Foreword

Computer science aims to explain the way computational systems behave for us. The notion of calculational process, or algorithm, is a lot older than computing technology; so, oddly enough, a lot of computer science existed before modern computers. But the invention of real stored-program computers presented enormous challenges; these tools can do a lot for us if we describe properly what we want done. So computer science has made immense strides in ways of *presenting* data and algorithms, in ways of manipulating these presentations themselves as data, in matching algorithm description to task description, and so on. Technology has been the catalyst in the growth of modern computer science.

The first large phase of this growth was in free-standing computer systems. Such a system might have been a single computer program, or a multi-computer serving a community by executing several single programs successively or simultaneously. Computing theorists have built many mathematical models of these systems, in relation to their purposes. One very basic such model – the λ-calculus – is remarkably useful in this role, even if it was designed by Alonzo Church around 1940.

The second phase of the growth of computer science is in response to the advent of computer networks. No longer are systems freestanding; they interact, collaborate and interrupt each other. This has an enormous effect on the way we think about our systems. We can no longer get away with considering each system as sequential, goal-directed, deterministic or hierarchical; networks are none of these. So if we confine ourselves to such concepts then we remain dumb if asked to predict whether a network will behave in a proper – or an improper – way; for example, whether someone logging in to his bank may (as happened recently) find himself scanning someone else's account instead of his own.

The present book is a rigorous account of a basic calculus which aims to underpin our theories of interactive systems, in the same way that the λ-calculus did for freestanding computation. The authors are two of the original researchers on the π-calculus, which is now over ten years old and has served as a focus for much theoretical and practical experiment. It cannot claim to be definitive; in fact, since it was designed it has become common to express ideas about interaction and mobility in variants of the calculus. So it has become a kind of workshop of ideas.

That's the spirit in which the book is written. Half the book analyses the constructions of the calculus, searching out its meaning and exploring its expressivity by looking at weaker variants, or by looking at various type disciplines. Enthusiasts about types in programming will be struck to find that π-calculus types don't just classify *values*; they classify *patterns of behaviour*. This reflects the fact that what matters most in mobile interactive systems is not values, but connectivity and mobility of processes. With or without types, the unifying feature is *behaviour*, and what it means to say that two different processes behave the same.

The later part of the book deals with two generic applications. One of these is classical; how the π-calculus can actually do the old job which the λ-calculus does in underpinning conventional programming. The other is modern; how the calculus informs one of the most important models of interaction, the *object-oriented* model. These applications bring together much of the theory developed earlier; together, they show that a small set of constructs, provided that they emphasize *interaction* rather than calculation, can still bring some conceptual unity to the greatly extended scope of modern computing.

This book has been a labour of love for the authors over several years. Their scholarship is immense, and their organisation of ideas meticulous. As one privileged to have worked closely with them both, it's a great pleasure to be able to recommend the result as a storehouse of ideas and techniques which is unlikely to be equalled in the next decade or two.

Robin Milner
Cambridge
February 2001

Preface

Mobile systems, whose components communicate and change their structure, now pervade the informational world and the wider world of which it is a part. But the science of mobile systems is yet immature. This science must be developed if we are properly to understand mobile systems, and if we are to design systems so that they do what they are intended to do. This book presents the π-calculus, a theory of mobile systems, and shows how to use it to express systems precisely and reason about their behaviour rigorously.

The book is intended to serve both as a reference for the theory and as an extended demonstration of how to use the π-calculus to express systems and analyse their properties. The book therefore presents the theory in detail, with emphasis on proof techniques. How to use the techniques is shown both in proofs of results that form part of the theory and in example applications of it.

The book is in seven Parts. Part I introduces the π-calculus and develops its basic theory. Part II presents variations of the basic theory and important subcalculi of the π-calculus. A distinctive feature of the calculus is its rich theory of types for mobile systems. Part III introduces this theory, and Part IV shows how it is useful for understanding and reasoning about systems. Part V examines the relationship between the π-calculus and higher-order process calculi. Part VI analyses the relationship between the π-calculus and the λ-calculus. Part VII shows how ideas from π-calculus can be useful in object-oriented design and programming.

The book is written at the graduate level and is intended for computer scientists interested in mobile systems. It assumes no prior acquaintance with the π-calculus: both the theory and the viewpoint that underlies it are explained from the beginning.

Although the book covers quite a lot of ground, several topics, notably logics for mobility, and denotational and non-interleaving semantics, are not treated at all. The book contains detailed accounts of a selection of topics, chosen for their interest and because they allow us to explore concepts and techniques that can also be used elsewhere. Each Part ends with some references to sources and additional notes on related topics. We have not attempted the arduous task of referring to all relevant published work. The references given provide starting points for a reader who wishes to go more deeply into particular topics. Sometimes, an element of arbitrariness in the choice of references was inevitable.

Many exercises are suggested to help appreciation of the material; the more difficult of them are marked with an asterisk. We intend to maintain a Web page for general information and auxiliary material about the book. At the time of writing, this page is located at

```
http://www-sop.inria.fr/mimosa/personnel/Davide.Sangiorgi/
        Book_pi.html
```

Acknowledgements Our greatest debt is to Robin Milner. The field that is the subject of this book was shaped by his fundamental work on CCS and in creating and developing the π-calculus. Further, we have both been privileged to have worked with Milner, and his influence on our approach to the subject and how to write about it are profound.

We thank the many colleagues – too many to mention here – with whom we have worked on or discussed π-calculus and related topics, and whose insights and comments have contributed to our understanding.

We are grateful to the following people for reading parts of a draft of the book and offering comments that helped us improve it: Michael Baldamus, Silvia Crafa, Cédric Fournet, Daniel Hirshkoff, Kohei Honda, Naoki Kobayashi, Giovanni Lagorio, Cédric Lhoussaine, Huimin Lin, Barbara König, Robin Milner, Julian Rathke, Vasco Vasconcelos, Nobuko Yoshida, and especially Marco Pistore.

We record our appreciation of the work of David Tranah and his colleagues at Cambridge University Press in guiding the book into print.

Finally, we thank Laurence Sangiorgi and Katharine Grevling for their encouragement, assistance, and patience during the seemingly interminable process of writing.

General Introduction

Mobile systems are everywhere. Palpable examples are mobile communication devices and the networks that span the Earth and reach out into Space. And less tangibly, there is mobile code and the wondrous weaving within the World Wide Web. An accepted science of mobile systems is not yet established, however. The development of this science is both necessary and challenging. It is likely that it will consist of theories offering explanations at many different levels. But there should be something basic that underlies the various theories.

This book presents the π-calculus, a theory of mobile systems. The π-calculus provides a conceptual framework for understanding mobility, and mathematical tools for expressing mobile systems and reasoning about their behaviours. We believe it is an important stepping-stone on the path to the science of mobile systems.

But what is mobility? When we talk about mobile systems, what are the entities that move, and in what space do they move? Our broad answer is based on distinguishing two kinds of mobility. In one kind, it is *links* that move in an abstract space of *linked processes*. For example: hypertext links can be created, can be passed around, and can disappear; the connections between cellular telephones and a network of base stations can change as the telephones are carried around; and references can be passed as arguments of method invocations in object-oriented systems. In the second kind of mobility, it is *processes* that move in an abstract space of linked processes. For instance: code can be sent over a network and put to work at its destination; mobile devices can acquire new functionality using, for instance, the Jini technology [AWO$^+$99]; and procedures can be passed as arguments of method invocations in object-oriented systems.

The π-calculus treats the first kind of mobility: it directly expresses movement of links in a space of linked processes. There are two kinds of basic entity in the (untyped) π-calculus: names and processes. Names are names of links. Processes can interact by using names that they share. The crux is that the data that processes communicate in interactions are themselves names, and a name received in one interaction can be used to participate in another. By receiving a name, a process can acquire a capability to interact with processes that were unknown to it. The structure of a system – the connections among its component processes – can thus change over time, in arbitrary ways. The source of the π-calculus's strength is how it treats scoping of names and extrusion of names from their scopes.

The second kind of mobility, where it is processes (or, more generally, computational entities built from processes) that move, can be made precise in several ways. We will examine a theory based on process-passing, the Higher-Order π-calculus, in Part V. In the book, calculi based on process-passing mobility are called *higher-order* calculi, and those, such as the π-calculus, that are based on name-passing mobility are called *first-order* calculi.

What can be said by way of comparison between name-passing and process-passing, and in particular for the precedence given in this book to first-order calculi? First, naming and name-passing are ubiquitous: think of addresses, identifiers, links, pointers, references. Secondly, as we will see, name-passing as embodied in the π-calculus is extremely expressive. In particular, Part V shows how process-passing calculi can be modelled in π-calculus. But name-passing is also more refined than process-passing. For by passing a name, one can pass partial access to a process, an ability to interact with it only in a certain way. Similarly, with name-passing one can easily model sharing, for instance of a resource that can be used by different sets of clients at different times. It can be complicated to model these things when processes are the only transmissible values. Thirdly, it was possible to work out the theory of π-calculus, and the theory is tractable. The theory of process-passing is harder, and important parts of it are not yet well understood. Its advancement has been, and can continue to be, greatly helped by the existence of the simpler theory of the π-calculus, in much the same way that the development of π-calculus was made much easier by prior work on theories of non-mobile processes.

The π-calculus does not explicitly mention location or distribution of mobile processes. The issue of location and distribution is orthogonal to the question of name-passing or process-passing. One can envisage worlds in which processes reside at locations and exchange links, worlds in which they exchange processes, and worlds in which they exchange both links and processes. It is too early to be able to distil the right concepts for treating distributed mobile systems and all the associated phenomena. We may hope, however, that ideas from π-calculus will continue to contribute to the search for these concepts and the development of theories based on them. At the time of writing, many theories treating location or distribution are being investigated. Some of them are extensions or close

relatives of the π-calculus, for instance the Distributed Join Calculus [FGL$^+$96] and the Distributed π-calculus [HR98b], while others are influenced by it, such as the Ambient Calculus [CG98] and Oz [Smo95]. All of these calculi and languages benefit from the theory of π-calculus.

In π-calculus, names are names of links. But what is a link? The calculus is not prescriptive on this point: *link* is construed very broadly, and names can be put to very many uses. This point is important and deserves some attention. For example, names can be thought of as channels that processes use to communicate. Also, by syntactic means and using type systems, π-calculus names can be used to represent names of processes or names of objects in the sense of object-oriented programming. (Part VII is about objects and π-calculus.) Further, although the π-calculus does not mention locations explicitly, often when describing systems in π-calculus, some names are naturally thought of as locations. Finally, some names can be thought of as encryption keys as, for instance, in the Spi calculus [AG97], which applies ideas from π-calculus to computer security.

The π-calculus has two aspects. First, it is a theory of mobile systems. The π-calculus has a rich blend of techniques for reasoning about the behaviour of systems, as we will see in the book. There has been some initial work on development of (semi-) automatic tools to assist in reasoning, but substantial challenges, both theoretical and practical, remain. Second, the π-calculus is a general model of computation, which takes interaction as primitive. The relationship between the π-calculus and the λ-calculus, which is a general model of computation that takes function application as primitive, is studied in depth in Part VI. Just as the λ-calculus underlies functional programming languages, so the π-calculus, or a variant of it, is the basis for several experimental programming languages, for instance Pict [PT00], Join [INR], and TyCO [VB98].

Of central concern in concurrency theory is when two terms express processes that have the same observable behaviour. The technical basis for the account of behavioural equivalence in the book is the notion of *bisimulation*. Bisimulation is one of the most stable and mathematically natural concepts developed in concurrency theory. It is at the heart of a successful theory of behavioural equivalence for non-mobile processes [Mil89], and it has important connections with non-well-founded sets [Acz88], domain theory [Abr91], modal logic [HM85], and final coalgebras [RT94]. Two π-calculus terms will be deemed to express the same behaviour if they are *barbed congruent*, that is, if no difference can be observed when the terms are put into an arbitrary π-calculus context and compared using the appropriate bisimulation game. Although the book concentrates on barbed congruence and other equivalences based on bisimulation, almost all of the results presented hold for other contextually-defined equivalences. The basic theory of the π-calculus is presented in Part I and Part II.

When employing π-calculus to describe a system, one normally follows a discipline that governs how names can be used. Such disciplines can be made explicit using *types*. This brings several benefits, notably the possibility of statically detecting many programming errors. Using types also has important consequences for the behaviour of processes and the techniques for reasoning about behaviour. Types are one of the most important differences between π-calculus and non-mobile process calculi. They are studied in Part III and Part IV.

A technical theme that recurs throughout the book is interpreting one calculus or language in another. There are several reasons for presenting and studying interpretations. First, in many cases, doing so addresses fundamental concerns relating to the expressiveness of calculi, and gives insight into how to use them for modelling systems. Secondly, showing how to express terms of one language or calculus in another often demonstrates effective use of important programming idioms. And thirdly, by studying properties of encodings, we show various proof techniques in action and illustrate ideas that are useful for analysing systems.

Robin Milner's invention of the Calculus of Communicating Systems (CCS) in the late 1970s was a watershed in the theory of concurrency [Mil80]. CCS inspired the field of process calculus, which continues to flourish some twenty years later. The π-calculus was created in the late 1980s by Milner, Joachim Parrow, and the second author [MPW89]. It evolved from CCS via an Extended Calculus of Communicating Systems introduced by Mogens Nielsen and Uffe Engberg [EN86].

The first book treating π-calculus was written by Milner [Mil99]. Based on an undergraduate course, it recapitulates CCS and then introduces π-calculus, with emphasis on examples and using the calculus to express systems. [Mil99] is an excellent introduction to concurrency theory in general and to CCS and the π-calculus in particular, and a reader unfamiliar with the field may find it easier to start with it. The present book covers more of the basic theory and in greater depth, and takes the reader further into the subject. We hope that it forms a natural complement to Milner's book.

Part I

The π-calculus