Ph. D. in Computer Science (XVII Ciclo) 2nd Year Report: Thesis Development

Damiano Macedonio

Dipartimento di Informatica, Università Ca' Foscari di Venezia
via Torino 155, 30172 Venezia, Italy
mace@dsi.unive.it

1 Introduction

In our daily life mobile computing resources moving in an out of other computing resources are very common. Prime examples are smart cards [27] used in Subscriber Identity Module (SIM) cards or next generation credit cards, moving from card issuers to card holders and in and out of mobile phones or automatic teller machines (ATMs). In a distributed environment, in general, a user often employs programs which are sent or fetched from different sites to achieve his/her goals. Such programs may be run as a code to do simple calculation task or as interactive parallel programs using resources located almost everywhere in the world. Accordingly, the ability to reason about correctness of the behaviour of concurrent systems holding and/or using such resources, as well as the need of design and implementation tools, will raise to an increasing prominent role.

This prefigure exciting future perspectives, but it poses enormous challenger to computer science, and it is likely to require the development of innovative paradigms for information processing and task coordination. In fact traditional correctness properties and methodologies for sequential systems are not more applicable in presence of distributed and mobile systems. The lack of any kind of central control, the continuously mutating topology of the network, the lack of reliable information, the absence of any intrinsically trustable object imply the necessity of designing new formal models for description of and the reasoning on properties of distributed resources. This necessity has been recently recognized by several authors; milestone papers on this subject are [16, 44].

In the global computing model of mobility, resources are shared and distributed over the network, and agents are not tied to any specific system resource or to any geographical or logical network location. They need permission to cross administrative domains and to execute on remote locations using local resources, outside their control, as well as resources belonging to the domain of origin. Resource access control aims at providing guarantees of safety and authorization. Safety corresponds to building safeguards against misuse of data leading to runtime failures: authorization provides an insurance that access to resources is granted only to principals that have obtained appropiate permissions. A reliable software is a prerequisite for the success of the global computing infrastructure.

Following the traditional approaches for the analysis of concurrent systems, properties of mobile processes and distributed resources can be expressed in

terms of semantics (e.g.*behavioural equivalences* [41]), logics [15, 45, 48, 54], or types [42, 47]. Although correctness and security issues motivated the studies of abstract models for distributed systems, there are still few works dealing with the verification of correctness or security properties of concurrent and mobiles systems.

We propose to study semantic characterizations of distributed systems which are suitable to analyze processes in dynamic environments. Our purpose is to specify a logical/formal tool which makes it easier to deal with concurrent and mobile systems. A logical formalism should simplify the definition of correctness and security properties for a distributed system. A logical formula defines a property which assumes meaning in a defined model. On the one hand a formula can detect a class of processes, the processes that enjoys that property [12]. On the other hand a formula can model directly the observed properties of resources in a distributed system [45, 48].

Moreover the logical framework helps in deriving new properties as well as connections between different characterization of process properties or resource distributions. Our purpose is to individuate a logical language which is able to describe the behaviour and spatial structure of concurrent systems, and thus it is a useful instrument in deriving correct systems in a compositional way. Candidate languages could be Spatial Logic [12, 13], which provides a powerful language to formally describe the structure of concurrent processes, Bunched Implication [45] or Separation Logic [4, 48], which provide a powerful language to describe the distribution of resources in distributed systems.

Our principal intention is to play on logic, to describe process and resources behaviour w.r.t mobility and in particular security. In fact security is a basic property for distributed systems [51]. To this aim we started by analyzing the definition of secure process in well known formalisms, such as CCS [41] and π calculi [44].

A possibly title for our project could be: "A logical framework to deal with concurrency and security properties".

2 State of the Art

2.1 Process Calculi

Among the approaches and theories for the modelling, analysis and verification of concurrent distributed systems, process algebras have received a lot of attention for their mathematical rigour and modelling flexibility. Their theory takes off over twenty years ago from the seminal CCS [41] and other calculi [3, 28, 32] and led to the emergence of important notions of behavioural equivalences that are now part of the common way of reasoning about concurrent systems. CCS was surpassed by the introduction of π-calculus [43, 44, 55], which introduces name mobility and, therefore, puts network topologies under the control of the processes themselves. Thus an extra expressiveness has been achieved.

As the focus of international research on concurrency moved towards system distributed on wide-area networks, the communications offered by π-calculus

became less than perfect a choice for foundational calculi, in fact π-calculus makes an abstraction on communications that it is not easily implemented in a truly distributed setting[1]. This led to the definitions of several versions of the π-calculus featuring asynchronous message passing [33, 56], and to the development of the relative semantics theories, including bisimulation [1], and testing equivalence [17, 35]. A further step towards a faithful modelling of distributed computation was focusing on migration and location failures, as in the distributed join-calculus [24] and in Dπ [30], which introduced process migration and access control. An original viewpoint was brought forward by the ambient calculus [16]. Ambients are administrative or physical boundaries that confine their contents (including executing threads) and carry them along when autonomously moving. Ambients introduce new concepts (e.g. traversing the network opening of ambients, subjective versus objective moves) and new challenges. As the full ambient calculus is often perceived as too general, several proposal aimed at simplifying it, especially concerned with the control of access [2, 36, 9].

A prominent line of research in the semantics of distributed systems is the development of types systems for process calculi [42, 47] or the definition of new substructural logics [25, 45, 48]. Our research is focused on the foundations of logics for mobile distributed systems.

2.2 Logics

The relationships between computation and logic are regarded as fundamental, as perceived through paradigms of programming such as *proof-as-programs* (Curry-Howard isomorphism, functional programming), *proofs-as-computations* (logic programming), and *proofs-as-processes* (concurrent programming). Accordingly, modelling of concepts, mechanisms and computations is approached by the researchers through logic using methods based on automatized construction of proofs and structural analysis in substructural and constructive logics.

Semi-structured data recently arise as central in the exchange of information in computer science but adequate models and logics are necessary in order to represent, manipulate and reason about such data. One difficulty is to provide models that well reflect the structures (see Section 2.1) and logics that are enough expressive to represent data properties and enough restricted to decide if a given model satisfies a formula and if some properties entail other properties. In this context, recent works focus on separation logics [4, 14, 34, 46, 48].

Separation logic was initially introduced by Reynolds-O'Hearn to support compositional reasoning about sequential programs which manipulate pointers. Afterwards O'Hearn proposed a Hoare-style methodology based on separation

[1] In π-calculus, processes interact by sending communication links to each other. The basic computational step is the transfer of a communication link between two processes; the recipient can then use the link for further interaction with other parties. This is hardly implementable on physical system: how can we move the links to a process? We can neither change the physical topology of the system nor handle channels as a physical address, which should be known by a central trusted autority.

logic for reasoning about partial correctness of *shared-memory parallel* programs which manipulate pointers.

Reynolds and O'Hearn gave an operational semantics for sequential programs, by using an abstract model of machine state in which a state comprise a *store* and a *heap*. They extended the familiar pre- and post-condition notation of Hoare-style partial correctness formulas with a *separating conjunction*, used in pre- and post-conditions to specify disjointness assumptions and guarantees. The separation among formulae can be seen, e.g., as the separation between integers and addresses in a structure with pointers. An integer is stored into a location and it is in someway unique: two integers with the same value in different locations are two separate resources. Pointers that refer to the same location are indistinguishable, so their number is irrelevant.

Separation logic introduced the novel logical operation $\varphi * \psi$, (separating conjunction) that asserts that φ and ψ are formulae that hold for *disjoint* portions of the addressable storage. The prohibition of sharing is built into the operation. The intuitionistic character of this logic implied a monotonicity property: an assertion true for some portion of the addressable storage would remain true for any extension of that portion, such as might be created by later storage allocation.

The logic of bunched implication, **BF** [45], generalizes the idea of separation by dealing not only with pointers, but also with distributed resources in general. The basic idea is to model directly the observed properties of resources and then to give a logical axiomatization. The very first model of the logic was very simple. **BF** requires the following properties of resources: a *set* R of resources, a *combination* $\bullet$ of resources with a *zero* resource e, and a *comparison* $\sqsubseteq$ between resources. Mathematically, this set-up is modelled with a *commutative preordered monoid* $\mathcal{R} = (R, \bullet, e, \sqsubseteq)$ in which $\bullet$, with unit e, is functional w.r.t. $\sqsubseteq$. By taking such a structure as an algebra of *worlds*, we obtain a Kripke-style semantics for **BF** which freely combines multiplicative (intuitionistic linear $\otimes$ and $\multimap$) and additive (intuitionistic $\wedge$, $\rightarrow$ and $\vee$). The key property of the semantics is the *sharing interpretation*.

The elementary semantics or the multiplicative conjunction

$$m \models \varphi_1 * \varphi_2 \text{ iff there are } n_1 \text{ and } n_2 \text{ such that } m \sqsubseteq n_1 \bullet n_2,\ n_1 \models \varphi_1,\ n_2 \models \varphi_2$$

is interpreted as follows: the resource m is sufficient to support $\varphi_1 * \varphi_2$ just in case it can be divided into resources n_1 and n_2 such that n_1 is sufficient to support φ_1 and n_2 is sufficient to support φ_2. The assertions φ_1 and φ_2 – think of them as expressing properties of programs – *do no share* resources. In contrast, in the semantics of the additive conjunction

$$m \models \varphi_1 \,\&\, \varphi_2 \text{ iff } m \models \varphi_1 \text{ and } m \models \varphi_2$$

the assertions φ_1 and φ_2 *share* the resource m.

BF is different form Girard's Linear Logic (**LL**), principally because implications. In **LL** there is only a native implication: the multiplicative one (which

defines the additive one by means of a modality). In **BF** there are two native and independent implications. The semantics of multiplicative implication

$$m \models \varphi \mathrel{-\!\!*} \psi \text{ iff for all } n \text{ such that } n \models \varphi,\ m \bullet n \models \psi$$

is interpreted as follows: the resources m is sufficient to support $\varphi \mathrel{-\!\!*} \psi$ – think of the proposition as (the type of) a function – just in case for any resource n which is sufficient to support φ – think of it as the argument of the function – the combination $m \bullet n$ is sufficient to support ψ. The function and its argument *do not share resources*. In contrast, in the semantics of additive implication

$$m \models \varphi \to \psi \text{ iff for all } n \sqsubseteq m, \text{ if } n \models \varphi, \text{ then } n \models \psi$$

the function and its argument *share* the resource n. For a simple example of resource as *cost*, let the monoid be given by the natural numbers with addition and unit zero, ordered by less than equal.

BI logic has recently been extended with a modality for locations [4]. Now it can be viewed as a separation and a spatial logic: the **BI**'s multiplicatives naturally introduce the notion of resource separation and the location modality allows to gather resources in some locations and thus introduce a notion of spatial representation or resources. In [4] a new data model is given, it is called resource tree, that is a labelled tree structure in which contain nodes are elements of a partially defined monoid of resources.

Next step for the development of **BI** should be the introduction of the mobility for the resources, in order to perform nomadic distributed systems.

On the other hand, Caires and Cardelli's Spatial Logic displays an active parallel line of development on reasoning about concurrent processes and semi-structured data [12–14]. Spatial Logic has been proposed with the aim of describing the behaviour and spatial structure of concurrent systems.

Spatial Logic tackles the problem from another point of view. On the one hand **BI** is a logic that originates form a simple resource model and now is approaching to more complex models. On the other hand Spatial Logic originates from an already complex model.

The spatial properties that this logic consider are essentially of two kinds: whether a system is composed of two or more subsystems (i.e. "Composition" of π-calculus), and whether a system restricts the use of certain resources to certain subsystems (i.e. "Restriction" of π-calculus).

A model of Spatial Logic is just π-calculus, but the idea can be easily extended to other calculi, such as ambient calculi with locations. A formula in Spatial Logic describes a property of a particular concurrent system at a particular time; therefore it is modal both in space and in time.

In conclusion these logical calculi, can talk about fine details of process structure. This is what is required if we want meaningfully describe the distribution of processes and the use of resources over a network. We are considering both of them in order to detect the right logical formalism suitable to describe a real distributed system.

2.3 Security as Noninterference

Protecting the confidentiality of manipulated information is one of the most important problems in distributed systems. There is little assurance that concurrent systems protect data confidentiality and integrity. Analysing the confidentiality properties of a distributed system is difficult even when insecurity arises only from unintentional errors in the design or implementation. Additionally modern systems commonly incorporate untrusted, possibly malicious host or code, making assurance of confidentiality more difficult.

The standard way to protect confidential data is *access control*: some privileges are required in order to access files or objects containing the confidential data. Access control checks place restrictions on the *release* of information but not its *propagation*. Once information is released from its container, the accessing program may, through error or malice, improperly transmit the information in some form. To ensure that information is used only in accordance with the confidentiality policies, it is necessary to analyse how information flows within the using program. The analysis must show that information controlled by a confidentiality policy cannot flow to a location where that policy is violated.

Information flow security aims at guaranteeing that no high level (confidential) information is revealed to users running at low levels [19, 23, 40, 51, 57], even in the presence of any possible malicious process. An early attempt to formalize the absence of information flow was the concept of *noninterference* proposed in the seminal paper by Goguen and Meseguer [26], and further developed in [10, 19, 20, 31, 38, 50, 52]. Intuitively, to establish that information does not flow from high to low it is sufficient to establish that high behavior has no effect on what low level users can observe, i.e., the low level view of the system is independent of high behavior.

Noninterference policy has been implemented in various formalisms such as *Security Process Algebra* (SPA for short) [18, 19, 39], or π-calculus [29], or *ambient calculus* [10, 36].

SPA is a variation of Milner's CCS [41] where the set of visible actions is partitioned into high level actions (H) and low level ones (L), in order to specify multilevel systems. Noninterference among processes has been formalized in [18, 19] with the definition of *Bisimulation-based Non Deducibility on Compositions* (BNDC). The BNDC security property aims at guaranteeing that no information flow from the high to the low level is possible, even in the presence of malicious processes. The main motivation is to protect a system also from internal attacks, which could be performed by the so called *Trojan Horse* programs. Property BNDC is based on the idea of checking the system against all high level potential interactions, representing every possible high level malicious program.

If has been shown, [22], that BNDC property is not strong enough to analyse systems in dynamic execution environments. For instance, if code mobility is allowed, a program could migrate to different host in the middle of its computation. In this setting we have to guarantee that every reachable state of the process is secure. Another interesting example is the execution of an applet on a Java Card, where an attacker could try to bring the card in an unstable (in-

secure) state by powering off the card in the middle of applet computation. To deal with these situations it has been introduced the security property named P_BNDC [5, 8, 21, 22, 49] that requires that every state reachable by the process has to be secure.

Noninterference is given treatment also in π-calculus, e.g. [29], by using types. Numerous typing systems have been developed for this language. Most are based on judgments of the form $\Gamma \vdash P$, indicating that the process P is well-typed with respect to the security policy Γ, which associates capabilities with the free channel names of P. Usually these capabilities are of the form:

$\mathtt{r}\langle T\rangle$: ability to read values of type T form a channel (*read capability*);
$\mathtt{w}\langle T\rangle$: ability to write values of type T to a channel (*write capability*).

In addition, a complete lattice SL of security levels is associated to obtain *security types*. By varying the precise definition of a security type we can either implement resource access control methodologies, or ensure forms of noninterference.

Among the various techniques used for defining security properties we focused our attention on the proof-theoretical approach, such as in [19], which verifies that the behaviour of a given process conforms to a given definition.

Now our aim is to translate the noninterference notion of SPA into π-calculus in order to study the relations with the noninterference definitions of π-calculus.

3 Preliminary Results

Our purpose is to studying semantic characterizations of distributed systems which are suitable to analyze processes in dynamic environments. We aim at specifying a logical/formal tool which makes it easier to deal with concurrent or mobile systems. A logical formalism should simplify the definition of correctness and security properties for a distributed system.

We first considered a basic property for distributed systems: security, especially noninterference. So we started by analyzing the definition of secure process in well known formalisms, such as CCS [41] and π-calculus [44].

In [6] we consider information flow security in a multilevel system, which aims at guaranteeing that no high level information is revealed to low level users, even in the presence of any possible malicious process. Our work starts from the observation that this requirement could be too demanding when some knowledge about the environment (context) in which the process is going to run is available. To deal with these situations we introduce the notion of *secure contexts for a class of processes*.

A context is a process with a variable subprocess (a hole) that can be replaced by any process, in order to characterize the environments in which processes are evolving. The notion of secure context is parametric with respect to both the observation equivalence and the operation used to characterize the low level behavior of a process. We believe that a "secure" context is a context which

cannot change in unpredictable ways, but follows some predetermined rules. These behaviuoral constraints are reflected in the structure of the context itself.

We mainly analyze the cases of bisimulation and trace equivalence. We describe how to build secure contexts in these cases and we show that two well-known security properties, named *BNDC* and *NDC*, are just special instances of our general definition.

Contexts formalize systems with unspecified components. In general, in the design process of distributed systems we may have to replace abstract specifications of components by more concrete specifications, thus providing more detailed design information. This well-known approach is often referred to as *action refinement*. In [7] we study the relationships between action refinement, compositionality, and information flow security within the Security Process Algebra (SPA). In particular, we first formalize the concept of action refinement in terms of context composition. We study the compositional properties of our notion of refinement and provide conditions under which information flow security properties expressed in terms of bisimulation are preserved through action refinement.

The results reported in [8] are our first step towards π-calculus. In fact in [8] we consider CCS with another form of recursion expressed using the replication operator (!) instead of constant definitions. Replication is the main recursion operator in the π-calculus, where it has the same expressive power as constant definitions [55]. As recently proved in [11], replication cannot supplant recursion in CCS. We show that the class of *P_BNDC* processes is compositional with respect to the replication operator. This allows us to define a proof system which provides a very efficient technique for the stepwise development and the verification of recursively defined *P_BNDC* processes. Moreover, we prove a partial compositionality of *P_BNDC* with respect to constant definitions, i.e., we identify a class of constant definitions which can be safely added to the language with replication and treated by an extended proof system.

4 Future Work Organization

Our main goal is specifying a logical/formal tool which makes it easier to deal with concurrent and mobile systems. Accordingly, here we state the work program for the next future. Our work is roughly divided by objectives, though there are tasks to be accomplished in parallel.

Objective 1. Starting from CCS, consider other formalisms such as π-calculus, $D\pi$, or ambients in order to extend the properties studied in CCS. In particular transfer into more structured calculi the notions of noninterference of SPA, such as secure contexts, and try to express them within a logical framework.

Objective 2. Develop a logical framework suitable to deal with completely mobile distributed systems, which exploit both the spatial characteristics and interconnections of objects. This intention could be reach by following two complementary strategies:

– Extend a separation logic, such as **BI**, in order to deal with mobile distributed resources. Consider the model with location for **BI** [4] and introduce the mobility. This could lead to an ambient model for **BI**.
– Consider the central kernel of Spatial Logic and introduce a notion of process mobility. In fact Spatial Logic has been introduced for the purpose of describing communicating processes in a distributed environment. Spatial Logic is a classical logic not properly concentrated on locality of reasoning, which uses marriage of substructural conjunction and full boolean connectives. Hence it cannot catch the hint that a constructive logic, such as **BI** or Intuitionistic Linear Logic, can appreciate. On the other and, a strength for Spatial Logic is that its model is more accurate and more closed to a real distributed system than the resource model of Separation logic.

Our research is headed also toward foundations of logic. In particular Basic Logic [53], a substructural logic which has been introduced with foundational purposes, and in this sense it is the core for every substructural logic, such as **BI**. It should be interesting to study the inner relation between **BI** and Basic Logic. In fact it has recently been proved that Basic Logic admits a mathematical model [37]: a *relational monoid*, just a commutative monoid with a binary relation. Such a model is very close to **BI**'s resource model.

References

1. R. Amadio, I. Castellani, and D. Sangiorgi. On Bisimulations for the Asynchronous π-calculus. *Theoretical Computer Science*, 195(2):291–324, 1998.
2. F. Barbanera, M. Bugliesi, M. Dezani, and V. Sassone. A Calculus of Bounded Capacities. In *Proc. of Advances in Computing Science, 9th Asian Computing Science Conference, ASIAN'03*, LNCS, 2003. To appear.
3. J. Bergstra and W. Klop. Process Algebra for Synchronous Communication. *Information and Computation*, 60, 1984.
4. N. Biri and D. Galmiche. A Separation Logic for Resource Distribution (Extended Abstract). In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, LNCS. Springer-Verlag, 2003. To appear.
5. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. A Proof System for Information Flow Security. In M. Leuschel, editor, *Logic Based Program Development and Transformation*, volume 2664 of *LNCS*. Springer-Verlag, 2003. To appear.
6. A. Bossi, D. Macedonio, C. Piazza, and S. Rossi. Secure Contexts for Confidential Data. In *the 16th IEEE Computer Security Foundations Workshop (CSFW 2003)*, pages 14–28. IEEE Computer Society Press.
7. A. Bossi, D. Macedonio, C. Piazza, and S. Rossi. Compositional action refinement and information flow security. Technical Report CS-2003-13, Dipartimento di Informatica, Università Ca' Foscari di Venezia, 2003.
8. A. Bossi, D. Macedonio, C. Piazza, and S. Rossi. Information Flow Security and Recursive Systems. In C. Blundo and C. Laneve, editors, *Proc. of the Italian Conference on Theoretical Computer Science (ICTCS 2003)*, volume 2481 of *LNCS*. Springer-Verlag, 2003.

9. M. Bugliesi, G. Castagna, and S. Crafa. Boxed Ambients. In *Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACS'01)*, volume 2215 of *LNCS*, pages 38–63. Springer-Verlag, 2001.
10. M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication Interference in Mobile Boxed Ambients. In M. Agrawal and A. Seth, editors, *Proc. of Int. Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'02)*, volume 2556 of *LNCS*, pages 71–84. Springer-Verlag, 2002.
11. N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. Recursive Definitions in Channel Based Calculi. In *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP'03)*, volume 2719 of *LNCS*.
12. L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). In N. Kobayashi and B.C. Pierce, editors, *Proc. of 4th International Symposium on Theoretical Aspects of Computer Software (TACS'01)*, volume 2215 of *LNCS*, pages 1–37. Springer-Verlag, Berlin, 2001.
13. L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part II). In L. Brim, P. Jancar, M. Kretinsky, and A. Kucera, editors, *Proc. of the 13th International Conference on Concurrency Theory (CONCUR'02)*, volume 2421 of *LNCS*, pages 209–225. Springer-Verlag, Berlin, 2002.
14. C. Calcagno, L. Cardelli, and A. Gordon. Deciding Validity in a Spatial Logic for Trees. In *ACM Sigplan Workshop on Types in Language Design and Implementation (TLDI'03)*. ACM Press, 2003.
15. L. Cardelli and A.D. Gordon. Anytime, Anywhere. Modal Logics for Mobile Ambients. In *Proc. of the 27th ACM Symposium on Principles of Programming Languages*, pages 365–377. ACM Press, 2000.
16. L. Cardelli and A.D. Gordon. Mobile Ambients. *Theoretical Computer Science, Special Issue on Coordination*, 2000.
17. I. Castellani and M. Hennessy. Testing Theories for Asynchronous Languages. In V. Arvind and R. Ramanujam, editors, *Proc. of Int. Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'98)*, volume 1530 of *LNCS*, pages 90–101. Springer-Verlag, 1998.
18. R. Focardi. *Analysis and Automatic Detection of Information Flows in Systems and Networks*. Phd thesis, ublcs-99-16, University of Bologna, 1999.
19. R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*, pages 331–410. Springer-Verlag, 2001.
20. R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 744–755. Springer-Verlag, 2000.
21. R. Focardi, C. Piazza, and S. Rossi. Proof Methods for Bisimulation based Information Flow Security. In A. Cortesi, editor, *Proc. of Int. Workshop on Verification, Model Checking and Abstract Interpretation (VMCAI'02)*, volume 2294 of *LNCS*, pages 16–31. Springer-Verlag, 2002.
22. R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 307–319. IEEE Computer Society Press, 2002.
23. S. N. Foley. A Universal Theory of Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 116–122. IEEE Computer Society Press, 1987.

24. C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, and D. Rémy. A Calculus of Mobile Agents. In *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR'96)*, volume 1119 of *LNCS*, pages 406–421. Springer-Verlag, 1996.
25. J.Y. Girard. Linear Logic. *Theoretical Computer Science*, 50(1):1–120, 1987.
26. J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.
27. U. Hansmann, M.S. Nicklous, Thomas Schäck, and F. Seliger. *Smart Card Application Development Using Java*. Springer, 2000.
28. M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.
29. M. Hennessy. The Security Picalculus and Non-interference. *Journal of Logic and Algebraic Programming*, 2003. To appear.
30. M. Hennessy and J. Riely. Resource Access Control in Systems of Mobile Agents. In U. Nestmann and B.C. Pierce, editors, *High-Level Concurrent Languages (HLCL'98)*, volume 16.3, pages 3–17. Elsevier Science Publishers, 1998.
31. M. Hennessy and J. Riely. Information Flow vs. Resource Access in the Asynchronous Pi-calculus. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 24(5):566–591, 2002.
32. C.A.R. Hoare. *Communicating Sequential Processes*. Pretience Hall, 1985.
33. K. Honda and M. Tokoro. An Object calculus for Asynchronous Communication. In P. America, editor, *Proc. of the European Conference on Object-Oriented Programming (ECOOP'91)*, volume 512 of *LNCS*, pages 133–147. Springer-Verlag, 1991.
34. S. Ishtiaq and P.O'Hearn. BI as an Asseertion Language for Mutable Data Structures. In *Proc. of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*, pages ??–?? ACM Press, 2001.
35. C. Laneve. May and Must Testing in the Join-Calculus. Technical Report UBLCS 96-04, University of Bologna, 1996.
36. F. Levi and D. Sangiorgi. Controlling Interferences in Ambients. In *Proc. of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00)*, pages 352–364. ACM Press, 2000.
37. D. Macedonio and G. Sambin. Relational Semantics for Basic Logic. *The Journal of Symbolic Logic*. To appear.
38. H. Mantel. Possibilistic Definitions of Security - An Assembly Kit -. In *Proc. of the 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 185–199. IEEE Computer Society Press, 2000.
39. F. Martinelli. *Formal Methods for the Analysis of Open Systems with Applications to Security Properties*. Phd thesis, University of Siena, 1999.
40. J. McLean. Security Models and Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 180–187. IEEE Computer Society Press, 1990.
41. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
42. R. Milner. Sorts in the π-calculus (Extended Abstract). In E. Best and G. Rozenberg, editors, *Proc. of the 3rd Workshop on Concurrency and Compositionality*, volume 191 of *GMD-Studien*. GMD, Bonn, 1991. Also avaible as Report 6/91 from University of Hildesheim.
43. R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.
44. R. Milner, J. Parrow, and D. Walker. Calculus of Mobile Processes, parts I and II. *Information and Computation*, 100(1):1–77, 1992.

45. P.W. O'Hearn and D. Pym. The Logic of Bunched Implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.
46. P.W. O'Hearn, J. Reynolds, and H. Yang. Local Reasoning about Programs that Alter Data Structures. In *Proc. of the 15th Int. Workshop on Computer Science Logic (CSL'01)*, volume 2142 of *LNCS*, pages 1–19. Springer-Verlag, 2001.
47. B. Pierce and D. Sangiorgi. Typing and Subtyping for Mobile Processes. *Mathematical Structures in Computer Sciences*, 6(5):409–454, 1996.
48. J. Reynolds. Separation Logic: a Logic for Shared Mutable Data Structures. In *Proc. of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS'02)*, pages 55–74. IEEE Computer Society Press, 2002.
49. P.Y.A Ryan. Mathematical Models of Computer Security. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*, pages 1–62. Springer-Verlag, 2001.
50. P.Y.A. Ryan and S. Schneider. Process Algebra and Non-Interference. *Journal of Computer Security*, 9(1/2):75–103, 2001.
51. A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communication*, 21(1):5–19, 2003.
52. A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 200–215. IEEE Computer Society Press, 2000.
53. G. Sambin, G. Battilotti, and C. Faggian. Basic Logic: Reflection, Symmetry, Visibility. *Journal of Symbolic Logic*, 65:979–1013, 2000.
54. D. Sangiorgi. Extensionality and Intensionality of the Ambient Logic. In *Proc. of the 28th Symposium on Principles of Programming Languages (POPL'01)*. ACM Press, 2001.
55. D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge Universtity Press, 2001.
56. P. Sewell and P.T. Wojciechowsky. Nomadic Pict: Language and Infrastructure Design for Mobile Agents. *IEEE Concurrency*, 8(2), 2000.
57. G. Smith and D. M. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'98)*, pages 355–364. ACM Press, 1998.