Mobile Processes: A Commented Bibliography

Silvano Dal Zilio

Mobile Processes: a Commented Bibliography

Silvano Dal Zilio

Microsoft Research

Abstract. We propose a short bibliographic survey of calculi for mobile processes. Contrasting with other similar exercises, we consider two related, but distinct, notions of mobile processes, namely *labile processes*, which can exhibit dynamic changes in their interaction structure, as modelled in the π-calculus of Milner, Parrow and Walker for example, and *motile processes*, which can exhibit motion, as modelled in the ambient calculus of Cardelli and Gordon. A common characteristic of the algebraic frameworks presented in this paper is the use of names as first class values and the support for the dynamic generation of new, fresh names.

1 Introduction

Process algebras have proved to be valuable mathematical tools to reason about the behaviour of concurrent and communicating systems. For more than ten years now, research has been conducted on semantics of higher-order processes that allow communication channels or even processes to be carried across by communications. Process calculi featuring the ability to dynamically create and exchange channel names are often referred to as *mobile*, a term popularised by the seminal introduction to the π-calculus [1], a prominent example of calculus with mobile processes.

1. Robin Milner, Joachim Parrow, David Walker: A Calculus of Mobile Processes, (parts I and II). Information and Computation **100**(1) (1992) 1–77
2. Robin Milner: Communicating and Mobile Systems: the Pi-Calculus. Cambridge University Press (2000)

Unfortunately, the term mobility is overloaded with meaning and the notion of mobility supported by the π-calculus encompasses only part of all the abstractions meaningful to mobility in a distributed system. For instance, the π-calculus does not directly model phenomena such as the distribution of processes within different localities, their migrations, or their failures.

As a matter of fact, the term mobility is related to two distinct notions. First, mobility is a property of systems undergoing frequent changes. In this situation, we say that the system is *labile*, by analogy with the labile compounds of a chemical reaction. Another notion associated with mobility is related to systems capable of changing their physical location. We say that these systems

F. Cassez, C. Jard, B. Rozoy and M. Ryan (Eds.): Modelling and Verification of Parallel Processes, LNCS 2067, pp. 207–223, 2001.

are *motile*, borrowing a term commonly used in biology to describe living form demonstrating movement by independent means.

Labile systems are well modelled by the π-calculus, which can represents systems that dynamically reorganize their communication structure throughout time. But as we said earlier, the π-calculus does not directly models motile systems. Based on this distinction, and to grasp the core meaning of location and process migration, process calculi with explicit locations, such as the *ambient calculus* of Cardelli and Gordon [3], have been recently proposed.

3. Luca Cardelli, Andrew D. Gordon: Mobile Ambients. In Proc. of FoSSaCS, Springer LNCS 1378 (1998) 140–155

This paper offers a short bibliographic survey of calculi for mobile processes. Contrasting with other similar exercises, we consider both labile and motile process calculi and we concentrate especially on the π-calculus, the ambient calculus, and some extension of π with distribution primitives.

This commented bibliography is not meant as an introduction to the π-calculus or the ambient calculus. In particular, we do not define these calculi, nor give hints at their syntax or semantics, as we cite several introductory materials that answer this purpose. Instead, this paper is an attempt to explain the differences and similarities between the two different notions of mobility that motivated the design of the π and ambient calculi, and to attract notice to interesting research problems arisen from the study of mobility.

We organize the rest of the paper as follows. Section 2 is concerned with labile process calculi, that is, calculi featuring mobility of names, and more particularly with the π-calculus. We suppose that the reader is familiar with process calculus and their equational theory, such as found in [2] for example. Familiarity with distributed programming languages and type systems would also be an advantage. In Section 3, we present calculi with distributed and migrating processes, namely motile calculi. We start by considering extension of the π-calculus with explicit locations and primitives for location failures and process migration. Then we look at the ambient calculus, an archetypal calculus for the mobility of computations.

Since bibliographical references make an important part of this paper, we provide them directly within the text. The complete list of references is provided at the end of this paper.

2 Mobility of Names

As explained in introduction of this paper, the notion of mobility in process calculi often refers to the capability for a process to exchange names as values. The idea of using channel names as data, together with the ability to generate fresh and unique names, is the basis on which the π-calculus is founded.

In his 1991 Turing award Lecture, Milner [4] gives an illuminating account on the concepts of interaction and naming. The study of the relation between

computation and naming is further developed in Gordon's survey on *nominal calculi* [5], that we discuss in the following section.

4. Robin Milner: Elements of Interaction. Communications of the ACM **36**(1) (1993) 78–89
5. Andrew D. Gordon: Notes on Nominal Calculi for Security and Mobility. In Proc. of FOSAD, Springer LNCS (2001), to appear

2.1 The Significance of Names

In his Turing award lecture, Milner argues that when one talks about mobility in a system of interacting agents, what really matters is not the mobility of the agents per se (an agent can even not exist), but rather the movement of the access paths to the agents. These access paths — channel names in the π-calculus or references in object-oriented terminology — are the key element we actually need to reason about.

As noted by Gordon [5], this emphasis on the role of naming in the comprehension of computational systems is not isolated. Indeed, contemporarily to the π-calculus definition, Needham [6] was advocating the importance of the notion of *pure names* in the formalization of distributed objects. In his own words, a pure name is "*nothing but a bit pattern that is an identifier, and is only useful for comparing for identity with other bit patterns — which includes looking up in tables in order to find other information.*" Compare this definition with the usage of (channel) names in the π-calculus, where a process can read from a named channel, emit in a named channel, or test the equality of two names.

Another example of research on the significance of pure names can be found in the nu-calculus of Pitts and Stark [7], a typed lambda-calculus extended with state in the form of dynamically generated names, in which pure names are introduced in order to models the effect of adding references to a functional language like ML.

6. Roger M. Needham: Names. In S. Mullender (ed.): Distributed Systems, Addison-Wesley (1989) 89–101
7. Andrew M. Pitts, Ian D. B. Stark: Observable Properties of Higher Order Functions that Dynamically Create Local Names, or: What's New? In Proc. of MFCS, Springer LNCS 711 (1993) 122–141

2.2 The π-Calculus

As the λ-calculus, from which many similarities can be drawn, the π-calculus is a parsimonious algebraic framework built from a reduced number of operators, yet expressive enough to model a wide range of computational systems and data structures.

A good introduction to the π-calculus can be found in Milner's tutorial book [2], a preliminary version of which exists as a LFCS research report [8]. Sangiorgi and Walker provide a reference book on the π-calculus theory [9], with

emphasis on proof techniques. Related papers, more targeted towards the application to concurrent and distributed programming languages, are Benjamin Pierce's introduction to the π-calculus for the engineer [10], and Peter Sewell's report on applied pi [11].

Beside these papers, the reader interested by a thorough presentation of the π-calculus will benefit from the notable bibliography on mobile processes compiled by Kohei Honda. Other interesting collections of resources are the bibliography and web pages on mobile process calculi maintained by Uwe Nestmann and Björn Victor and available at the following address: `http://move.to/mobility`.

8. Robin Milner: The Polyadic π-Calculus: a Tutorial. Technical Report ECS-LFCS-91-180, University of Edinburgh (1991)
9. Davide Sangiorgi, David Walker: The π-Calculus: a Theory of Mobile Processes. Cambridge University Press (2001)
10. Benjamin C. Pierce: Foundational Calculi for Programming Languages. In A. B. Tucker (ed.): Handbook of Computer Science and Engineering, CRC Press (1996)
11. Peter Sewell: Applied Pi – A Brief Tutorial. Technical Report 498, University of Cambridge (2000)
12. Kohei Honda: Selected Bibliography on Mobile Processes. Unpublished notes, available electronically (1998)
13. Uwe Nestmann, Björn Victor: Calculi for Mobile Processes: Bibliography and Web Pages. Bulletin of the EATCS **64** (1998) 139–144

The definition of the π-calculus is not steady and many different evolution of π can be found in the literature. The π-calculus is actually more a family of calculi than just a unique calculus. Such evolutions, briefly summarized in [11], include asynchronous, internal or receptive version of the π-calculus; extensions with primitive for testing the (in)equality of names; etc. In addition, several process calculi based on name-passing has been proposed: the fusion calculus of Parrow and Walker, a simplification of π with a more symmetric form of communication; the spi-calculus of Abadi and Gordon, an extension of π designed for the description and analysis of cryptographic protocols; the join-calculus of Fournet, Gonthier et al; the blue-calculus of Boudol; etc.

14. Joachim Parrow, Björn Victor: The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In Proc. of LICS, IEEE Computer Society Press (1998) 176–185
15. Martìn Abadi, Andrew D. Gordon: A Calculus for cryptographic protocols: the spi calculus. Information and Computation **148** (1999) 1–70
16. Cédric Fournet, Georges Gonthier: The Reflexive Chemical Abstract Machine and the Join-Calculus. In Proc. of POPL, ACM Press (1996) 372–385
17. Gérard Boudol: The π-Calculus in Direct Style. Higher-Order and Symbolic Computation **11** (1998) 177–208

There are also definitions of higher-order process calculi, like CHOCS for instance [18], where whole processes and not simply names can be exchanged

during communication. Interestingly enough, Sangiorgi proved that it is possible to encode a primitive for higher-order communication in π [19].

18. Bent Thomsen: Plain CHOCS. A Second Generation Calculus for Higher Order Processes. Acta Informatica **30**(1) (1993) 1–59
19. Davide Sangiorgi: From pi-Calculus to Higher-Order pi-Calculus – and Back. In Proc. of TAPSOFT, Springer LNCS 668 (1993) 151–166

In the remainder of this section, we follow the style of [12] and present a selection of articles following a somewhat arbitrary decomposition into topics.

2.3 Equational Theory and Properties of Processes

The operational semantics of concurrent systems are commonly defined using labelled transition systems. For example, the π-calculus semantics given in [1] is based on such a presentation. Nonetheless, the now conventional dynamic semantics of π is based on a reduction relation defined on top of a *structural congruence* relation that identifies processes up to elementary rearrangements. This presentation, first introduced in [20] and inspired by the chemical abstract machine of Berry and Boudol, allows for a simple and compact definition of the reduction rules in which the sub-processes having to interact appear in contiguous position. It also accounts for much of the elegance and simplicity of the π-calculus semantics.

20. Robin Milner: Functions as Processes. Mathematical Structures in Computer Science **2** (1992) 119–141
21. Gérard Berry, Gérard Boudol: The Chemical Abstract Machine. Theoretical Computer Science **96** (1992) 217–248

There exist several definitions of behavioural equivalences for π-calculus processes based on labelled-semantics, like for example the early and late bisimulations defined in [1]. See [11,12] for a general account on these equivalences and [22] for a good introduction to the semantics of concurrent process calculi.

In the reduction-based semantics, an interesting notion of equivalence is obtained using barbed equivalence. See also Honda and Yoshida reduction-based equivalence for an asynchronous process calculus [24].

The use of reduction-based equivalences is interesting because these kinds of equivalences are amenable to comparison between different calculi and semantics. Moreover, experience show that it is easier to define a reduction semantics for a calculi with explicit locations than to define (the equivalent) labelled transition semantics.

22. Robin Milner: Semantics of Concurrent Processes. In J. van Leeuwen (ed.): Handbook of theoretical computer science, Elsevier (1990) 1203–1241
23. Robin Milner, Davide Sangiorgi: Barbed Bisimulation. In Proc. of ICALP, Springer LNCS 623 (1992) 685–695

24. Kohei Honda, Nobuko Yoshida: On Reduction-Based Process Semantics. Theoretical Computer Science **152**(2) (1995) 437–486

Another general method to express and verify properties of processes is to use a logical system. In the case of CCS, for example, a modal logic known as Hennessy-Milner logic has been defined that can be used as an alternative characterization of bisimulation equivalences: two processes are equivalent if and only if they satisfies the same formulas. Milner, Parrow and Walker have extended this logic to mobile processes in [26]. Another interesting reference is the work of Mads Dam [27] on an extension of the modal μ-calculus used to define a proof system for π.

25. Matthew Hennessy, Robin Milner: Algebraic laws for Non-Determinism and Concurrency. Journal of the ACM **32** (1985) 137–161
26. Robin Milner, Joachim Parrow, David Walker: Modal Logics for Mobile Processes. Theoretical Computer Science **114**(1) (1993) 149–171
27. Mads Dam: Model Checking Mobile Processes. Information and Computation **129**(1) (1996) 35–51

2.4 The π-Calculus as a Programming Model

One of the major successes of the π-calculus is its use in the validation of concepts for concurrent programming languages; in almost the same manner that functional programming has been established from the computational model provided by the λ-calculus. In particular, programming languages directly based on the π-calculus have been proposed. Examples are Pict, developed by Pierce and Turner at the University of Edinburgh, and the join-calculus, a programming language based on the homonymous process calculus [16], developed at INRIA.

The theoretical foundations of these programming languages take support from the study of abstract machines for intermediate languages derived from π. For instance, Pict is based on an asynchronous version of π without choice operator [28]. See [29] for a study on how to encode choice. Likewise, the join-calculus is obtained as a restriction of π that makes it easier to implement in a distributed scenario [30].

28. David N. Turner: The Polymorphic Pi-Calculus: Theory and Implementation. PhD thesis, University of Edinburgh (1995)
29. Uwe Nestmann, Benjamin C. Pierce: Decoding Choice Encodings. In Proc. of CONCUR, Springer LNCS 119 (1996) 179–194
30. Cédric Fournet: Le join-calcul: un calcul pour la programmation répartie et mobile. PhD thesis, École Polytechnique (1998)

Other theoretical foundations for the design of concurrent programming languages are provided by studies on how to model various computational model in π. Fundamental studies include the encoding of the functional and object-oriented paradigm in the π-calculus — including actors and concurrent objects.

For instance, a complete tutorial on the different encoding of the λ-calculus in the π-calculus, extending the initial article of Milner on the encoding of functions as processes [20], can be found in [31]. See also Part VI of [9].

31. Davide Sangiorgi: Interpreting Functions as pi-Calculus Processes: a Tutorial. INRIA Research Report 3470 (1999)

The study of objects using process calculus (and process calculus techniques) is also particularly fruitful. For instance, the π calculus has been used in semantics of concurrent object-oriented programming languages [32] and to prove the validity of program transformations [33].

32. Cliff B. Jones: A π-Calculus Semantics for an Object-Based Design Notation. In Proc. of CONCUR, Springer LNCS 715 (1993) 158–172
33. Davide Sangiorgi: Typed pi-Calculus at Work: A Correctness Proof of Jones's Parallelisation Transformation on Concurrent Objects. Theory and Practice of Object Systems, **5**(1) (1999) 25–33

The π-calculus as also been used in interpretations of typed calculi of objects [34,35,36] and as a model for Obliq [37], an object-based programming language with distributed and mobile objects developed by Cardelli [38]. Other interesting works are concerned with the study of dedicated nominal calculi proposed as formalism to reason about concurrent objects [39,40,41].

34. Davide Sangiorgi: An Interpretation of Typed Objects Into Typed π-Calculus. Information and Computation **143**(1) (1998) 34–73
35. Josva Kleist, Davide Sangiorgi: Imperative Objects and Mobile Processes. In Proc. of PROCOMET, Chapman & Hall (1998)
36. Silvano Dal Zilio: An Interpretation of Typed Concurrent Objects in the Blue Calculus. In Proc. of IFIP TCS, Springer LNCS 1872 (2000) 409–424
37. Josva Kleist, Massimo Merro, Uwe Nestmann: Local pi-Calculus at Work: Mobile Objects as Mobile Processes. In Proc. of IFIP TCS, Springer LNCS 1872 (2000) 390–408
38. Luca Cardelli: A language with distributed scope. Computing Systems **8**(1) (1995) 27–59
39. Kohei Honda, Mario Tokoro: An Object Calculus for Asynchronous Communication. In Proc. of ECOOP, Springer LNCS 512 (1991) 133–147
40. Vasco T. Vasconcelos: Typed Concurrent Objects. In Proc. of ECOOP, Springer LNCS 821 (1994) 100–117
41. Andrew D. Gordon, Paul D. Hankin: A Concurrent Object Calculus: Reduction and Typing. In Proc. of HLCL, Elsevier ENTCS **16**(3) (1998)

2.5 Verification and Type Systems

In the case of the π-calculus, verification is related to the action of checking bisimilarity relations or checking whether a process satisfies some given logical specification, for instance expressed in a temporal or modal logic. An interesting

example of verification problem is given by the work of Abadi and Gordon on the spi-calculus, in which processes represent protocols and security properties are stated in terms of behavioural equivalences.

A substantial number of works related to verification for mobile processes concentrate on *finite control processes* [42], a class of processes that correspond to CCS finite state processes. See for instance the mobility Workbench, an automated tool for analysing π-calculus processes developed by Faron Moller and Björn Victor [43], and work in the HAL environment [45].

42. Mads Dam: On the Decidability of Process Equivalences for the π-Calculus. SICS Research Report 94-20 (1994)
43. Björn Victor, Faron Moller; The Mobility Workbench – a Tool for the π-Calculus. In Proc. of CAV, Springer LNCS 818 (1994) 428–440
44. Marco Pistore, Davide Sangiorgi: A Partition Refinement Algorithm for the π-Calculus. In Proc. of CAV, Springer LNCS 1102 (1996) 38–49
45. Gianluigi Ferrari, Stefania Gnesi, Ugo Montanari, Marco Pistore, Gioia Ristori: Verifying Mobile Processes in the HAL Environment. In Proc. of CAV, Springer LNCS 1427 (1998)

Due to the presence of recursion and dynamic generation of names, and in contrast with the situation in CCS, processes can exhibit an infinite-state behaviour. Therefore, model checking for mobile processes is related to the problem of verification of infinite-state and parameterised systems, a problem potentially undecidable. See the work of Esparza, among others, at the Technische Universität of Munich. An interesting proposition to automate the verification of "infinite" mobile processes (especially proofs of bisimilarity properties) relies on the use of theorem prover, such as Coq [46] or Isabelle [47].

46. Daniel Hirschkoff: A Full Formalisation of π-Calculus Theory in the Calculus of Constructions. In Proc. of TPHOL, Springer LNCS 1275 (1997)
47. Christine Röckl, Javier Esparza: Proof-Checking Protocols using Bisimulations. In Proc. of CONCUR, Springer LNCS 1664 (1999) 525–540

Another technique used to verify properties of processes relies on type systems. This approach of verification is particularly useful because it is generally simpler to type a process than to verify a property given in a modal logic.

Type systems in mobile calculus, like in functional calculus, are useful to prevent so-called run-time errors, but also to enforce security policies, to specify synchronization behaviours, or to validate program transformations and equivalences. For example, type systems have been used to attack problems related to information flow analysis, to prove the correctness of cryptographic protocols, or to prove the absence of deadlock in a program.

The first notion of type for π-calculus processes, called *sort*, is defined in Milner's tutorial [8]. Works on extension of the sorting system are well summarized in [12], and include, among others, the addition of (second-order) polymorphism, subtyping [48] and linearity [49]. More precise type analysis, based on the extension of the sorting system with modal operators, have also been proposed [50,51,52].

48. Benjamin C. Pierce, Davide Sangiorgi: Typing and Subtyping for Mobile Processes. Mathematical Structures in Computer Science **6**(5) (1996) 409–453
49. Naoki Kobayashi, Benjamin C. Pierce, David N. Turner: Linearity and the pi-Calculus. In Proc. of POPL, ACM Press (1996) 358–371
50. Nobuko Yoshida: Graph Types for Monadic Mobile Processes. In Proc. of FST & TCS, Springer LNCS 1180 (1996) 371–386
51. Gérard Boudol: Typing the Use of Resources in a Concurrent Calculus. In Proc. of ASIAN, Springer LNCS 1345 (1997) 239–253
52. Naoki Kobayashi: A Partially Deadlock-Free Typed Process Calculus. In Proc. of LICS, IEEE Computer Society Press (1997) 128–139

An original application of type systems to concurrent processes is in the definition of new behavioural equivalences. See [53] for an example. In this context, types are viewed as a way to establish a contract between a process and the possible contexts in which it can be executed, i.e., tested. Therefore, an equivalence defined using typed processes is coarser than its untyped counterpart and can be used to prove properties based on given assumption about the (execution) environment.

53. Benjamin C. Pierce, Davide Sangiorgi: Behavioral Equivalence in the Polymorphic pi-Calculus. In Proc. of POPL, ACM Press (1997) 242–255

2.6 Locality-Based Semantics

Before moving on to the next section of this presentation, concerned with motile process calculi, we briefly consider some early work on localities in CCS.

As pointed out in the introduction, mobility is strongly related to the concept of *locality*, which is the key notion used to represent where things are changing, or moving. In process calculi such as CCS, for example, it is possible to make the distributed structure of processes explicit by assigning different locations to each component of a parallel composition. Using this model of locations, it is possible to refine the usual behavioural equivalences defined between processes. An example of such equivalence is given in [55].

54. Ilaria Castellani, Matthew Hennessy: Distributed Bisimulations. Journal of the ACM, **36**(4) (1989) 887–911
55. Gérard Boudol, Ilaria Castellani, Matthew Hennessy, Astrid Kiehn: A Theory of Processes with Localities. Formal Aspects of Computing **6**(2) (1994) 165–200

Based on the same idea, Sangiorgi [56], and later Degano and Priami [57], have conducted related works in the context of mobile processes.

56. Davide Sangiorgi: Locality and Interleaving Semantics in Calculi for Mobile Processes. Theoretical Computer Science, **155**(1) (1996) 39–83

57. Pierpaolo Degano, Corrado Priami: Non Interleaving Semantics for Mobile Processes. Theoretical Computer Science **216**(1-2) (1999) 237–270

We mention this thread of research because it proposes an early treatment of the notion of locality in process calculi. Nevertheless, the notion of locality obtained with this approach is too syntactical and, consequently, is not adequate to deal with phenomenon such as the migration of processes. To get round this limitation, process calculi with an explicit notion of locality and explicit primitives for migrating processes have been proposed.

3 Mobility of Processes

The π-calculus is best used to model concurrent systems where interacting programs and processes can freely address each other and share resources. This is, for instance, the model commonly chosen to program systems of distributed objects over local area network, where a dedicated infrastructure can ensure the consistency of an abstract layer of services, like transparent routing of messages or failures recovery.

In programming over wide area network, such as the Internet, distribution introduces new issues of its own and breaks many postulates commonly assumed in concurrent system. See [58] for a good overview of the new problems faced in computations over large-scale network. In this paper, Cardelli points out that many events kept hidden in concurrent systems suddenly become apparent. Examples of such events, or *observable*, are the existence of explicit physical locations (because of the existence of latency in communication), the existence of virtual locations (because security policies can restrict the access to some protected resources), or the existence of failures.

Faced with these intrinsic differences, a new computing paradigm based on the migration of code or agents, instead of the migration of references, has been advocated.

58. Luca Cardelli: Abstractions for Mobile Computation. In J. Vitek and C. Jensen (eds.): Secure Internet Programming: Security Issues for Mobile and Distributed Objects, Springer LNCS 1603 (1999) 51–94

The existence of objective barriers to the free mobility of names makes the π-calculus an unsatisfactory choice for the modelling of computations over large-scale networks. But other limitations exist that require the definition of a new model, like the ability to represent containment or repudiation behaviours. For example, once a name has been communicated in π, it is impossible to withdraw the knowledge or the capabilities associated with this name to the receiving process. Likewise, even if the π-calculus has proved useful in modelling concurrent objects, it is difficult to use π to model groups of objects, a very important notion in component-based systems. Indeed, groups are important for defining set of objects sharing a common behaviour — the involvement in a transaction, a strategy regarding the concurrent access to resources, etc. — or a common

attribute — for example a given security policy. At the opposite, models based on explicit locations, such as the ambient calculus, provide an easy way to define "containment" or sharing properties.

The phenomena discussed in this introduction have little to do with (pure) concurrency or name mobility, and are therefore not directly captured by labile process calculi. Considering their significance in the understanding and the modelling of computations over large-scale networks, they nevertheless require an extensive theoretical treatment. For this reason, several motile process calculi have been defined, which directly include locations and primitives for moving processes.

3.1 Distributed Process Calculi

An early attempt to add an explicit notion of location to a process calculus is the work of Amadio and Prasad on π_1 [59]. In this paper, authors remark that while site failure is an essential aspect of distributed systems, it was not adequately modelled in the π-calculus. To model failures, they propose a process calculus in which processes are run at distributed locations. Their calculus provides operators to kill locations, to test the status of locations (ping), and to spawn processes at remote locations.

59. Roberto Amadio, Sanjiva Prasad: Localities and Failures. In Proc. of FST & TCS, Springer LNCS 880 (1994) 205–216
60. Roberto Amadio: An Asynchronous Model of Locality, Failure, and Process Mobility. In Proc. of COORDINATION, Springer LNCS 1282 (1997)

Riely and Hennessy have considered subsequent distributed versions of π. In [61,62] they describe a foundational language for specifying dynamically evolving networks of distributed processes, Dpi, which extends π with notions of remote execution and migration. Novel features of Dpi are that (channel) names are endowed with permissions and that the holder of a name may only use it in the manner allowed by these permissions. See also the model of distribution and failure proposed, independently, for the join-calculus [65].

In the works of Hennessy and Riely, the administration of permissions can be controlled using a type system: well-typed processes use their names in accordance with the permissions allowed by the types. For instance, types are used to guarantee that distributed agents cannot access the resources of a system without first being granted the capability to do so (the language studied allows agents to move between distributed locations and to augment their set of capabilities via communication with other agents). Another example of type system for distributed version of π is given in [66].

61. Matthew Hennessy, James Riely: Resource Access Control in Systems of Mobile Agents. In Proc. of HLCL, Electronic Notes in Theoretical Computer Science **16**(3) (1998)
62. Matthew Hennessy, James Riely: A Typed Language for Distributed Mobile Processes. In Proc. of POPL, ACM Press (1998) 378–390

63. Peter Sewell: Global/Local Subtyping and Capability Inference for a Distributed π-Calculus. In Proc. of ICALP, Springer LNCS 1443 (1998) 695–706
64. Nobuko Yoshida, Matthew Hennessy: Subtyping and Locality in Distributed Higher Order Processes. In Proc. of CONCUR, Springer LNCS 1664 (1999) 557–572
65. Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, Didier Rémy; A Calculus of Mobile Agents. In Proc. of CONCUR, Springer LNCS 1119 (1996) 406–421
66. Roberto Amadio, Gérard Boudol, Cédric Lhoussaine: The Receptive Distributed pi-Calculus. In Proc. of FST & TCS, Springer LNCS 1738 (1999) 304–315

3.2 The Ambient Calculus

Very recently, Cardelli and Gordon have proposed a new process algebra, the ambient calculus [3], for describing systems with mobile computations. In this calculus, processes may reside within a hierarchy of locations, called *ambients*. Each location is a cluster of processes and sub-ambients that can move as a group.

Ambients provide an interesting abstraction that combines, within the same theoretical framework, notions such as *mobile computations*, i.e., computations that can dynamically change the place where they are executed and are continuously active before and after movement (like agents), the *sites* where these computations happen: processor, router, etc. and the *mobility* of these sites, such as found with mobile, or even simply temporarily disconnected, computers, or in the crossing of administrative boundary, like applets crossing a firewall.

In the ambient calculus, each ambient has a name — the counterpart of a channel name in the π-calculus — used to define a set of possible capabilities, namely the capability of entering, of exiting or of opening an ambient. The result is a concise process calculus permitting to describe both the mobility and the security behaviours of a system using the same primitives. Instead of extending an existing process calculus with a hierarchical system of locations, Cardelli and Gordon have designed a calculus of locations and migration primitives sufficiently expressive to encode π.

An equational theory for the ambient calculus, as well as the proof of some algebraic laws, is given in [67]. Other works are related to the definition of type systems, like for instance type systems to guarantee that certain ambients remain immobile, or that the execution environment cannot dissolved certain ambients [68,69].

67. Luca Cardelli, Andrew D. Gordon: Equational Properties of Mobile Ambients. In Proc. of FoSSaCS, Springer LNCS 1578 (1999)
68. Luca Cardelli, Andrew D. Gordon: Types for Mobile Ambients. In Proc. of POPL, ACM Press (1999) 79–92

69. Luca Cardelli, Giorgio Ghelli, Andrew D. Gordon: Mobility Types for Mobile Ambients. In Proc. of ICALP, Springer LNCS 1644 (1999) 230–239

To define stronger and finer properties of processes, Cardelli and Gordon have also defined a new logic for the ambient calculus [70,71], which includes both temporal modalities, to specify the behaviour of processes after some reductions, and space modalities, to specify the behaviour of sub-processes at a given location. The modal logic for ambients has also been used as the basis for a query language on semistructured data [72].

70. Luca Cardelli, Andrew D. Gordon: Anytime, Anywhere: Modal Logics for Mobile Ambients. In Proc. of POPL, ACM Press (2000) 365–377
71. Davide Sangiorgi: Extensionality and Intensionality of the Ambients Logics. In Proc. of POPL, ACM Press (2001) 4–13
72. Luca Cardelli, Giorgio Ghelli: A Query Language Based on the Ambient Logic. In Proc. of ESOP, Springer LNCS (2001), to appear

Another definition of process calculus based on mobile location is the Seal calculus of Vitek and Castagna. An interesting property of this calculus is that it allows expressing directly the possibility to seize a capability given at a certain point, something that can only be modelled in an ambient calculus equipped with a type system enforcing a linear use of capabilities.

73. Jan Vitek, Guiseppe Castagna: Seal: A Framework for Secure Mobile Computations. In Internet Programming Languages, Springer LNCS 1686 (1999)

4 Summary

This paper loosely surveys ten years of research on mobile process calculus. Our choice was to concentrate on algebraic formalism based on the notion of naming, also called nominal calculi by Gordon [5], and we have therefore omitted other possible formalism like coordination languages (such as LINDA) or Hewitt's models of actors.

Although many work has already been achieved, there are still promising research developments to expect in the study of the concept of naming and interaction, notions at the core of Milner's action calculi, a unifying framework introduced to study various notions of concurrent interactive behaviour.

74. Robin Milner: Calculi for Interaction. Acta Informatica **33**(8) (1996) 707–737

Another promising research development is to extend the notion of naming, only used for processes at present, to the level of types or even logical formulas. Example of type systems based on pure names can be found in [50,51] and in the system of groups defined by Cardelli, Ghelli and Gordon for the ambient calculus [75], which has also been applied to the π-calculus [76,77]. A similar example, at the level of logic, can be found in the recent extension of the modal logic of ambients with a new quantifier to express the freshness of names [79], modelled after Gabbay and Pitts work [78].

75. Luca Cardelli, Giorgio Ghelli, Andrew D. Gordon: Ambient Groups and Mobility Types. In Proc. of IFIP TCS, Springer LNCS 1872 (2000) 332–347
76. Luca Cardelli, Giorgio Ghelli, Andrew D. Gordon: Secrecy and Group Creation. In Proc. of CONCUR, Springer LNCS 1877 (2000) 365–379
77. Silvano Dal Zilio, Andrew D. Gordon: Region Analysis and a π-calculus with Groups. In Proc. of MFCS, Springer LNCS 1893 (2000) 1–20
78. Murdoch J. Gabbay, Andrew M. Pitts: A New Approach to Abstract Syntax Involving Binders. In Proc. of LICS, IEEE Computer Society Press (1999) 214–224
79. Luca Cardelli, Andrew D. Gordon: Logical Properties of Name Restriction. In Proc. of FoSSaCS, Springer LNCS (2001), to appear

Acknowledgments

I would like to thank Uwe Nestmann and Peter Sewell for helpful comments. Luca Cardelli and Andy Gordon commented on a previous version of this text.

References

1. Robin Milner, Joachim Parrow, David Walker: A Calculus of Mobile Processes, parts I and II. Information and Computation **100** (1992) 1–77
2. Robin Milner: Communicating and Mobile Systems: the Pi-Calculus. Cambridge University Press (2000)
3. Luca Cardelli, Andrew D. Gordon: Mobile Ambients. In Proc. of FoSSaCS, Springer LNCS 1378 (1998) 140–155
4. Robin Milner: Elements of Interaction. Communications of the ACM **36**(1) (1993) 78–89
5. Andrew D. Gordon: Notes on Nominal Calculi for Security and Mobility. In Proc. of FOSAD, Springer LNCS (2001), to appear
6. Roger M. Needham: Names. In S. Mullender (ed.): Distributed Systems, Addison-Wesley (1989) 89–101
7. Andrew M. Pitts, Ian D. B. Stark: Observable Properties of Higher Order Functions that Dynamically Create Local Names, or: What's New? In Proc. of MFCS, Springer LNCS 711 (1993) 122–141
8. Robin Milner: The Polyadic π-Calculus: a Tutorial. Technical Report ECS-LFCS-91-180, University of Edinburgh (1991)
9. Davide Sangiorgi, David Walker: The π-Calculus: a Theory of Mobile Processes. Cambridge University Press (2001)
10. Benjamin C. Pierce: Foundational Calculi for Programming Languages. In A. B. Tucker (ed.): Handbook of Computer Science and Engineering, CRC Press (1996)
11. Peter Sewell: Applied Pi – A Brief Tutorial. Technical Report 498, University of Cambridge (2000)
12. Kohei Honda: Selected Bibliography on Mobile Processes. Unpublished notes, available electronically (1998)
13. Uwe Nestmann, Björn Victor: Calculi for Mobile Processes: Bibliography and Web Pages. Bulletin of the EATCS **64** (1998) 139–144

14. Joachim Parrow, Björn Victor: The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In Proc. of LICS, IEEE Computer Society Press (1998) 176–185
15. Martìn Abadi, Andrew D. Gordon: A Calculus for cryptographic protocols: the spi calculus. Information and Computation **148** (1999) 1–70
16. Cédric Fournet, Georges Gonthier: The Reflexive Chemical Abstract Machine and the Join-Calculus. In Proc. of POPL, ACM Press (1996) 372–385
17. Gérard Boudol: The π-Calculus in Direct Style. Higher-Order and Symbolic Computation **11** (1998) 177–208
18. Bent Thomsen: Plain CHOCS. A Second Generation Calculus for Higher Order Processes. Acta Informatica **30**(1) (1993) 1–59
19. Davide Sangiorgi: From pi-Calculus to Higher-Order pi-Calculus – and Back. In Proc. of TAPSOFT, Springer LNCS 668 (1993) 151–166
20. Robin Milner: Functions as Processes. Mathematical Structures in Computer Science **2** (1992) 119–141
21. Gérard Berry, Gérard Boudol: The Chemical Abstract Machine. Theoretical Computer Science **96** (1992) 217–248
22. Robin Milner: Semantics of Concurrent Processes. In J. van Leeuwen (ed.): Handbook of theoretical computer science, Elsevier (1990) 1203–1241
23. Robin Milner, Davide Sangiorgi: Barbed Bisimulation. In Proc. of ICALP, Springer LNCS 623 (1992) 685–695
24. Kohei Honda, Nobuko Yoshida: On Reduction-Based Process Semantics. Theoretical Computer Science **152**(2) (1995) 437–486
25. Matthew Hennessy, Robin Milner: Algebraic laws for Non-Determinism and Concurrency. Journal of the ACM **32** (1985) 137–161
26. Robin Milner, Joachim Parrow, David Walker: Modal Logics for Mobile Processes. Theoretical Computer Science **114**(1) (1993) 149–171
27. Mads Dam: Model Checking Mobile Processes. Information and Computation **129**(1) (1996) 35–51
28. David N. Turner: The Polymorphic Pi-Calculus: Theory and Implementation. PhD thesis, University of Edinburgh (1995)
29. Uwe Nestmann, Benjamin C. Pierce: Decoding Choice Encodings. In Proc. of CONCUR, Springer LNCS 119 (1996) 179–194
30. Cédric Fournet: Le join-calcul: un calcul pour la programmation répartie et mobile. PhD thesis, École Polytechnique (1998)
31. Davide Sangiorgi: Interpreting Functions as pi-Calculus Processes: a Tutorial. INRIA Research Report 3470 (1999)
32. Cliff B. Jones: A π-Calculus Semantics for an Object-Based Design Notation. In Proc. of CONCUR, Springer LNCS 715 (1993) 158–172
33. Davide Sangiorgi: Typed pi-Calculus at Work: A Correctness Proof of Jones's Parallelisation Transformation on Concurrent Objects. Theory and Practice of Object Systems, **5**(1) (1999) 25–33
34. Davide Sangiorgi: An Interpretation of Typed Objects Into Typed π-Calculus. Information and Computation **143**(1) (1998) 34–73
35. Josva Kleist, Davide Sangiorgi: Imperative Objects and Mobile Processes. In Proc. of PROCOMET, Chapman & Hall (1998)
36. Silvano Dal Zilio: An Interpretation of Typed Concurrent Objects in the Blue Calculus. In Proc. of IFIP TCS, Springer LNCS 1872 (2000) 409–424
37. Josva Kleist, Massimo Merro, Uwe Nestmann: Local pi-Calculus at Work: Mobile Objects as Mobile Processes. In Proc. of IFIP TCS, Springer LNCS 1872 (2000) 390–408

38. Luca Cardelli: A language with distributed scope. Computing Systems **8**(1) (1995) 27–59
39. Kohei Honda, Mario Tokoro: An Object Calculus for Asynchronous Communication. In Proc. of ECOOP, Springer LNCS 512 (1991) 133–147
40. Vasco T. Vasconcelos: Typed Concurrent Objects. In Proc. of ECOOP, Springer LNCS 821 (1994) 100–117
41. Andrew D. Gordon, Paul D. Hankin: A Concurrent Object Calculus: Reduction and Typing. In Proc. of HLCL, Elsevier ENTCS **16**(3) (1998)
42. Mads Dam: On the Decidability of Process Equivalences for the π-Calculus. SICS Research Report 94-20 (1994)
43. Björn Victor, Faron Moller; The Mobility Workbench – a Tool for the π-Calculus. In Proc. of CAV, Springer LNCS 818 (1994) 428–440
44. Marco Pistore, Davide Sangiorgi: A Partition Refinement Algorithm for the π-Calculus. In Proc. of CAV, Springer LNCS 1102 (1996) 38–49
45. Gianluigi Ferrari, Stefania Gnesi, Ugo Montanari, Marco Pistore, Gioia Ristori: Verifying Mobile Processes in the HAL Environment. In Proc. of CAV, Springer LNCS 1427 (1998)
46. Daniel Hirschkoff: A Full Formalisation of π-Calculus Theory in the Calculus of Constructions. In Proc. of TPHOL, Springer LNCS 1275 (1997)
47. Christine Röckl, Javier Esparza: Proof-Checking Protocols using Bisimulations. In Proc. of CONCUR, Springer LNCS 1664 (1999) 525–540
48. Benjamin C. Pierce, Davide Sangiorgi: Typing and Subtyping for Mobile Processes. Mathematical Structures in Computer Science **6**(5) (1996) 409–453
49. Naoki Kobayashi, Benjamin C. Pierce, David N. Turner: Linearity and the pi-Calculus. In Proc. of POPL, ACM Press (1996) 358–371
50. Nobuko Yoshida: Graph Types for Monadic Mobile Processes. In Proc. of FST & TCS, Springer LNCS 1180 (1996) 371–386
51. Gérard Boudol: Typing the Use of Resources in a Concurrent Calculus. In Proc. of ASIAN, Springer LNCS 1345 (1997) 239–253
52. Naoki Kobayashi: A Partially Deadlock-Free Typed Process Calculus. In Proc. of LICS, IEEE Computer Society Press (1997) 128–139
53. Benjamin C. Pierce, Davide Sangiorgi: Behavioral Equivalence in the Polymorphic pi-Calculus. In Proc. of POPL, ACM Press (1997) 242–255
54. Ilaria Castellani, Matthew Hennessy: Distributed Bisimulations. Journal of the ACM, **36**(4) (1989) 887–911
55. Gérard Boudol, Ilaria Castellani, Matthew Hennessy, Astrid Kiehn: A Theory of Processes with Localities. Formal Aspects of Computing **6**(2) (1994) 165–200
56. Davide Sangiorgi: Locality and Interleaving Semantics in Calculi for Mobile Processes. Theoretical Computer Science, **155**(1) (1996) 39–83
57. Pierpaolo Degano, Corrado Priami: Non Interleaving Semantics for Mobile Processes. Theoretical Computer Science **216**(1-2) (1999) 237–270
58. Luca Cardelli: Abstractions for Mobile Computation. In J. Vitek and C. Jensen (eds.): Secure Internet Programming: Security Issues for Mobile and Distributed Objects, Springer LNCS 1603 (1999) 51–94
59. Roberto Amadio, Sanjiva Prasad: Localities and Failures. In Proc. of FST & TCS, Springer LNCS 880 (1994) 205–216
60. Roberto Amadio: An Asynchronous Model of Locality, Failure, and Process Mobility. In Proc. of COORDINATION, Springer LNCS 1282 (1997)
61. Matthew Hennessy, James Riely: Resource Access Control in Systems of Mobile Agents. In Proc. of HLCL, Electronic Notes in Theoretical Computer Science **16**(3) (1998)

62. Matthew Hennessy, James Riely: A Typed Language for Distributed Mobile Processes. In Proc. of POPL, ACM Press (1998) 378–390
63. Peter Sewell: Global/Local Subtyping and Capability Inference for a Distributed π-Calculus. In Proc. of ICALP, Springer LNCS 1443 (1998) 695–706
64. Nobuko Yoshida, Matthew Hennessy: Subtyping and Locality in Distributed Higher Order Processes. In Proc. of CONCUR, Springer LNCS 1664 (1999) 557–572
65. Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, Didier Rémy; A Calculus of Mobile Agents. In Proc. of CONCUR, Springer LNCS 1119 (1996) 406–421
66. Roberto Amadio, Gérard Boudol, Cédric Lhoussaine: The Receptive Distributed pi-Calculus. In Proc. of FST & TCS, Springer LNCS 1738 (1999) 304–315
67. Luca Cardelli, Andrew D. Gordon: Equational Properties of Mobile Ambients. In Proc. of FoSSaCS, Springer LNCS 1578 (1999)
68. Luca Cardelli, Andrew D. Gordon: Types for Mobile Ambients. In Proc. of POPL, ACM Press (1999) 79–92
69. Luca Cardelli, Giorgio Ghelli, Andrew D. Gordon: Mobility Types for Mobile Ambients. In Proc. of ICALP, Springer LNCS 1644 (1999) 230–239
70. Luca Cardelli, Andrew D. Gordon: Anytime, Anywhere: Modal Logics for Mobile Ambients. In Proc. of POPL, ACM Press (2000) 365–377
71. Davide Sangiorgi: Extensionality and Intensionality of the Ambients Logics. In Proc. of POPL, ACM Press (2001) 4–13
72. Luca Cardelli, Giorgio Ghelli: A Query Language Based on the Ambient Logic. In Proc. of ESOP, Springer LNCS (2001), to appear
73. Jan Vitek, Guiseppe Castagna: Seal: A Framework for Secure Mobile Computations. In Internet Programming Languages, Springer LNCS 1686 (1999)
74. Robin Milner: Calculi for Interaction. Acta Informatica **33**(8) (1996) 707–737
75. Luca Cardelli, Giorgio Ghelli, Andrew D. Gordon: Ambient Groups and Mobility Types. In Proc. of IFIP TCS, Springer LNCS 1872 (2000) 332–347
76. Luca Cardelli, Giorgio Ghelli, Andrew D. Gordon: Secrecy and Group Creation. In Proc. of CONCUR, Springer LNCS 1877 (2000) 365–379
77. Silvano Dal Zilio, Andrew D. Gordon: Region Analysis and a π-calculus with Groups. In Proc. of MFCS, Springer LNCS 1893 (2000) 1–20
78. Murdoch J. Gabbay, Andrew M. Pitts: A New Approach to Abstract Syntax Involving Binders. In Proc. of LICS, IEEE Computer Society Press (1999) 214–224
79. Luca Cardelli, Andrew D. Gordon: Logical Properties of Name Restriction. In Proc. of FoSSaCS, Springer LNCS (2001), to appear