Twenty years on: Reflections on the CEDISYS project. Combining true concurrency with process algebra.

Gérard Boudol[1], Ilaria Castellani[1], Matthew Hennessy[2], Mogens Nielsen[3], and Glynn Winskel[4]

[1] INRIA, 2004 route des Lucioles, 06902 Sophia Antipolis Cedex, France
[2] Department of Computer Science, O'Reilly Institute, Trinity College, Dublin 2, Ireland
[3] Department of Computer Science, University of Aarhus, IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark
[4] Computer Laboratory, University of Cambridge, William Gates Building, JJ Thomson Avenue, Cambridge CB3 0FD, UK

Abstract. We recall some memories of the Esprit Basic Research Action CEDISYS, a small, well-focussed and fruitful project which brought together researchers at the meeting point of true concurrency and process algebra, in the period 1988-1991. The project was initiated and effectively animated by Ugo Montanari, a passionate and long-time advocate of both these approaches to the semantics of concurrency.

1 Genesis of the project

Twenty years after its start, we revisit some of the results of the CEDISYS project, trying to place them in their historical context and to emphasise their impact on later work. The Basic Research Action CEDISYS (Compositional Distributed Systems) was funded by the European Community under the Esprit programme, between January 1988 and December 1991. The project involved four institutions: the three Computer Science Departments of the Universities of Aarhus (Denmark), Pisa (Italy), and Sussex (United Kingdom), and the INRIA Research Unit of Sophia Antipolis (France). The action was coordinated by Ugo Montanari.

The CEDISYS project did not come about by chance. It emerged quite naturally from pre-existing interactions among the partners, and from their increasing convergence of interests and approaches. The Aarhus partners were experts in domain theory and true concurrency models and had been among the founders of Event Structures, a nonsequential model of computation closely related to domains. The Sussex and Sophia partners were process calculi specialists, with a strong interest in true concurrency semantics. As for the Pisa site, Ugo Montanari had been from the very start an enthusiastic advocate of both Milner's calculus CCS and of true concurrency models, whereas other members of the group had been active researchers in the process calculi community. Finally, two

younger members of the project had been both Ugo Montanari's Master students in Pisa and Matthew Hennessy's PhD students in Edinburgh (this tradition of mobility among the sites was to be maintained during and after the project).

It was undoubtedly Ugo's merit to have detected the potential of bringing together these particular partners, and to propose a well-focussed project whose objectives could be easily shared by all. In retrospect, it is clear that the initial conditions were there for the project to become operational very quickly. Indeed, in the three years of its activity, the project turned out to be both lively and cohesive, and, in our view, also very productive.

In the remaining sections we shall try to highlight some of the project's results, without aiming in any way at an exhaustive account (the interested reader is referred for more details to the project's final report [87]).

2 Goal and context

In this section, we recall the goal of the project and point out its connection to previous and contemporary research. In the final CEDISYS report [87], the overall objective of the project is described as follows:

"developing a fundamental understanding of the nature of concurrency and providing a formal framework useful for describing concurrent and distributed systems. This formal framework should support the specification and development of such systems and should lead to methodologies for proving systems correct and more generally for deriving their properties".

More specifically, the project aimed at *"a theory of concurrency where the distributed nature of processes is properly taken into account".* The plan was to *"compare existing formalisms, develop models, languages and logics with compositionality and abstraction capabilities"*, as well as to *"experiment with techniques and tools for supporting the implementation of the proposed formalisms".*

At the time when the project was proposed, two main approaches to the theory of concurrency had been investigated:

- The *true concurrency* approach. Here concurrent processes were represented by means of event-based nonsequential models, among which the most established ones were Petri nets [100, 108] and Event Structures [110, 94][5]. These models had pleasant characteristics: they were expressive enough to represent the new phenomena arising in concurrent computations, and they were sufficiently concrete to serve both as system models and as execution models. In some sense, they seemed to be able to play the roles of "denotational" and "operational" models at the same time. Moreover, a formal result established that Event Structures were concrete representations of particular

[5] Other proposed models were Mazurkiewicz traces [83], Grabowski's partial words [70] and Gischer and Pratt's pomsets [55, 103, 104]. However, these appeared to be more suited as execution models than as system models.

domains: as such, they appeared as a natural candidate for denoting concurrent processes. However, both Petri nets and Event Structures lacked two important properties of denotational semantics: abstraction, and, up to the mid-eighties, also compositionality. Indeed, the first studies on Petri nets had concentrated on analysis techniques, rather than on techniques for the synthesis and extension of nets. In connection with the advent of process calculi in the early eighties, combinators for Petri nets and Event Structures had started to be investigated, mainly by Winskel [111, 113] but also by other researchers [32, 33]. Still, these models appeared too intensional to offer a proper denotational semantics for concurrency.

– The *process algebraic* approach. This was a relatively new but very active line of research, initiated by Milner in 1980 with his proposal of a Calculus of Communicating Systems (CCS) [85]. Here concurrent processes were defined as terms of an algebra and interpreted as labelled transition systems, using Plotkin's structural operational semantics (SOS) [102]. The abstract semantics of processes was to be recovered in a second step, by quotienting processes with respect to an observational equivalence, and notably with respect to bisimulation equivalence [99, 86]. This equivalence could furthermore be characterised by a set of axioms. The simplicity and elegance of CCS, which made it formally close to the lambda-calculus, together with the powerful mathematical tools that accompanied it (SOS semantics, bisimulation proof method, algebraic laws), immediately turned CCS, and the family of process calculi which rapidly grew around it, into an appealing field of experimentation for a large community of researchers. Indeed, the introduction of CCS and SOS semantics resulted in a renewed interest in operational semantics for concurrency, after the difficult quest for a denotational semantics in the late 70's (conducted by Milner himself and by other researchers). However, the emphasis of the process algebraic approach was on compositionality rather than on abstraction. Abstraction was only recovered a posteriori through observational equivalences, and thus the resulting semantic model amounted to a (syntactic) term model. In spite of this, CCS had many of the qualities of a denotational model. In particular, the axiomatisations of bisimulation and other behavioural equivalences (e.g. testing equivalence [41]) made it possible to reason about processes and reduce them to normal forms. Another important property of process calculi was their ability to describe both complete systems and their abstract specifications, so that the satisfaction relation of the latter by the former could be simply implemented via observational equivalence. Labelled transition systems were also the prime models for a simple modal logic called Hennessy-Milner logic [74], which precisely characterised bisimulation equivalence.

The two approaches just described had somewhat complementary advantages and drawbacks:

– Process calculi enjoyed the properties of compositionality and substitutivity, by virtue of their algebraic syntax and of the existence of behavioural congruences. On the other hand, they used an *interleaving semantics*, simulating concurrency by a nondeterministic choice of sequential interleavings. The characteristic semantic equation of process calculi, as formally expressed by Milner's expansion law, could informally be rendered as: concurrency = sequentiality + nondeterminism. Hence the notions of causality and distribution, which were present in the syntax, were lost in the semantics.
– Petri nets and Event Structures allowed for a clear distinction between concurrency, causality and conflict, which were all primitive relations on events. In contrast with the interleaving equation, they embodied the true concurrency inequation: concurrency $\neq$ causality + conflict. The main drawback of these models was their lack of abstract semantics.

In the light of this situation, two natural questions came to mind: 1) How to formally relate models? and 2) How to bridge the gap between the two approaches, so as to get the best of each world?

Initial answers to these questions had already started to be provided, both by CEDISYS members (to be) and by other researchers. For instance, a comparison of different concurrency models, related by adjunctions within a common categorical framework, had been carried out by Winskel in [114]. Attempts at narrowing the gap between process calculi and models had also started to be made, either by interpreting process combinators in the models, or by increasing the "observational power" of the operational semantics of calculi so as to capture (some amount of) concurrency and causality.

Among the early interpretations of process calculi (CCS, CSP [75], TCSP [21], CCSP [98]) into nonsequential models, we could cite:

– CCS: interpretations into Petri nets by De Cindio et al. [33], Goltz and Mycroft [63], Winskel [113], Nielsen [92], and Vaandrager and van Glabbeek [62]; interpretations into Event Structures by Winskel [111] and (for a subset of CCS) by Boudol and Castellani [13];

– CSP and TCSP: interpretations into Petri nets by De Cindio et al. [32], by Goltz and Reisig [64] and by Goltz and Loogen [82];

– CCSP (a mixture of CCS and TCSP): interpretation into Petri nets by Olderog [98].

A few proposals for enriching the operational semantics of CCS (sometimes in relation to the interpretations above) had also been put forward. The general idea here was to retain more of the syntax of processes in their semantics, by adding "structure" to the labelled transition system of CCS. This structure could be introduced either via structured actions (generalising atomic actions to composite actions) or via structured states (somehow departing from an extensional view of transition systems), or combining both. We could cite here:

– Structured labels: "pomsets" in Boudol and Castellani's semantics for a subset of CCS [13], and "concurrent histories" in the semantics by Degano, De Nicola and Montanari [42];

– Structured states: pairs of local and concurrent residuals in the distributed bisimulation semantics of Castellani and Hennessy [24, 27], "grapes" in the partial order semantics of [42].

To conclude this introductory section, we could say that in the course of the eighties, a shift had been made from denotational towards operational semantics: thanks to Plotkin's structural approach and to the rapid development of process calculi, operational semantics had acquired new dignity and had gradually become the touchstone for models of concurrency.

One of the central questions of the CEDISYS project was then: how to enrich the operational semantics of CCS so as to take into account concurrency and distribution, and how to relate the new operational semantics to interpretations of CCS into Petri nets or Event Structures?

3 Some achievements of the project

The project's work on noninterleaving semantics for concurrency was structured along five axes: 1) comparison of existing formalisms, 2) models, 3) languages, 4) logics and proof systems, and 5) implementation. It is not our intention here to give an exhaustive account of the work accomplished. Neither shall we try to give a well-balanced description of the project's results. Rather, we aim at giving an idea of the project's collaborative work by picking some representative results which benefitted from the interaction among the partners within axes (1) – (3). We shall try to reconstruct the history of these results within the "4-player ping-pong game" of CEDISYS, and to analyse their influence on subsequent research. As will become evident, Ugo's role in the project was a very prominent and pervasive one: not only was he the inspired instigator and supportive coach of the game (which terminated with four winners and no loser), but he was also personally involved in most of the themes of the project. If the term "pervasive computing" had not yet appeared at that time, the term "pervasive researching" could well have been coined on purpose for Ugo!

As it should be apparent from the discussion in the previous section, there was already a strong convergence of ideas and interests within the consortium, even before the project started. We wish now to show how this convergence turned into a fruitful "cross-fertilization" among the sites in the course of the project: how some existing ideas came to their full development, and how new ideas emerged from the project's "chemical soup" (in the sense of the CHAM model [10] described below...). In doing so, we shall focus on two main strands: 1) noninterleaving operational semantics for CCS, and 2) nonsequential models and abstract semantics.

3.1 Noninterleaving operational semantics for CCS

Process calculi include a parallel composition operator and a form of sequential composition. These operators specify respectively a concurrency and a causality relation between the actions of their components. However, these relations are "forgotten" when passing from the syntax to the standard labelled transition semantics of the calculus. To define a noninterleaving operational semantics for CCS the idea was then, quite naturally, to add structure to its labelled transition system so as to retain in the semantics some of the information about concurrency and causality which was present in the syntax. As mentioned in Section 2, some proposals for enriching the operational semantics of CCS along these lines, by adding structure to the actions or to the states of transitions, had already been presented by some of the project's partners. However, these proposals did not completely fulfill the objective, either because they were defined only on subsets of CCS, or because they did not fully agree with existing interpretations of CCS into Event Structures and Petri nets. Indeed, the initial criterion for these noninterleaving operational semantics for CCS was their agreement with interpretations of the calculus into event based semantic domains.

This line of work, aiming at obtaining noninterleaving operational semantics for CCS by enriching its labelled transition system, was intensively pursued in the project. Notable advances were made, both on structured actions and on structured states:

1. **Structured actions**. In the standard labelled transition system of CCS, transitions are labelled by atomic actions. Two ways of generalising such actions were investigated:

   • *Composite actions*. The idea here was to relax the atomicity constraint to allow transitions to be labelled by whole nonsequential computations. In the early work by Boudol and Castellani [13, 14], transitions labelled by *pomsets* (partially ordered multisets) had been defined for a simple CCS-like language with parallel composition but no communication. In that case, pomset transitions could be directly constructed by means of new transition rules, whose effect was to transfer the constructs of sequential and parallel composition from the processes to the actions labelling their transitions. However, it was not clear how to extend this direct pomset semantics to the full language CCS, since the parallel composition of two pomsets does not in general yield a pomset when communication is allowed. This led the authors to propose, in [16, 15], an indirect construction of a pomset transition starting from a class of permutation-equivalent transition sequences. This construction "a posteriori" of a partial order from an equivalence class of sequential computations was similar to that used for defining Mazurkiewicz traces [84]. The novelty of [16, 15] was the use of enriched transitions for CCS called *proved transitions* (because they were labelled with a representation of their proof in the inference system of CCS), from which the concurrency relation between transitions could be derived by syntactic means. A different construction of partial order computations for CCS, starting from enriched

sequential computations called *concurrent histories*, obtained by concatenating *atomic histories* (corresponding to single steps) and recording their causality relation, was proposed by Degano, De Nicola and Montanari in [42, 47, 46]. In that case, the construction exploited structured states rather than structured actions, and made use of the causality rather than the concurrency relation. This semantics will be discussed in more detail later, under the heading "structured states". Note that, while in the direct pomset semantics of [13, 14] atomic actions were replaced with composite actions in the transition system itself, in both the proved transition semantics [16, 15] and the concurrent history semantics [45, 44] actions remained atomic in the basic transition system but were decorated with additional information (they were instances of the "decorated atomic transitions" discussed below). Partial order transitions were only retrieved in a second step.
Clearly related to the generalisation of atomic actions to composite actions was the issue of *action refinement*, which was thoroughly studied within the project. However, the motivation behind action refinement was not that of modelling nonsequential computations, bur rather of managing different levels of abstraction in the description of computations (sequential or not). The project's work on action refinement will be described in Section 3.2.

• *Decorated (atomic) actions.* In this case the idea was to remain with atomic actions but to annotate them with syntactic information so as to identify the component responsible for them. Several kinds of decorated actions were proposed:

– *Actions with localities* [19, 20, 4, 89, 91]. Localities were introduced to distinguish different parallel components: they could either be assigned statically to processes (*static localities* [4, 91, 25]) and then transmitted to their actions, or be dynamically created and associated with actions, and then recorded in the residual process (*dynamic localities* [19, 20]). The notion of dynamic locality proposed by Boudol, Castellani, Hennessy and Kiehn was already implicitly present in the distributed bisimulation semantics of [27], embodied in the idea of splitting the residual of a transition among multiple observers; indeed, the idea of associating localities with actions and processes emerged after some insightful (but unsuccessful) attempts by Kiehn [78] at extending to full CCS the distributed bisimulation semantics of [27]. As it were, in the presence of global scope operators like restriction, the idea of splitting the residual of a transition among multiple observers could only be implemented by inserting the observers themselves in the residual, which exactly amounted to the introduction of dynamic localities. Similarly, the notion of static locality was already implicitly contained in both that of "atomic history" by Degano and Montanari [42, 49], and in that of "proof" by Boudol and Castellani [16, 15]. In this sense, Montanari and Yankelevic's paper on parametric localities [89] may be seen as an accomplishment of both the concept of atomic history and of the notion of locality (and particularly of static locality [4]) which had been meanwhile developed in the project.

- *Actions with causes.* In the model of *causal trees* by Darondeau and Degano [36, 37], the actions labelling the branches were annotated with their set of (global) causes in the computation given by the path from the root to the action. In Kiehn's subsequent papers [79, 80], this causal semantics was compared with the dynamic locality semantics within a common framework (the "local-global cause transition system"), which also allowed a stronger semantics to be defined, based on the conjunction of local and global causes.
- *Actions with proofs.* In the proved transition system of Boudol and Castellani [16, 15], labels go all the way from simple atomic actions to complete "proofs" of transitions. As remarked by the authors in [18], the notion of proof could be weakened in various ways to obtain more specific semantics (for instance, the static locality semantics). Again, the notion of "atomic history" [42] could be seen as a precursor to that of "proof" (although it did not record choices, and hence did not support the definition of a concurrency relation between transitions).
- *Actions with past.* Both actions with dynamic localities and actions with causes are instances of actions with portions of their "computational past" or "global history". In contrast, actions with a "static past" or "local history" were considered in the *event transition system* proposed for CCS by Boudol and Castellani in [18], as an intermediate among three different interpretations of the calculus (into proved transition systems, flow event structures and flow nets).

2. **Structured states**. New forms of structured states were explored, often in connection with the decorated actions described above:
   - *Grapes.* In their early model of *concurrent histories* [49], Degano and Montanari had defined concrete computations with structured initial and final states. These structured states, called *grapes* [42], were essentially the sets of sequential components of a process, each of them being annotated with its "position" within the process. This annotation allowed the computations to be concatenated while recording the causality relation among their actions. The constituent blocks of such computations, called *atomic histories*, could be directly derived by operational rules. From a concurrent history, a partial ordering of actions could be abstracted away. This model was used by Degano, De Nicola and Montanari to define a partial ordering semantics for CCS in [42, 47].
   - *Processes with past.* Whenever actions with past are used, it is also necessary to record the past in the states, so that the computational history can be incremented at each step and propagated with subsequent actions. Thus, corresponding to actions with dynamic localities and actions with causes, *processes with localities* and *processes with causes* were introduced. A more general notion of process with past or "marked process" was considered in the event transition system of [18]. A marked process contains all the information of the original process, but additionally specifies which part of the process has been executed. It corresponds to

an unfolded Petri net with a marking, or to an event structure with a configuration.

Although not expressly tailored for true concurrency, but reflecting in some sense the notions of distribution and mobility associated with concurrency, a new model for describing the operational semantics of parallel processes, called the *Chemical Abstract Machine* (CHAM), was proposed by Berry and Boudol in [10]. In this model, the state of a system of concurrent agents is viewed as a "chemical solution", in which floating "molecules" can interact with each other according to "reaction rules". The CHAM model introduced a new, simpler way of expressing a reduction semantics in a concurrent scenario, which was to become quite popular in later research on process calculi (notably on the π-calculus [86], the join-calculus [54] and the ambient calculus [22]).

3.2 Abstract noninterleaving semantics

Most of the noninterleaving semantics for CCS described in the previous section were compared with interpretations of CCS into nonsequential models such as Event Structures and Petri nets, or used as a basis for defining behavioural equivalences on processes, in order to obtain more abstract semantics.

Nonsequential models As mentioned earlier, the initial criterion for assessing the new noninterleaving operational semantics for CCS was their agreement with "denotational semantics", given by interpretations of CCS into semantic domains like Event Structures and Petri nets. Thus, for instance, in [45, 44] Degano, De Nicola and Montanari showed the consistency of their operational semantics of [43] with interpretations of CCS into *prime event structures* [94] and *condition-event systems* (a class of Petri nets). In [16], Boudol and Castellani showed that their proved transition semantics agreed with an interpretation of CCS (and SCCS) into *flow event structures*, a class of event structures lying between prime and stable event structures [114], thus generalising their previous result of [13, 14]. In a joint paper between Aarhus and Sophia [28], it was shown that parallel composition of flow event structures can be characterized as a categorical product, and that flow event structures obtained as interpretations of CCS terms satisfy a particular structural property (which generalises that given in [13] for a subset of CCS). In [18], the results of [14, 16] were extended to *flow nets* [12], a class of "stable" Petri nets, and to *asynchronous transition systems*, a model introduced by Bednarczyk in [9]. Indeed, the need for interpreting process algebras gave a new impulse to the study of semantic domains. Besides the above-mentioned flow event structures and flow nets, which were designed to fit typical process combinators, we could also cite here the models of *bundle event structures* [81] (developed outside the project but shown to correspond to safe flow nets in [17]), *Δ-free event structures* [38], and later on, *transition systems with independence* [117].

On a more abstract level, a line of research that was actively pursued within the project was the comparison of different semantic models within an algebraic or categorical framework. A web of strong formal connections, expressed as adjunctions among various transition-based models presented as categories, had already been established by Winskel in [112, 115]. A notable contribution was made by Nielsen, Rozenberg and Thiagarajan with the introduction of the notion of *region*, which allowed the above connections to be extended to a class of structured transition systems [95–97]. The idea of region was to be further used by Winskel and Nielsen in [117] to give an adjunction between Petri nets and asynchronous transition systems [9].

The unification of models of concurrency into a common algebraic framework had also been a central concern of Ferrari and Montanari's work [51, 53], where the permutation semantics of [14] was lifted to a more abstract setting, which could then be specialised in various ways. For instance, by introducing axioms to equate permutation-equivalent computations, one could obtain partial ordering computations. Also, by considering categories having CCS models as objects, one could use morphisms to represent a specification-implementation relation between two CCS models. In the paper [52], the same authors proposed a generalisation of Milner's expansion theorem to a language called CCCS, a variant of CCS with an "observation prefixing" operator. Observations were elements of an algebra (in fact, they were "proofs" in the sense of [14]). By adding axioms to this algebra, one could instantiate the *extended expansion theorem* to characterise different bisimulation equivalences, such as standard bisimulation and pomset bisimulation. As it were, the extended expansion theorem was a generalisation of that proposed in [13] for pomset bisimulation (where a notation for "pomset prefixing" had similarly been introduced for axiomatisation purposes). In the same spirit, a parametric approach for axiomatising bisimulation equivalences was later proposed in [48]: in this case, the underlying structures, called "observation structures", were node-labelled graphs where the labels represented computations. This framework allowed for the characterisation of a larger number of equivalences. The last model we wish to mention is Gorrieri and Montanari's SCONE calculus [67], a calculus of Petri nets generated by a set of combinators, which was used in particular to implement CCS.

As might be apparent from the above discussion, there were two main "angles of attack" in the project. In general, Aarhus and Pisa partners had a strong concern in generality and parametricity, and tended to privilege a "uniform" or "parametric" approach to the study of nonsequential models. By contrast, Sophia and Sussex partners preferred to focus on particular calculi or models, adopting a more "language-specific" or "model-specific" approach.

Noninterleaving equivalence notions Based on the various concrete operational semantics described in Section 3.1, several noninterleaving behavioural equivalences and preorders were proposed for CCS. For some of them, axiomatisations or logical characterisations were also provided. Let us briefly recall the main proposals:

- *NMS-equivalence.* In [43], Degano, De Nicola and Montanari had proposed a partial ordering semantics based on a tree-model called Nondeterministic Measurement Systems (NMS). By applying "observation functions" to NMS's, different equivalence notions could be obtained, among which a partial ordering equivalence called NMS-equivalence.
- *Pomset bisimulation.* The pomset transition system of [13] and its extension in [16] gave rise to the notion of pomset bisimulation. Pomset bisimulation equivalence was unrelated to NMS-equivalence. An axiomatisation of pomset bisimulation for a subset of CCS was given in [13].
- *Mixed-ordering equivalence.* Based on the concurrent history semantics [49], Degano, De Nicola and Montanari proposed in [45, 44] an equivalence called "mixed-ordering equivalence", which combined the observation of the temporal ordering with that of the causal ordering of actions. This equivalence was stronger than pomset bisimulation, and was shown by Aceto [2] to coincide with an equivalence proposed by Goltz and Van Glabbeek to deal with action refinement, called *history-preserving bisimulation* [57].
- *Distributed bisimulation.* Some advances were made on distributed bisimulation equivalence. In his PhD thesis [31], Christensen presented a logical characterisation and a decidability result for distributed bisimulation (in its initial formulation with local and concurrent residuals [24]). In [78], Kiehn showed the limits of the original definition of distributed bisimulation, which appeared to be incompatible with the CCS restriction operator. Indeed, in the light of Kiehn's observations [78] and Aceto's results [3], the "right" extension of distributed bisimulation to full CCS turned out to be dynamic location equivalence (see below, and previous discussion at page 7).
- *Location equivalence and preorder.* The static [4] and dynamic [19, 20] locality semantics supported two different definitions of "location equivalence" and "location preorder". For the dynamic versions, an axiomatisation was also provided in [20]. In [4], Aceto established the coincidence of the static and dynamic notions for a subset of CCS with only top-level parallelism (this work was later to be extended to full CCS by Castellani [25] and by Mukund and Nielsen [91]). Since static and dynamic localities do not have the same meaning - static localities represent sites, while dynamic localities represent sets of local causes - this result was not obvious. It amounted to proving that observing distribution and observing local causality were essentially the same thing. Dynamic location equivalence was shown to coincide with distributed bisimulation on the subset of CCS without restriction, where the latter had been originally defined. On the smaller subset of CCS with parallel composition but no communication, it was finer than pomset bisimulation (because it recorded how computations were locally extended). As soon as communication was introduced, it became incomparable with pomset bisimulation.
- *Local mixed-ordering equivalence.* A transition system for CCS labelled with static localities, called "spatial transition system", was studied by Montanari and Yankelevich in [118, 90]. In this case, transitions with localities were not directly used to define an equivalence, but rather to build a sec-

ond transition system, labelled by "local mixed partial orders" (where the ordering was a mixture of temporal ordering and local causality). This new transition system was then used to define a behavioural equivalence called local mixed-ordering equivalence. This equivalence, a variant of the mixed-ordering equivalence discussed above, was shown in [118] to coincide with dynamic location equivalence.
- *Causal bisimulation.* The bisimulation equivalence associated with Darondeau and Degano's causal trees, called causal bisimulation [36], was also investigated in relation to action refinement in [38]. It was shown to coincide with history-preserving bisimulation, and hence also with mixed-ordering equivalence. A complete axiomatisation for causal bisimulation was given in [36].
- *Local-global cause equivalence.* A "local-global cause transition system" for CCS, where actions were decorated with both their local and global causes, was proposed by Kiehn in [79] and [80] (see also [77]). By disregarding the set of global causes, respectively local causes, of actions, one could then obtain two different equivalences called "local cause equivalence" and "global cause equivalence", which were shown to coincide respectively with dynamic location equivalence and causal bisimulation. This semantics also supported a stronger equivalence, called local-global cause equivalence, arising from the joint observation of the two sets of causes.

An interesting and somewhat unexpected outcome of these studies on noninterleaving equivalences was that distributed and causal semantics, exemplified respectively by location equivalence and causal bisimulation, were in general incomparable: they only coincided on the finite fragment of CCS without communication (essentially because in that case all causality is local). Another observation was that, while both location equivalence and causal bisimulation are "history-preserving" equivalences (recording respectively the local and the global history), other equivalences like pomset bisimulation were instead "forgetful". A detailed comparison of noninterleaving equivalences was given by Aceto in [2, 3]. A more recent comparison, including other equivalences, may be found in [26].

Action refinement Another question that was thoroughly studied within the project was that of action refinement. The motivation here was to allow for a specification of systems at different levels of abstraction. This was to be used as a conceptual underpinning for the so called "top-down design" of distributed systems, whereby a system is developed by successive refinements of atomic actions into more complex behaviours. Usually an "implementation relation" between two successive descriptions was introduced, in order to guarantee the correctness of the refinement process.

Approaches to action refinement may be classified according to various criteria:

- *atomic* versus *non-atomic* refinement: in the first case the refining behaviour is required to be executed atomically, while in the latter it may be freely interleaved with other behaviours;

– *horizontal* versus *vertical* refinement: in the first case the abstract and concrete descriptions are given in the same formalism, while in the latter case they are given in different formalisms, and an encoding from the first to the second formalism has to be provided;
– *syntactic* versus *semantic* refinement: in the first case action refinement is introduced in a language by means of a specific operator, and implemented as syntactic substitution; in the second case it is defined on a semantic model.

In some early work on atomicity by Boudol and Castellani [14, 11], as well as in the paper [66] by Gorrieri, Marchetti and Montanari, methods for treating computations as *atomic* (that is, both *recoverable* and *interference-free*, in the terminology of [11]) had been considered. In both cases, an auxiliary labelled transition system was introduced for describing atomic computations. However, these proposals were not explicitly concerned with the issue of action refinement, but rather with that of atomicity of behaviours. Subsequently, all the project's work on action refinement was essentially concerned with non-atomic refinement, both syntactic and semantic, and mostly of the horizontal type. Indeed, non-atomic refinement was the most popular approach at that time.

Although the motivation behind action refinement was quite different from that of noninterleaving semantics, a connection between the two questions was deemed to exist for some time, and resulted in a concurrent development of their theories. Let us briefly recall the research context at the time.

When dealing with syntactic action refinement, a new refinement operator is introduced in the language. Then a natural question to address is whether this operator preserves behavioural equivalences. In a short note [29] published in 1987, Castellano, De Michelis and Pomello showed that standard bisimulation equivalence was not preserved by action refinement, and conjectured that the recourse to a noninterleaving semantics was necessary to cope with refinement. Robustness with respect to action refinement became a new criterion to assess equivalence notions, and spurred the search for new noninterleaving equivalences.

Hennessy's unpublished manuscript "On the relationship between time and interleaving" [71], later to become "Axiomatising finite processes" [72], was probably the first attempt to consider actions with duration without explicitly introducing time, by splitting atomic actions into a beginning subaction and an ending subaction. Strong bisimulation equivalence with respect to these subactions was called *timed equivalence* in [6], where it was shown to be preserved by action refinement in the simple setting of CCS without communication and restriction. The *owl example* from [56] shows that this is no longer the case when the expressiveness of the language is increased to allow communication and some form of hiding. A strengthening of *timed equivalence* called *refine equivalence* is obtained by requiring, in the bisimulation game, that the matching of begin actions should determine the matching of the corresponding end actions; the identity of the original action which gives rise to the begin and end actions is remembered in the bisimulation game. In [7], the weak version of this equivalence, called *weak refine equivalence*, was shown to be preserved by refinement in a more expressive language with communication. Although somewhat complicated to define, the

latter equivalence seems more natural than timed equivalence. It is also technically easier to handle; for example the refinement theorem in [6], in the simple language, was actually proved for the strong version of refine equivalence, and the result was only obtained indirectly for timed equivalence by showing that the two equivalences coincided on the simple language.

Similar ideas were pursued outside the project by van Glabbeek and Vaandrager [62, 56], in the setting of non-interleaving models such as Event Structures and Petri Nets. Their equivalence, called *ST-bisimulation equivalence*, was very close in spirit to refine equivalence, although the decision of which end actions are to be matched was determined by the previous history of associations between actions. In [62], van Glabbeek and Vaandrager showed that ST-bisimulation was robust for action refinement in the setting of Event Structures. Since ST-bisimulation is not a partial order equivalence (it was conceived for real-time and disregarded some causality) this result invalidated the conjecture of [29].

Meanwhile Goltz and van Glabbeek [57] had shown that, except for pomset trace equivalence, none of the partial order equivalences existing at the time was preserved by refinement. In particular pomset bisimulation and NMS equivalence were not robust with respect to refinement. In the same paper, it was shown that a stronger partial order equivalence called *history preserving bisimulation* (already used by Rabinovitch and Trakhtenbrot under the name "behaviour structure equivalence" [107], but reformulated on Event Structures by Goltz and van Glabbeek), was preserved by refinement on prime event structures. This result was to be strengthened in subsequent papers, by considering more general forms of refinement and more general classes of Event Structures [58–60].

The above-mentioned work by Aceto and Hennessy mostly concentrated on syntactic refinement. In Aceto's thesis [1], on the other hand, both syntactic and semantic refinement were considered. In a similar vein, Nielsen, Engberg and Larsen presented in [93] four different fully abstract models, based on sets of pomsets, for a simple CCS-like process calculus (essentially the same as in [13]) equipped with a refinement construct. These results were generalised by Aceto and Engberg to a model based on *pomsets failures* (pomsets with refusal sets) in [5]. Semantic refinement was studied by Gorrieri and Montanari in [67, 68], as well as in Gorrieri's thesis [65]; here, vertical refinement was privileged and the use of implementation relations, parameterised by refinement functions, was investigated. More specifically, in [67] Gorrieri and Montanari proposed an implementation of the subset of CCS without restriction and relabelling into the net calculus SCONE. This result was generalised in [65] to full CCS and to SCONE$^+$, an extension of SCONE. In [68] the same authors put forward a general methodology for "atomic linear refinement", where each refined process could be proved to be a correct atomic implementation of its source process.

In [38], Darondeau and Degano explored the issue of syntactic and semantic action refinement in their model of causal trees [36]. To this purpose, they proposed an adaptation of prime event structures called "free event structures", which was closed with respect to event refinement. Causal trees were already known to correspond to event structures up to history-preserving bisimulation.

Refinement operators were then introduced for causal trees and the agreement of syntactic and semantic refinement was established.

4 Conclusion

In this section, we attempt a short, and certainly incomplete, analysis of the impact of the CEDISYS project on contemporary and subsequent research. As can be gathered from the discussion in the previous sections, the project was the theatre of intense "cross-fertilization". Many examples of mutual influences among the partners have already been cited. Let us point out some others, and mention some later influences on other researchers.

The work on *localities* may be viewed as carrying the project's "trademark". As explained earlier, the notion of dynamic locality stemmed from previous work on distributed bisimulation, which in turn had been inspired by *abstraction homomorphisms* [23, 88] (or rather their extension to true concurrency), and by some earlier work by Degano, De Nicola and Montanari on partial order semantics for CCS [42]. The later work by Corradini and De Nicola on localities [34] was also very much in the spirit of CEDISYS. Localities are now an intrinsic component of the wide range of calculi used to study "network aware" computing, with varying characteristics reflecting the different facets of network computing; typical examples include the ambient calculus [22], the distributed π-calculus [73], the join-calculus [54], and the language KLAIM (locality-based LINDA) [39, 40].

The essential characteristics of the *chemical abstract machine* [10] have been widely adopted, and now form part of the natural framework for designing and studying computational calculi. In particular they provide a very convenient and useful methodology for defining the reduction semantics of such calculi.

The notion of *proved transition system* has also been very influential. Apart from its use within the project, let us cite the work by Badouel and Darondeau, who established in [8] a correspondence between (a variant of) the proved transition semantics and an interpretation of CCS into Stark's trace automata [109]. In [50], Degano and Priami studied *proved trees*, a subset of proved transition systems which was well-suited for the comparison with causal trees. Subsequently, proved transition systems were taken up by Priami [105], under the name *enhanced transition systems*, and used for many different purposes; a particularly interesting application area is that of *Computational Systems Biology*, [106]. The study of reversible computing [35, 101] also benefitted from this idea; for example, the model used in [101] is very close to that of *event transition systems* [18].

The project's work on *refinement* was both inspired and used by Goltz and van Glabbeek [60, 59] as well as by Rensink [69]. These authors also took up the model of *flow event structures*, which turned out to be well-suited for certain forms of refinement. In particular, in [61], Goltz and van Glabbeek proposed a subclass of flow event structures which was both closed under action refinement and well-behaved with respect to parallel composition, thus generalising the work by Castellani and Zhang [28].

The notion of *open map bisimulation* [76, 30], introduced a few years after the end of the project, was partly inspired by that of abstraction homomorphism. The general issue of abstraction, namely how to make models such as event structures and Petri nets and their morphisms abstract enough to support a fully fledged "domain theory for concurrency", continues to be pursued, one recent line being through the introduction of a formal treatment of symmetry on such models [116].

We could not close this recollection of the CEDISYS project without a special praise for "Ugo's management style", which led the project to meet its commitments with a Swiss clock punctuality, while being experienced by all its members as a most stimulating and enjoyable collaboration.

Acknowledgements We would like to thank Luca Aceto, Rocco De Nicola and the anonymous reviewer for helpful comments.

References

1. L. Aceto. *Action-refinement in process algebras*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1992.
2. L. Aceto. History preserving, causal and mixed-ordering equivalence over stable event structures (Note). *Fundamenta Informaticae*, 17(4):319–331, 1992.
3. L. Aceto. Relating distributed, temporal and causal observations of simple processes. *Fundamenta Informaticae*, 17(4):369–397, 1992.
4. L. Aceto. A static view of localities. *Formal Aspects of Computing*, 6:201–222, 1994. Previously appeared as INRIA Research Report n. 1483, 1991.
5. L. Aceto and U. Engberg. Failure semantics for a simple process language with refinement. In *Proceedings FST-TCS 91*, number 560 in LNCS, pages 89–108, 1991.
6. L. Aceto and M. Hennessy. Towards action-refinement in process algebras. *Information and Computation*, 103(2):204–269, 1993. Extended abstract in *Proceedings LICS 89*, IEEE Computer Society Press, 1989.
7. L. Aceto and M. Hennessy. Adding action refinement to a finite process algebra. *Information and Computation*, 115(2):179–247, 1994. Extended abstract in *Proceedings ICALP 91*, number 510 in LNCS, 1991.
8. E. Badouel and P. Darondeau. Structural operational specifications and trace automata. In *Proceedings CONCUR 92*, number 630 in LNCS, 1992.
9. M. Bednarczyk. *Categories of Asynchronous Systems*. PhD thesis, University of Sussex, 1988.
10. G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992. Extended abstract in *Proceedings POPL 90*, pages 81-94, 1990.
11. G. Boudol. Atomic actions (Note). *Bulletin of the European Association for Theoretical Computer Science* 38, pp. 136-144, 1989.
12. G. Boudol. Flow event structures and flow nets. In *Proceedings LITP Spring School, La Roche-Posay*, number 469 in LNCS, pages 62–95, 1990.
13. G. Boudol and I. Castellani. On the semantics of concurrency: partial orders and transition systems. In *Proceedings TAPSOFT 87*, number 249 in LNCS, pages 123–137, 1987.

14. G. Boudol and I. Castellani. Concurrency and atomicity. *Theoretical Computer Science*, 59:25–84, 1988.
15. G. Boudol and I. Castellani. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informaticae*, 11(4):433–452, 1988.
16. G. Boudol and I. Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS. In *Proceedings REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, number 354 in LNCS, 1988.
17. G. Boudol and I. Castellani. Flow models of distributed computations: event structures and nets. Research Report 1482, INRIA, 1991. Extended version of paper by G. Boudol in *Proceedings LITP Spring School*, La Roche-Posay, number 469 in LNCS, 1990.
18. G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. *Information and Computation*, 114(2):247–314, 1994. Extended abstract in *Proceedings LITP Spring School*, La Roche-Posay, number 469 in LNCS, 1990.
19. G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *Theoretical Computer Science*, 114:31–61, 1993. Extended abstract in *Proceedings MFCS 91*, number 520 in LNCS, 1991.
20. G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. *Formal Aspects of Computing*, 6:165–200, 1994. Extended abstract in *Proceedings CONCUR 92*, number 630 in LNCS, 1992.
21. S. Brookes, C.A.R. Hoare, and A. Roscoe. A theory of communicating sequential processes. *Journal of ACM*, 31(3):560–599, 1984.
22. L. Cardelli and A. D. Gordon. Mobile ambients. In *Proceedings FoSSaCS 98*, number 1378 in LNCS, 1998.
23. I. Castellani. Bisimulations and abstraction homomorphisms. *Journal of Computer and System Sciences*, 34:210–235, 1987.
24. I. Castellani. *Bisimulations for Concurrency.* PhD thesis, University of Edinburgh, 1988.
25. I. Castellani. Observing distribution in processes: static and dynamic localities. *Int. Journal of Foundations of Computer Science*, 4(6):353–393, 1995. Extended abstract in *Proceedings MFCS 93*, number 711 in LNCS, 1993.
26. I. Castellani. Process algebras with localities. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 945–1045. North-Holland, Amsterdam, 2001.
27. I. Castellani and M. Hennessy. Distributed bisimulations. *JACM*, 36(4):887–911, 1989.
28. I. Castellani and G. Q. Zhang. Parallel product of event structures. *Theoretical Computer Science*, 179(1-2):203–215, 1997. Previously appeared as DAIMI Research Report PB-301, 1989, and as INRIA Research Report 1078, 1989.
29. L. Castellano, G. De Michelis, and L. Pomello. Concurrency vs interleaving: an instructive example. *Bulletin of the EATCS*, 31:12–15, 1987.
30. G. Cattani and G. Winskel. Profunctors, open maps and bisimulation. *MSCS*, 15(3):553–614, 2005.
31. S. Christensen. Distributed bisimilarity is decidable for a class of infinite state-space systems. In *Proceedings CONCUR 92*, number 630 in LNCS, 1992.
32. F. De Cindio, G. De Michelis, L. Pomello, and C. Simone. A Petri net model for CSP. In *Proceedings CIL 81*, Barcelona, 1981.

33. F. De Cindio, G. De Michelis, L. Pomello, and C. Simone. Milner's Communicating Systems and Petri Nets. In *European Workshop on Applications and Theory of Petri Nets*, pages 40–59, 1982.
34. F. Corradini and R. De Nicola. Locality based semantics for process algebras. *Acta Informatica*, 34:291–324, 1997.
35. V. Danos and J. Krivine. Reversible communicating systems. In P. Gardner and N. Yoshida, editors, *CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 292–307. Springer, 2004.
36. Ph. Darondeau and P. Degano. Causal trees. In *Proceedings ICALP 89*, number 372 in LNCS, 1989.
37. Ph. Darondeau and P. Degano. Causal trees: interleaving + causality. In *Proceedings LITP Spring School,* La Roche-Posay, number 469 in LNCS, 1990.
38. Ph. Darondeau and P. Degano. Refinement of actions in event structures and causal trees. *Theoretical Computer Science*, 118(1):21–48, 1993. Extended abstract in *Proceedings MFCS 90*, number 452 in LNCS, 1990.
39. R. De Nicola, G. Ferrari, and R. Pugliese. Locality based LINDA: programming with explicit localities. In *Proceedings TAPSOFT-FASE 97*, number 1214 in LNCS, 1997.
40. R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM: a kernel language for agents interaction and mobility. *IEEE Trans. on Software Engineering*, 24(5):315–330, 1998.
41. R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 43:83–133, 1984.
42. P. Degano, R. De Nicola, and U. Montanari. Partial ordering derivations for CCS. In *Proceedings FCT 85*, number 199 in LNCS, pages 520–533, 1985.
43. P. Degano, R. De Nicola, and U. Montanari. Observational equivalences for concurrency models. In *Proceedings 3rd IFIP WG 2.2 Working Conference,* Ebberup 1986. North-Holland, 1987.
44. P. Degano, R. De Nicola, and U. Montanari. A distributed operational semantics for CCS based on condition/event systems. *Acta Informatica*, 26(1/2):59–91, 1988.
45. P. Degano, R. De Nicola, and U. Montanari. On the consistency of truly concurrent operational and denotational semantics. In *Proceedings LICS 88*. IEEE Computer Society Press, 1988.
46. P. Degano, R. De Nicola, and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Proceedings REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, number 354 in LNCS, 1989.
47. P. Degano, R. De Nicola, and U. Montanari. A partial ordering semantics for CCS. *Theoretical Computer Science*, 75(3):223–262, 1990.
48. P. Degano, R. De Nicola, and U. Montanari. Universal axioms for bisimulations. *Theoretical Computer Science*, 114:63–91, 1993. Extended abstract in *Proceedings 3rd Workshop on Concurrency and Compositionality*, GMD-Studien Nr. 191, 1991.
49. P. Degano and U. Montanari. Concurrent histories: A basis for observing distributed systems. *Journal of Computer and System Sciences*, 34(2/3):422–461, 1987.
50. P. Degano and C. Priami. Proved trees. In *Proceedings ICALP 92*, number 623 in LNCS, 1992.
51. G. Ferrari. *Unifying Models of Concurrency.* PhD thesis, University of Pisa, 1990.

52. G. Ferrari, R. Gorrieri, and U. Montanari. An extended expansion theorem. In *Proceedings TAPSOFT 91*, number 494 in LNCS, 1991.
53. G. Ferrari and U. Montanari. Towards the unification of models of concurrency. In *Proceedings CAAP 90*, number 431 in LNCS, 1990.
54. C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proceedings POPL 96*, pages 372–385, 1996.
55. J.L. Gischer. *Partial orders and the axiomatic theory of shuffle.* PhD thesis, Stanford University, 1984.
56. R.J. van Glabbeek. The refinement theorem for ST-bisimulation semantics. In *Proceedings IFIP TC2 Working Conference on Programming Concepts and Methods*, pages 27–52. North-Holland, 1990.
57. R.J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *Proceedings MFCS 89*, number 379 in LNCS, 1989.
58. R.J. van Glabbeek and U. Goltz. Equivalences and refinement. In *Proceedings LITP Spring School,* La Roche-Posay, number 469 in LNCS, 1990.
59. R.J. van Glabbeek and U. Goltz. Refinement of actions in causality based models. In *Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness*, volume 430 of LNCS, 1990.
60. R.J. van Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37(4/5):229–327, 2001. Previously appeared as Research Report 6/98, University of Hildesheim, 1998.
61. R.J. van Glabbeek and U. Goltz. Well-behaved flow event structures for parallel composition and action refinement. *Theoretical Computer Science*, 311(1-3):463–478, 2004.
62. R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency. In *Proceedings PARLE 87*, number 259 in LNCS, 1987.
63. U. Goltz and A. Mycroft. On the relationship of CCS and Petri nets. In *Proceedings ICALP 84*, number 172 in LNCS, 1984.
64. U. Goltz and W. Reisig. CSP programs as nets with individual tokens. In *Advances in Petri nets 1984*, number 188 in LNCS, pages 169–196, 1985.
65. R. Gorrieri. *Refinement, Atomicity and Transactions for Process Description Languages.* PhD thesis, University of Pisa, 1991.
66. R. Gorrieri, S. Marchetti, and U. Montanari. A^2CCS: Atomic actions for CCS. *Theoretical Computer Science*, 72:203–223, 1990. Extended abstract in *Proceedings CAAP 88*, number 299 in LNCS, 1988.
67. R. Gorrieri and U. Montanari. SCONE: A simple calculus of nets. In *Proceedings CONCUR 90*, number 458 in LNCS, 1990.
68. R. Gorrieri and U. Montanari. Towards hierarchical description of systems: a proof system for strong prefixing. *Int. Journal of Foundations of Computer Science*, 1(3):277–293, 1990.
69. R. Gorrieri and A. Rensink. Action refinement. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 1047–1147. North-Holland, Amsterdam, 2001.
70. J. Grabowski. On partial languages. *Fundamenta Informaticae*, 4(1):427–498, 1981.
71. M. Hennessy. On the relationship between time and interleaving. Unpublished draft, Sophia-Antipolis, 1980.
72. M. Hennessy. Axiomatising finite concurrent processes. *SIAM Journal of Computing*, 17(5):997–1017, 1988.
73. M. Hennessy. *A Distributed Pi-calculus.* Cambridge University Press, Cambridge, UK, 2007.

74. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *JACM*, 32, 1985.
75. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
76. A. Joyal, M. Nielsen, and G. Winskel. Bisimulation and open maps. In *Proceedings LICS 93*, pages 418–427. IEEE Computer Society Press, 1993.
77. A. Kiehn. *Concurrency in process algebras*. Habilitation Thesis, Technische Universität München, 1999.
78. A. Kiehn. Distributed bisimulations for finite CCS. Report 7/89, University of Sussex, 1989.
79. A. Kiehn. Local and global causes. Technical Report 342/23/91, Technische Universität München, 1991.
80. A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 31:697–718, 1994.
81. R. Langerak. Bundle event structures: a non-interleaving semantics for LOTOS. In *Proceedings IFIP TC6/WG6.1 Fifth International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*, 1992. Previously appeared as University of Twente Research Report, 1991.
82. R. Loogen and U. Goltz. Modelling nondeterministic concurrent processes with event structures. *Fundamenta Informaticae*, XIV(1):39–74, 1991.
83. A. Mazurkiewicz. Concurrent program schemes and their interpretation. Report DAIMI PB-78, Aarhus University, 1977.
84. A. Mazurkiewicz. Trace theory. In *Proceedings Advances in Petri Nets 1986,* number 255 in LNCS, 1987.
85. R. Milner. *A Calculus of Communicating Systems*, volume 92 of LNCS. Springer-Verlag, 1980.
86. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
87. U. Montanari, G. Boudol, I. Castellani, G. Ferrari, M. Hennessy, M. Nielsen, and G. Winskel. ESPRIT Basic Research Action n. 3011 CEDISYS - Final Report, 1992. Downloadable at the URL: http://www-sop.inria.fr/mimosa/personnel/Ilaria.Castellani/CEDISYS-final-report.pdf.
88. U. Montanari and M. Sgamma. Canonical representatives for observational equivalence. In Proceedings Colloquium on the Resolution of Equations in Algebraic Structures, pages 292–319, 1989. Academic Press.
89. U. Montanari and D. Yankelevich. A parametric approach to localities. In *Proceedings ICALP 92*, number 623 in LNCS, 1992.
90. U. Montanari and D. Yankelevich. Location equivalence in a parametric setting. *Theoretical Computer Science*, 149:299–332, 1995.
91. M. Mukund and M. Nielsen. CCS, locations and asynchronous transition systems. In *Proceedings FST-TCS 92*, number 652 in LNCS, 1992.
92. M. Nielsen. CCS and its relationship to net theory. In *Proceedings Advances in Petri Nets 1986,* number 255 in LNCS, 1987.
93. M. Nielsen, U. H. Engberg, and K. S. Larsen. Fully abstract models for a process language with refinement. In *Proceedings REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, number 354 in LNCS, 1989.
94. M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, Part I. *Theoretical Computer Science*, 13(1):85–108, 1981.
95. M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Behavioural notions for elementary net systems. *Distributed Computing*, 4:45–57, 1990.
96. M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theoretical Computer Science*, 96(1):3–33, 1992.

97. M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Transition systems, event structures and unfoldings. *Information and Computation*, 118(2):191–207, 1995.
98. E.-R. Olderog. Operational Petri net semantics for CCSP. In *Proceedings Advances in Petri Nets 1987*, volume 266 of LNCS, 1987.
99. D. Park. Concurrency and automata on infinite sequences. In *Proceedings 5th GI-Conf. on Theoretical Computer Science*, number 104 in LNCS, 1981.
100. C.A. Petri. Nonsequential processes. Research Report 77-05, GMD, Sankt Augustin, 1977.
101. I. Phillips and I. Ulidowski. Reversibility and models for concurrency. *Electron. Notes Theor. Comput. Sci.*, 192(1):93–108, 2007.
102. G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
103. V.R. Pratt. On the composition of processes. In *Proceedings POPL 82*, 1982.
104. V.R. Pratt. Modelling concurrency with partial orders. *Journal of Parallel Programming*, 15(1), 1986.
105. C. Priami. *Enhanced Operational Semantics for Concurrency*. PhD thesis, University of Pisa, 1996.
106. C. Priami and G. D. Plotkin, editors. *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in Computer Science*. Springer, 2006.
107. A. Rabinovich and B.A. Trakhtenbrot. Behavior structures and nets. *Fundamenta Informaticae*, XI(4):357–404, 1988.
108. W. Reisig. *Petri Nets*. EATCS Monographs on Theoretical Computer Science. 1985.
109. E. W. Stark. Connections between a concrete and an abstract model of concurrent systems. In *Proceedings of the Fifth Conference on Mathematical Foundations of Program Semantics*, volume 442 of LNCS, pages 53–79, 1989.
110. G. Winskel. *Events in Computation*. PhD thesis, University of Edinburgh, 1980.
111. G. Winskel. Event structure semantics for CCS and related languages. In *Proceedings ICALP 82*, number 140 in LNCS, 1982.
112. G. Winskel. Categories of models for concurrency. In *Proceedings Seminar on concurrency*, number 197 in LNCS, 1984.
113. G. Winskel. A new definition of morphism on Petri nets. In *Proceedings STACS 84*, number 166 in LNCS, pages 140–150, 1984.
114. G. Winskel. Event structures. In *Proceedings Advances in Petri Nets 1986,* number 255 in LNCS, pages 325–392, 1987.
115. G. Winskel. Petri nets, algebras, morphisms and compositionality. *Information and Control*, 72:197–238, 1987.
116. G. Winskel. Symmetry and concurrency. In *Proceedings CALCO 07*, number 4624 in LNCS, pages 40–64, 2007.
117. G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of Logic in Computer Science*, volume 4, pages 1–148. Oxford, 1995.
118. D. Yankelevich. *Parametric Views of Process Description Languages*. PhD thesis, University of Pisa, 1993.