding a layer of convenient syntactic sugar and a static type system to a tiny core.

The core language — an asynchronous variant of Milner, Parrow, and Walker's pi-calculus — has been used as a theoretical foundation for a broad class of concurrent computations. The goal in Pict is to identify high-level idioms that arise naturally when these primitives are used to build working programs – idioms such as basic data structures, protocols for returning results, higher-order programming, selective communication, and concurrent objects. The type system integrates a number of features found in recent work on theoretical foundations for typed object-oriented languages: higher-order polymorphism, simple recursive types, subtyping, and a powerful partial type inference algorithm.

The Second Year progress report concentrated on implementation efforts. These are now largely complete, and we are now in the process of releasing the compiler to an increasingly large circle of users. The current release includes a Pict-to-C compiler, reference manual, language tutorial, libraries for common data structures, example programs, and a rudimentary X-based widget toolkit. It is in active use at several sites, and is being used for undergraduate teaching in Warwick and Cambridge.

Besides polishing the software for the release, a good deal of our energy this year has gone into documentation. The tutorial notes on the language have been greatly expanded (partly in use for teaching), we have published one conference paper on our designs for concurrent objects, and we are working on a "language retrospective" paper for journal publication.

Our work during the past two years on type systems for Pict — as well as related work on type systems for sequential object-oriented languages — also bore fruit this year in several new papers. The most exciting of these is also the newest (by Kobayashi, Pierce, and Turner). It describes a type system with *linear* types, capable of ensuring that certain channels are only used once during the execution of a program. This refinement justifies some useful program transformations, such as a "tail call optimisation," which do not preserve program equivalence under existing notions of equivalence.

[1] Ghelli, G., Pierce, B.: "Bounded Existentials and Minimal Typing". Submitted to Information and Computation, 1995.

[2] Hofmann, M., Pierce, B.: "Positive Subtyping". Proceedings of Twenty-Second Annual ACM Symposium on Principles of Programming Languages, January 1995. Full version submitted for publication.

[3] Kobayashi, N., Pierce, B.C., Turner, D.N.: "Linearity and the Pi-Calculus". Submitted to POPL '96.

[4] Pierce, B.C.: "Programming in the Pi-Calculus: An Experiment in Programming Language Design". Tutorial notes on the Pict language. Available with the Pict distribution, 1995.

[5] Pierce, B.C., Turner, D.N.: "Concurrent Objects in a Process Calculus". Theory and Practice of Parallel Programming (TPPP), Sendai, Japan, Nov. 1994. (Invited talk.) Springer-Verlag LNCS 907.

[6] Pierce, B.C., Turner, D.N.: "Pict: A Programming Language Based on the Pi-Calculus". Invited journal article, in preparation, 1995.

[7] Pierce, B.C.: "Foundational Calculi for Programming Languages". CRC Press Handbook of Computer Science and Engineering. To appear, 1996.

5.4.3 Observational equivalence of Pict programs

The Pict programming language is sufficiently concrete to admit detailed consideration of the interactions between a Pict program and its environment. A new project by

Sewell has described these informally and given a formal model of the composition of a program and an environment, differing in several ways from the standard π-calculus parallel composition. Using this model definitions can be given. of abstract-machine-correctness and observational equivalence that are closely related to the observations a Pict user might make.

5.4.4 The Facile programming language

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 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).

As mentioned in last years report, the availability of the first release of Facile – the Facile Antigua Release – was publicly announced in July 1994. At the same time technology transfer to the ECRC shareholder companies took place. Currently, there are more than 25 licenced sites, including sites in Europe, Asia and the US. In the Autumn of 1995 we expect to make a second release of Facile – the Facile Barbados Release. Apart from tuning and bug fixes, the major novel features of the Facile Barbados Release will be support for multi-platform environments (SPARC and Intel 486 processors), reliable group communication and support for interoperability with systems implemented in different programming languages (such as C, C++, prolog) via the ONC-PRC/XDR industrial standard.

Beyond the Facile Barbados Release, we expect future releases of Facile to include a much lighter weight implementation (the current implementation requires 10MB per Facile node) and integration of a set of sophisticated "middleware" services, such as security, authentication and persistence. As already mentioned, a new type system for Facile based on the principle of effects analysis is being developed in the context of the LOMAPS BRA 8130 action.

[ECRC/M3/R1] Thomsen, B.: "Polymorphic Sorts and Types for Concurrent Functional Programs". Proceedings of the 6th International Workshop on the Implementation of Functional Languages (Ed. J. Glauert), UEA Norwich, UK, 1994.

[ECRC/M3/R6] Amadio, R., Leth, L. and Thomsen, B.: "From a Concurrent λ-calculus to the π-calculus". Technical report, ECRC-95-18, 1995.

[ECRC/M3/R14] Thomsen, B., Leth, L. and Kuo, T.-M.: "FACILE – from Toy to Tool". To appear in: *ML with Concurrency: Design, Analysis, Implementation, and Application*, Flemming Nielson (Editor), Springer-Verlag, 1996.

5.4.5 Mobile Service Agents

The last few years have seen a growing interest in mobile computing. The use of portable computers in conjunction with wireless communication networks provides new means of interaction between users and computers. As the hardware infrastructure begins to materialise, there are more possibilities (and greater need!) for software applications that are adapted to this highly dynamic setting. Facile's basis on formal process models for mobility (e.g., the π-calculus and CHOCS) and on functional programming languages has allowed it to inherit features and a basic approach that provide an edge in developing a significant class of mobile applications. We call this class Mobile Service Agent (MSA). MSAs are self-contained pieces of software that can move between computers on a network. Agents can serve as local representatives for remote services, provide interactive access to data they accompany, and carry out tasks for a mobile user temporarily disconnected from the network. Agents also provide a means for the set of software available to a user to change dynamically according to the user's needs and interests.

In the MSA architecture, the Global Information Infrastructure (GII) is seen as consisting of sites each supporting various services based on agents. Upon connecting to a site, a "directory agent" is first dispatched to the user's machine. This directory agent serves simultaneously as an informational guide to the site and as a conduit through which other agents from the site can be introduced to the user's computer.

As proof of concept we have constructed a demonstration of the MSA architecture that is centered on the scenario of a person from ECRC in Munich going to a meeting in Brussels. The person is carrying a portable computer and is temporarily disconnected from the GII while in transit.

The two sites, ECRC and Brussels, both provide various services that the user may access at various points in the trip. For example, prior to departure from ECRC, the user can connect to Brussels via the GII and obtain such services (implemented as agents) as an interactive map of the conference site and an agenda for the conference. While in transit the user can continue to use these services even though the connection to Brussels is unavailable. Upon reconnection to the GII in Brussels or elsewhere, the agenda agent automatically takes advantage of the opportunity to communicate with its home location and updates itself with the latest information and bulletins for the user.

References

[ECRC/M3/R8] Thomsen, B., Leth, L., Knabe, F. and Chevalier, P.-Y.: "Mobile Agents". Technical report, ECRC-95-21, 1995.

[ECRC/M3/R9] Thomsen, B., Knabe, F., Leth, L. and Chevalier, P.-Y.: "Mobile Agents Set to Work". Communications International, July, 1995.

5.4.6 Einrichten

Einrichten is a collaborative design system developed at ECRC. It is a demonstration of interactive graphics and real-time video for the purpose of interior design. The system combines the use of a heterogeneous database system of graphical models, an augmented reality system, and the distribution of 3D graphics events over a computer network. This application shows how improvements in computing and communication hardware can be combined with sophisticated software platforms to produce powerful results for end users.

The scenario for this application consists of an office manager who is working with an interior designer on the layout of a room. The office manager intends to order furniture for the room. On a computer monitor the pair see a picture of the room from the viewpoint of the camera. By interacting with various manufacturers over a network, they select furniture by querying databases using a graphical paradigm. The system provides descriptions and pictures of furniture that is available from the various manufacturers who have made models available in their databases. Pieces or groups of furniture that meet certain requirements such as colour, manufacturer, or price may be requested. The users choose pieces from this "electronic catalogue", and 3D renderings of this furniture appear on the monitor along with the view of the room. The furniture is positioned using a 3D mouse. Furniture can be deleted, added and rearranged until the users are satisfied with the result; they view these pieces on the monitor as they would appear in the actual room. As they move the camera they can see the furnished room from different points of view.

The users can consult with colleagues at remote sites who are running the same system. Users at remote sites manipulate the same set of furniture using a static picture of the room that is being designed. Changes by one user are seen instantaneously by all of the others, and a distributed locking mechanism ensures that a piece of furniture is moved only by one user at a time. In this way groups of users at different sites can work together on the layout of the room. The group can record a list of furniture and the layout of that furniture in the room for future reference.

The 3D graphics and augmented vision components of the system are built with GSP, a software platform which combines interactive 3D graphics and computer vision technology to calibrate, align, and display 3D models and real-time video.

The system also provides a means of distributing graphics events in a transparent manner. The communication of these events to multiple GSP applications, the locking, and the management of users joining and leaving the group is achieved through Facile, which provides the tools for reliable connections, disconnections and multicasts.

[ECRC/M3/R4] Ahlers, K. et al.: "Distributed Augmented Reality for Collaborative Design Applications". Technical report, ECRC-95-03, 1995.

[ECRC/M3/R5] Ahlers, K. et al.: "Distributed Augmented Reality for Collaborative Design Applications". To appear in Proceedings of Eurographics '95, Maastricht, the Netherlands (August 28 to 1 September 1995).

5.4.7 Cooperative Mechanical Repair

The main goal of this application is to assist a field engineer in maintaining and repairing a complex artifact (e.g., an aircraft engine) using augmented reality (AR) technology.

Using a head-mounted or portable display with a see-through visor, the engineer is able to work on the engine while "seeing" additional computer generated information (e.g., identification of a component). Annotations may identify the names of parts, describe their function, or present other important information like maintenance or manufacturing records. AR may lead the mechanic through a specific task by highlighting parts that must be sequentially removed and showing the path of extraction. The system may also provide safety information. Parts that are hot or electrified can be flagged as a constant reminder. The mechanic may also be assisted by a remote expert who can control what information is displayed on the mechanic's AV system. The application supports remote mechanical inspection and allow the field engineer and a remote expert to interact and complete the repair task together. Access to maintenance and supplier databases (e.g., history, prices, physical characteristics) will be available. Diagnostic information (generated by a remote expert system) may be presented.

On the technical side and apart from the AR, the main features of this application are:

- Video transmission
- Event broadcast (remote interaction and tracking)
- Fault-tolerance (replicated data and dynamic joining)
- Interoperability C++/Facile (contract)

5.4.8 Correct Transformations of Nondeterministic Programs

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 extends previous activities regarding correctness of unfolding of nondeterministic expressions.

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

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

The following abstract setting has been used: the computational part of a system is given by a confluent abstract reduction system $\rightarrow_D$, and the descriptive part by a possibly non-confluent (i.e. non-deterministic) system $\rightarrow_N$. The system itself is described by the union of these reduction systems. A semantics is defined as the set of limits, under some monotone interpretation to a c.p.o., for the various (possibly infinite) reduction paths, in the vein of Boudol [**?**]. This can be thought of as a "normal form semantics", but extended to deal with infinite and divergent (but "fair") computations. In particular, we consider reduction systems over terms, with interpretation to c.p.o.'s of Böhm trees over the same signature. Within this framework, we prove the following:

1. If $\rightarrow_D$ and $\rightarrow_N$ *commute*, then for any *cofinal* reduction strategy F for $\rightarrow_D$ there exists a semantically correct, nondeterministic (w.r.t. $\rightarrow_N$) reduction strategy, where F implements the computational part, for any monotone interpretation.

2. If $\rightarrow_D$ and $\rightarrow_N$ commute, then certain transformations preserve the semantics, for any monotone interpretation.

"Semantically correct" here means to have the same set of "normal forms" as $\rightarrow_D \cup \rightarrow_N$. Thus, there is always a deterministic "implementation" of the computational part that "preserves all possibilities" prescribed by the semantics, i.e., does not destroy the description of the environment.

The semantics-preserving transformations include symbolic folding and unfolding of recursive programs defined by term rewriting rules.

The results can be understood in the following way: the fundamental condition that $\rightarrow_D$ and $\rightarrow_N$ commute makes a language with the considered semantics *referentially transparent* w.r.t. the convertibility relation $\leftrightarrow_D$, i.e. if $a \leftrightarrow_D a'$ then they are interchangeable in any context, even nondeterministic ones. It follows that program transformations which are correct w.r.t. $\leftrightarrow_D$ can be used freely even when computing in nondeterministic environments.

A possible application is the design of recursive languages with process primitives where the "serial part" is referentially transparent. Another is the design of semantically correct "macro languages", with facilities found in ordinary programming languages, to put on top of pure process algebraic languages being used in e.g. verification tools like the Mobility Workbench. This eliminates the need to encode typical "serial" operations and data in the process algebra. Our approach is complementary to PICT where everything is encoded in the π-calculus.

[1] Lisper, B.: "Computing in Unpredictable Environments: Semantics, Reduction Strategies, and Program Transformations", 1995. Submitted to POPL'96.

5.4.9 Implementation of Optimal Reduction

Andrea Asperti has developed a prototype implementation of (a variant of) Lamping's optimal graph reduction technique: the Bologna Optimal Higher-order Machine [BOHM]. BOHM is a simple interpreter written in C, and it runs on most systems having a C compiler, including Personal Computers.
BOHM's source language constitutes the full core of a typical (dynamically-typed) lazy functional language, including a few data-types, conditional expressions, recursive values and multiple recursion and lazy pattern matching.
The extension of Lamping's graph reduction techniques to this language is essentially based on Asperti and Laneve's work on Interaction Systems [AL93d]. These Rewriting Systems provide a nice integration of the functional paradigm with a rich class of data structures (all inductive types), and basic control flow constructs such as conditionals and (primitive or general) recursion. More specifically, Interaction Systems are the *intuitionistic* generalisation of Lafont's Interaction Nets, from which they borrow the logical setting, the bipartition of operator into constructors and destructors, and the principle of binary interaction. ¿From a different point of view, Interaction Systems (IS's) can be also regarded as the subclass of Klop's Combinatory Reduction Systems where the Curry-Howard (Proofs as Proposition) analogy still "makes sense". This means that we may associate every IS with a suitable "intuitionistic" system: constructors and destructors respectively correspond to right and left introduction rules, interaction is cut and computation is cut-elimination.
The generalisation of Lamping's optimal graph reduction technique to the case of Interaction System is described in [AL93a, AL93c]. This implementation deeply exploits the "logical nature" of Interaction Systems, and in particular their relations with Interaction Nets and Linear Logic.
The implementation of BOHM also put in evidence the dramatical relevance of the problem of *accumulation of control operators*. A (partial) solution to this problem has been recently proposed by Asperti in [RTA95]. This solution has been implemented in BOHM (in a more sophisticated and general way), and it looks quite satisfactory in practice.
We have compared BOHM's performance with two standard and fully developed implementations of functional languages: Caml-light (Rocquencourt, France) and Yale Haskell (UK). The choice of these two languages was motivated by the fact that they are respectively eager and lazy implementations, and it may come as a surprise to many people that BOHM can be better than both these paradigms at the same time. In general, BOHM's performance is always comparable with that of Yale Haskell (i.e. approximately ten time slower than Caml-light), while it can run surprisingly faster on many interesting examples. Two typical cases are the incremental modification of arrays represented as functions from indices to values, and especially the direct evaluation of Denotational Semantics for imperative languages. These two problems are absolutely

unrealistic with traditional implementation techniques, while having a reasonable time behaviour in BOHM.

[AL93a] Asperti, A., Laneve, C.: "Optimal Reductions in Interaction Systems". Proc. of the 4th Joint Conference on the Theory and Practice of Software Development, TAPSOFT'93, Orsay (France). April 1993.

[AL93c] Asperti, A., Laneve, C.: "Interaction Systems II: the practice of optimal reductions". Theoretical Computer Science, to appear (also available as Technical Report UBLCS-93-12, Laboratory for Computer Science, university of Bologna, Italy).

[AL93d] Asperti, A., Laneve, C.: "Interaction Systems". International Workshop on Higher-Order Algebra, Logic and Term Rewriting. Amsterdam, September 1993.

[RTA95] Asperti, A.: "$\delta \circ !\epsilon = 1$. Optimising optimal λ-calculus implementations". Proc. of the sixth International Conference on Rewriting Techniques and Applications, RTA'95, Kaiserslautern, Germany. 1995.

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

5.4.10 Concurrent program verification

Doligez and Gonthier completed the mechanical verification of their concurrent garbage collector. Using the TLP system of Guttag, Horning and Engberg (which is based on Lamport's TLA), they checked a 22000 line proof script that demonstrated the partial correctness of their algorithm, making it the first realistic, fully checked garbage collection algorithm. The proof, which stressed the theorem prover to its limits, should provide a good benchmark for future concurrent program verification systems.

[1] Doligez, D.: "Conception, réalisation et certification d'un glaneur de cellules concurrent". Doctoral dissertation, Université Paris VII, May 1995

5.4.11 Explicit substitutions

Hardin and Maranget together with Bruno Pagano, defined a weak λ-calculus, $\lambda\sigma_w$, as a subsystem of the full λ-calculus with explicit substitutions. They used this calculus to prove the correctness of many skeleton compilers and runtime systems modeled as abstract machines, including the Krivine machine, the CAM, the SECD, the FAM, the TIM, the G-machine. This provides evidence that $\lambda\sigma_w$ could be the archetypal output language of functional compilers, just as the λ-calculus is their universal input language, and that its theory could be adequate to establish the correctness of functional compilers.

[1] Dowek, G., Hardin, T., Kirchner, C.: "Higher-Order Unification via Explicit Substitutions". Proceedings of the Tenth Annual IEEE Symposium on Logic in Computer Science, San Diego, California, June 1995

[2] Hardin, T., Maranget, L., Pagano, B.: "Abstract Machines are Strategies in the Weak Lambda-Calculus with Substitutions". Submitted for publication

[3] Melliès, P.-A.: "Typed λ-calculi with explicit substitutions may not terminate". Proceedings of the Second International Conference on Typed Lambda Calculi and Applications, Edinburgh, United Kingdom, April 1995

Chapter 6

Appendices

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

CONFER-130	Z.M. Ariola J.W. Klop	*Equational term graph rewriting* CWI Report CS-R9552, July 1995 (short version also to appear in Fundamenta Informaticae 1995)
CONFER-131	J.A. Bergstra P. Klint	*The discrete Time ToolBus* Technical Report P9502, March 1995, Programming Research Group, University of Amsterdam
CONFER-132	Jan A. Bergstra Inge Bethke Alban Ponse	*Process Algebra with Iteration and Nesting* The Computer Journal, Vol. 37, No. 4, 1994
CONFER-133	J.R. Kennaway J.W. Klop M.R. Sleep F.-J. de Vries	*Infinitary lambda calculus* (published in RTA-95 proceedings as 'Infinitary lambda calculi and Böhm models', Kaiserslautern, April 1995, Springer LNCS 914, p.257-270, ed. Jieh Hsiang), CWI Report CS-R9535, May 1995.
CONFER-134	V. van Oostrom F.van Raamsdonk	*Weak orthogonality implies confluence: the higher-order case* (short version already appeared in proceedings of LFCS '94, Springer LNCS 813,p.379-392) CWI Report CS-R9501, January 1995.
CONFER-135	F.van Raamsdonk P. Severi	*On normalisation* CWI Report CS-R9545, June 1995

CONFER-136	Robin Milner	*Calculi for Interaction*, To appear in Acta Informatica, 1996.
CONFER-137	Philippa Gardner	*A Name-free Account of Action Calculi*, Proceedings of Mathematical Foundations of Programming Semantics, New Orleans, 1995.
CONFER-138	Alex Mifsud Robin Milner John Power	*Control Structures*, Proceedings of IEEE Conference on Logics in Computer Science, San Diego, 1995.
CONFER-139	Claudio Hermida John Power	*Fibrational control structures*, Proceedings of CONFER Conference, 1995.
CONFER-140	Ghelli, G Pierce, B.	*Bounded Existentials and Minimal Typing*, Submitted to Information and Computation, 1995.
CONFER-141	Hofmann, M. Pierce, B.	*Positive Subtyping* Proceedings of Twenty-Second Annual ACM Symposium on Principles of Programming Languages, January 1995. Full version submitted for publication.
CONFER-142	Kobayashi, N. Pierce, B.C. Turner, D.N.	*Linearity and the Pi-Calculus* Submitted to POPL '96.
CONFER-143	Pierce, B.C. Turner, D.N.:	*Concurrent Objects in a Process Calculus*, Theory and Practice of Parallel Programming (TPPP), Sendai, Japan, Nov. 1994. (Invited talk.) Springer-Verlag LNCS 907.
CONFER-144	Pierce, B.C.	*Foundational Calculi for Programming Languages*, CRC Press Handbook of Computer Science and Engineering. To appear, 1996.
CONFER-145	Philippa Gardner Robin Milner	*A short note on Closed Action Calculi,* DRAFT, 1995.
CONFER-146	John Power	*Elementary control structures* January 1995
CONFER-147	John Power	*Control structures III: arity monoids and their associated naming monoids* August 1994
CONFER-148	Pierce, B.C.	*Programming in the Pi-Calculus: An Experiment in Programming Language Design.* Tutorial notes on the Pict language. Available with the Pict distribution, 1995.
CONFER-149	Pierce, B.C. Turner, D.N.	*Pict: A Programming Language Based on the Pi-Calculus* Invited journal article, in preparation, 1995.

CONFER-150	Thomsen, B.	*Polymorphic Sorts and Types for Concurrent Functional Programs*, Proceedings of the 6th International Workshop on the Implementation of Functional Languages (Ed. J. Glauert), UEA Norwich, UK, 1994.
CONFER-151	Lévy, J.-J. Thomsen, B. Leth, L. Giacalone, A.	*Second Year Report for Esprit Basic Research Action 6454-CONFER CONcurrency and Functions: Evaluation and Reduction*, Bulletin of EATCS, Number 55, 1995, pp. 68-87.
CONFER-152	Thomsen, B.	*A Theory of Higher Order Communicating Systems*, *Information and Computation* Vol. 116, No 1, pp. 38-57, January 1995.
CONFER-153	Ahlers, K. et al.	*Distributed Augmented Reality for Collaborative Design Applications*, Technical report, ECRC-95-03, 1995.
CONFER-154	Ahlers, K. et al.	*Distributed Augmented Reality for Collaborative Design Applications*, To appear in Proceedings of Eurographics '95, Maastricht, the Netherlands (August 28 to 1 September 1995).
CONFER-155	Amadio, R. Leth, L. Thomsen, B.	*From a Concurrent λ-calculus to the π-calculus* Technical report, ECRC-95-18, 1995.
CONFER-156	Amadio, R. Leth, L. Thomsen, B.	*From a Concurrent λ-calculus to the π-calculus* To appear in Proceedings of Fundamentals of Computation Theory, FCT'95.
CONFER-157	Thomsen, B. Leth, L. Knabe, F. Chevalier, P.-Y.	*Mobile Agents*, Technical report, ECRC-95-21, 1995.
CONFER-158	Thomsen, B. Knabe, F. Leth, L. Chevalier, P.-Y.	*Mobile Agents Set to Work*, Communications International, July, 1995.
CONFER-159	Leth, L. Thomsen, B.	*Some Facile Chemistry (Summary)*, Accepted for publication in Formal Aspects of Computing, 1995.
CONFER-160	Leth, L. Thomsen, B.	*Some Facile Chemistry*, Accepted for electronic publication in Formal Aspects of Computing, 1995.
CONFER-161	Thomsen, B. Abramsky, S.	*A Fully Abstract Denotational Semantics for the Calculus of Higher Order Communicating Systems*, Accepted for publication in TCS, 1994.

CONFER-162	Thomsen, B.	*Logic, Domains and Higher-Order Processes*, To appear in the Proceedings of Workshop on Logic, Domains, and Programming Languages (LDPL'95), Technische Hochschule Darmstadt, Germany, May 24–27, 1995.
CONFER-163	Thomsen, B. Leth, L. Kuo, T.-M.	*FACILE – from Toy to Tool*, To appear in: *ML with Concurrency: Design, Analysis, Implementation, and Application*, Flemming Nielson (Editor), Springer-Verlag, 1996.
CONFER-164	R. Amadio M. Dam	*Reasoning about higher-order processes.* In CAAP 95, Aarhus, SLNCS 915. Also appeared as SICS Research Report 94-18, 1994.

CONFER-165 R. Amadio, L. Leth, B. Thomsen. *From a concurrent λ-calculus to the π-calculus.* In *Foundations of Computation Theory 95, Dresden*, SLNCS to appear. Expanded version appeared as ECRC-95-18. Available at ftp.ecrc.de.

CONFER-166 V. Danos, L. Regnier. *Reversible and Irreversible Computations (GOI and Environment* Machines), `http://boole.logique.jussieu.fr/www.danos/experience.html`

CONFER-167 C. Lavatelli. *Full abstraction for an extended resource language*, Technical Report, LIENS 95-18.

CONFER-168 S. Abramsky, S. J. Gay, R. Nagarajan. *Specification Structures and Propositions-as-Types for Concurrency.* Logics for Concurrency: Structure vs. Automata—Proceedings of the VIIIth Banff Higher Order Workshop. (G. Birtwistle and F. Moller, ed.). Springer-Verlag Lecture Notes in Computer Science, 1995.

CONFER-169 S. J. Gay, R. Nagarajan. *A typed calculus of synchronous processes.* In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995.

CONFER-170 S. J. Gay. *Linear types for communicating processes.* PhD thesis. University of London, 1995.

CONFER-171 C. L. Hankin, I. C. Mackie, R. Nagarajan *Theory and Formal Methods 1994:* Proceedings of the Second Imperial College Department of Computing Theory and Formal Methods Section Workshop Cambridge, September 1994. IC Press, 1995.

CONFER-172 F. Lamarche *Generalizing coherence spaces and hypercoherences.* In *Proceedings, Eleventh Conference on the Mathematical Foundations of Programming Semantics.* 1995.

CONFER-173 F. Lamarche *From Chu spaces to cpos.* 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. IC Press, 1995.

CONFER-174 F. Lamarche *Games Semantics for Full Popositional Linear Logic.* In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995.

CONFER-175 I. C. Mackie *The geometry of implementation.* PhD thesis. University of London, 1995.

CONFER-176 I. C. Mackie *The Geometry of Interaction Machine.* In *Proceedings, 22nd ACM Symposium on Principles of Programming Languages (POPL).* ACM Press, 1995.

CONFER-177 R. Nagarajan *Proceedings of the Fifth CONFER Workshop* Imperial College, London, October 1994. Imperial College Department of Computing Technical Report No. 95-6, February 1995.

CONFER-178 R. Nagarajan *Typed Concurrent Programs: Specification & Verification.* PhD thesis. University of London. To appear.

CONFER-179 D. Pavlović. *Convenient Categories of Processes and Simulations I: Modulo Strong Bisimilarity.* In *Proceedings, Sixth Conference on Category Theory and Computer Science.* 1995.

CONFER-180	A. Asperti C. Laneve	*Interaction Systems II: the practice of optimal reductions.* Theoretical Computer Science, to appear (expected in Vol.160, june 1996).
CONFER-181	A. Asperti C. Laneve.	*Relating λ-calculus translations in sharing graphs.* Proc. of the Second International Conference on Typed Lambda Calculi and Applications, TLCA'95, Edinburgh, Scotland. 1995.
CONFER-182	A. Asperti.	$\delta \circ ! \epsilon = 1$*: optimizing optimal λ-calculus implementations.* Proc. of the Sixth International Conference on Rewriting Techniques and Applications, RTA'95, Kaiserlautern, Germany. 1995.
CONFER-183	G. Boudol C. Laneve,	*Termination, deadlock and divergence in the λ-calculus with multiplicities.* Proc. of the Eleventh Conference on the Mathematical Foundations of Programming Semantics, MFPS'95, Electronic Notes in Computer Science n. 1, pp. 63 – 76, Springer-Verlag, 1995.
CONFER-184	C. Laneve.	*An Investigation on the Optimal Implementation of Processes.* Atti del V Convegno Italiano di Informatica Teorica, Ravello, 9-11 Novembre 1995.
CONFER-185	C. Fournet G. Gonthier,	*The reflexive CHAM and the join-calculus.* To be presented at POPL'96, Jan 96, St Petersburgh.
CONFER-186	Damien Doligez	*Conception, réalisation et certification d'un glaneur de cellules concurrent*, PhD thesis (in French), Université Paris 7, May 1995.
CONFER-187	Gilles Dowek Thérèse Hardin Claude Kirchner	*Higher-order unification via explicit substitutions*, LICS'95, Jul 95, San Diego.
CONFER-188	Paul-André Melliès,	*Typed λ-calculi with explicit substitutions may not terminate*, TLCA'95, Apr 95, Edinburgh.

CONFER-189 G. Boudol, C. Laneve — *The discriminating power of multiplicities in the λ-calculus*, INRIA Research Report 2441, December 1994, submitted for publication.

CONFER-190 G. Boudol, C. Laneve — *Termination, deadlock and divergence in the λ-calculus with multiplicities*, Proceedings of the 11th Conference on the Mathematical Foundations of Programming Semantics, Electronic Notes in Computer Science 1, 1995.

CONFER-191 G. Boudol, C. Laneve — *λ-Calculus, multiplicities and the π-calculus*, INRIA Research Report 2581, June 1995, submitted for publication.

CONFER-192 D. Sangiorgi — *Bisimulation in higher-order calculi*, INRIA Research Report 2508, 1995. Revised version of a paper appeared in the proceedings of the IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94), North Holland, 1994.

CONFER-193 D. Sangiorgi — *Lazy functions and mobile processes*, INRIA Research Report 2515, 1995, submitted for publication.

CONFER-194 D. Sangiorgi — πI: *A symmetric calculus based on internal mobility*, TAPSOFT'95, LNCS, 1995.

CONFER-195 D. Sangiorgi — *Internal mobility and agent passing calculi*, ICALP'95, LNCS, 1995.

CONFER-196 D. Sangiorgi — *On proof method for bisimulation*, Proceedings of the Conference on the Mathematical Foundations of Computer Science, LNCS, 1995. A revised version of the technical report ECS-LFCS-94-299 has been submitted for publication.

CONFER-197 D. Sangiorgi — *π-Calculus, internal mobility and agent-passing calculi*, INRIA Research Report 2539, 1995, submitted for publication.

CONFER-198	Ferrari, G. Montanari, U. Quaglia, P.	*The Weak Late π-calculus Semantics as Observation Equivalence*, to appear Proc. CONCUR'95, LNCS, 1995.
CONFER-199	Ferrari, G. Modoni, G. Quaglia, P.	*Towards a Semantic-based Verification Environment for the π-calculus*, To appear Fifth Italian Conference on Theoretical Computer Science, 1995.
CONFER-200	Gadducci, F. Montanari, U.	*SOS Contexts as Cells in Double-Categories*, submitted for publication, 1995.
CONFER-201	Gadducci, F. Miculan, M.	*Modal μ-types for Processes*, Proc. LICS'95, 1995.
CONFER-202	Montanari, U. Pistore, M.	*Checking Bisimilarity for Finitary π-calculus*, to appear Proc. CONCUR'95, LNCS, 1995.
CONFER-203	Ferrari, G. Montanari U., Quaglia, P.	*A π-calculus with Explicit Substitution.* To appear in Theoretical Computer Science.
CONFER-204	Ferrari, G. Montanari, U.	*Type Additive Concurrency*, (A revised version entitled "Dynamic Matrices and the Cost Analysis of Concurrent Programs" appeared in Proc. AMAST'95, LNCS 936, 1995).
CONFER-205	Corradini, A. Gadducci F. Montanari, U.	*Relating two categorical models of term rewriting*, in Proc. Rewriting Techniques and Applications, (RTA'95), LNCS 914, 1995.
CONFER-206	Montanari, U. Pistore, M.	*Concurrent Semantics for the π-calculus*, in Proc. MFPS'95, ENTCS 1, 1995.

CONFER-207 B. Lisper *Computing in Unpredictable Environments: Semantics, Reduction Strategies, and Program Transformations*
Research Report, Swedish Institute of Computer Science (SICS), Kista (1995).
(submitted)

CONFER-208 J. Parrow *Interaction diagrams.*
Accepted for publication in *Nordic Journal of Computing* (1995).

CONFER-209 J. Parrow, D. Sangiorgi *Algebraic Theories for Name-Passing Calculi.*
Information and Computation,
120(2):174–197 (1995).