Event Structure Semantics of Parallel Extrusion in the pi-calculus

Silvia Crafa[1], Daniele Varacca[2], and Nobuko Yoshida[3]

[1] Dip. di Matematica - Univ. Padova
[2] PPS - CNRS & Univ. Paris Diderot
[3] Dept. of Computing - Imperial College London

Abstract. We give a compositional event structure semantics of the π-calculus. The main issues to deal with are the communication of free names and the extrusion of bound names. These are the source of the expressivity of the π-calculus, but they also allow subtle forms of causal dependencies. We show that free name communications can be modeled in terms of "incomplete/potential synchronization" events. On the other hand, we argue that it is not possible to satisfactorily model parallel extrusion within the framework of stable event structures. We propose to model a process as a pair $(\mathcal{E}, X)$ where $\mathcal{E}$ is a prime event structure and X is a set of (bound) names. Intuitively, $\mathcal{E}$ encodes the structural causality of the process, while the set X affects the computation on $\mathcal{E}$ so to capture the causal dependencies introduced by scope extrusion. The correctness of our true concurrent semantics is shown by an operational adequacy theorem with respect to the standard late semantics of the π-calculus.

1 Introduction

In the study of concurrent and distributed systems, the true-concurrent semantics approach takes concurrency as a primitive concept rather than reducing it to nondeterministic interleaving. One of the by-products of this approach is that the causal links between the process actions are more faithfully represented in true-concurrent models.

Prime event structures [NPW81] are a causal model for concurrency which is particularly suited for the traditional process calculi such as CCS and CSP since they directly represent causality and concurrency simply as a partial order and an irreflexive binary relation. A compositional event structure semantics of CCS has been proposed by Winskel [Win82], who also proved it to be operationally adequate with respect to the standard labelled transition semantics, hence sound with respect to bisimilarity. Similar results have been proved for variants of the π-calculus, namely for a restricted typed subcalculus [VY10] and for the internal πI-calculus [CVY07], which are however less expressive than the full calculus. In this paper we extend this result to the full π-calculus.

The main issues when dealing with the full π-calculus are name passing and the extrusion of bound names. These two ingredients are the source of the expressiveness of the calculus, but they are problematic in that they allow complex forms of causal dependencies, as detailed below.

1.1 Free name passing

Compared to pure CCS, (either free or bound) name passing adds the ability to dynamically acquire new synchronization capabilities. For instance consider the π-

calculus process $P = n(z).(\overline{z}\langle a\rangle.\mathbf{0} \mid m(x).\mathbf{0})$, that reads from the channel n and uses the received name to output the name a in parallel with a read action on m. Hence a synchronization along the channel m is possible if a previous communication along the channel n substitutes the variable z exactly with the name m. Then, in order to be compositional, the semantics of P must also account for "potential" synchronizations that might be activated by parallel compositions, like the one on the channel m.

To account for this phenomenon, we define the parallel composition of event structures so that synchronization events that involve input and output on different channels, at least one of which is a variable, are not deleted straight away. Moreover, the events produced by the parallel composition are relabelled by taking into account their causal history. For instance, the event corresponding to the synchronization pair $(\overline{z}\langle a\rangle, m(x))$ is relabelled into a τ action if, as in the process P above, its causal history contains a synchronization that substitutes the variable z with the name m.

1.2 The causal links created by name extrusion

Causal dependencies in π-calculus processes arise in two ways [BS98,DP99]: by nesting prefixes (called *structural* or *prefixing* or *subject* causality) and by using a name that has been bound by a previous action (called *link* or *name* or *object* causality). While subject causality is already present in CCS, object causality is distinctive of the π-calculus. The interactions between the two forms of causal dependencies are quite complex. We illustrate them by means of examples.

Parallel Scope extrusion. Consider the two processes $P = (\nu n)(\overline{a}\langle n\rangle.n(x).\mathbf{0})$ and $Q = (\nu n)(\overline{a}\langle n\rangle \mid n(x).\mathbf{0})$. The causal dependence of the action $n(x)$ from the output $\overline{a}\langle n\rangle$ is clear in the process P (i.e. there is a structural causal link), however, a similar dependence appears also in Q since a process cannot synchronize on the fresh name n before receiving it along the channel a (i.e. there is an objective causal link). Now consider the process $P_1 = (\nu n)(\overline{a}\langle n\rangle.\mathbf{0} \mid \overline{b}\langle n\rangle.\mathbf{0})$: in the standard interleaving semantics of pi-calculus only one output extrudes, either $\overline{a}\langle n\rangle$ or $\overline{b}\langle n\rangle$, and the other one does not. As a consequence, the second (free) output *depends* on the previous extruding output. However, in a true concurrent model we can hardly say that there is a dependence between the two parallel outputs, which in principle could be concurrently executed resulting in the parallel/simultaneous extrusion of the same name n to two different threads reading respectively on the channel a and on the channel b.

Dynamic addition of new extruders. We have seen that a bound name may have multiple extruders. In addition, the coexistence of free and bound outputs allows the set of extruders to dynamically change during the computation. Consider the process $P_2 = (\nu n)(\overline{a}\langle n\rangle \,|\, n(z)) \mid a(x).(\overline{x}\langle b\rangle \,|\, \overline{c}\langle x\rangle)$. It can either open the scope of n by extruding it along the channel a, or it can evolve to the process $(\nu n)(n(z) \mid \overline{n}\langle b\rangle \mid \overline{c}\langle n\rangle)$ where the output of the variable x has become a new extruder for both the actions with subject n. Hence after the first synchronization there is still the possibility of opening the scope of n by extruding it along the channel c.

The lesson we learned. The examples above show that the causal dependencies introduced by the scope extrusion mechanisms distinctive of the π-calculus can be understood in terms of the two ingredients of extrusion: name restriction and communication.

1. The restriction $(\nu n)P$ adds to the semantics of P a causal dependence between *every* action with subject n and *one of* the outputs with object n.
2. The communication of a restricted name adds *new* causal dependencies since both new extruders and new actions that need an extrusion may be generated by variable substitution.

A causal semantics for the π-calculus should account for such a dynamic additional objective causality introduced by scope extrusion. In particular, the first item above hints at the fact that we have to deal with a form of disjunctive (objective) causality. Prime event structures are stable models that represent disjunctive causality by duplicating events and so that different copies causally depend on different (alternative) events. In our case this amounts to represent different copies of any action with a bound subject, each one causally depending on different (alternative) extrusions. However, the fact that the set of extruders dynamically changes complicates the picture since new copies of any action with a bound subject should be dynamically spawned for each new extruder. In this way the technical details quickly become intractable, as discussed in Section 6.

In this paper we follow a different approach, that leads to extremely simple technical details. The idea is to represent the disjunctive objective causality in a so-called inclusive way: in order to trace the causality introduced by scope extrusion it is sufficient to ensure that whenever an action with a bound subject is executed, at least one extrusion of that bound name must have been already executed, but it is not necessary to record which output was the real extruder. Clearly, such an inclusive-disjunctive causality is no longer representable with stable structures like prime event structures. However, we show that an operational adequate true concurrent semantics of the π-calculus can be given by encoding a π-process simply as as a pair $(\mathcal{E}, X)$ where $\mathcal{E}$ is a prime event structure and X is a set of (bound) names. Intuitively, the causal relation of $\mathcal{E}$ encodes the structural causality of a process. Instead, the set X affects the computation on $\mathcal{E}$: we define a notion of *permitted configurations*, ensuring that any computation that contains an action whose subject is a bound name in X, also contains a previous extrusion of that name. Hence a further benefit of this semantics is that it clearly accounts for both forms of causality: subjective causality is captured by the causal relation of event structures, while objective causality is implicitly captured by permitted configurations.

2 The π-calculus

In this section we illustrate the synchronous, monadic π-calculus that we consider. We presuppose a countably-infinite set of names and a countably-infinite set of variables ranged over by $m, .., q$ and by $x, .., z$, respectively. We use a, b, c to range over both names and variables.

$$\begin{array}{lll} \textit{Prefixes} & \pi ::= a(x) \mid \overline{a}\langle b\rangle & \textit{Definitions} \quad A(\tilde{x}, \tilde{p} \mid \mathbf{z}, \mathbf{n}) = P_A \\ \textit{Processes} & P, Q ::= \sum_{i\in I} \pi_i.P_i \mid P \mid Q \mid (\nu n)P \mid A\langle \tilde{x}, \tilde{p} \mid \mathbf{z}, \mathbf{n}\rangle & \end{array}$$

The syntax consists of the parallel composition, name restriction, finite summation of guarded processes and recursive definition. In $\sum_{i\in I} \pi_i.P_i$, I is a finite indexing set; when I is empty we simply write $\mathbf{0}$ and denote with $+$ the binary sum. A process $a(x).P$ can perform an input at a and the variable x is the placeholder for the name so received. The

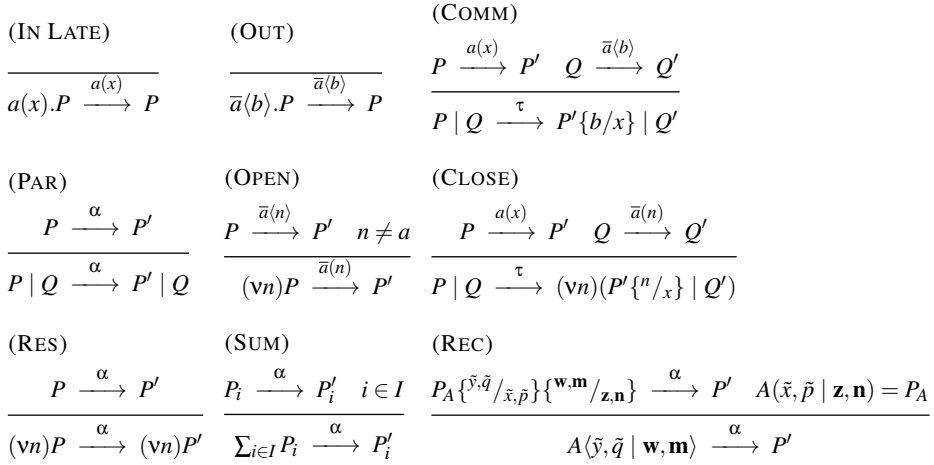


Fig. 1. Labelled Transition System of the π-calculus

output case is symmetric: a process $\overline{a}\langle b\rangle.P$ can perform an output along the channel a. Notice that an output can send a name (either free or restricted) or else a variable.

We assume that every constant A has a unique defining equation $A(\tilde{x},\tilde{p} \mid \mathbf{z},\mathbf{n}) = P_A$. The symbol $\tilde{p}$, resp. $\tilde{x}$, denotes a tuple of distinct names, resp. variables, that correspond to the free names, resp. variables, of P_A. $\mathbf{n}$, resp. $\mathbf{z}$, represents an infinite sequence of distinct names $\mathbb{N} \to \mathit{Names}$, resp. distinct variables $\mathbb{N} \to \mathit{Variables}$, that is intended to enumerate the (possibly infinite) bound names, resp. bound variables, of P_A. The parameters $\mathbf{n}$ and $\mathbf{z}$ do not usually appear in recursive definitions in the literature. The reason we add them is that we want to maintain the following Basic Assumption: *Every bound name/variable is different from any other name/variable, either bound or free*. In the π-calculus, this policy is usually implicit and maintained along the computation by dynamic α-conversion: every time the definition A is unfolded, a copy of the process P_A is created whose bound names and variables must be fresh. This dynamic choice is difficult to interpret in the event structures. Hence, in order to obtain a precise semantic correspondence, our recursive definitions prescribe all the names and variables that will be possibly used in the recursive process (see [CVY07] for some example). Notice that this assumption has no impact on the process behaviour since every π-process can be α-renamed so that it satisfies that assumption.

The sets of free and bound names and free and bound variables of P, denoted by $\mathrm{fn}(P), \mathrm{bn}(P), \mathrm{fv}(P), \mathrm{bv}(P)$, are defined as usual but for constant processes, whose definitions are as follows: $\mathrm{fn}(A\langle \tilde{x},\tilde{p} \mid \mathbf{z},\mathbf{n}\rangle) = \{\tilde{p}\}$, $\mathrm{bn}(A\langle \tilde{x},\tilde{p} \mid \mathbf{z},\mathbf{n}\rangle) = \mathbf{n}(\mathbb{N})$, $\mathrm{fv}(A\langle \tilde{x},\tilde{p} \mid \mathbf{z},\mathbf{n}\rangle) = \{\tilde{x}\}$ and $\mathrm{bv}(A\langle \tilde{x},\tilde{p} \mid \mathbf{z},\mathbf{n}\rangle) = \mathbf{z}(\mathbb{N})$. The operational semantics is given in Figure 1 in terms of an LTS (in late style) where we let α,β range over the set of labels $\{\tau, a(x), \overline{a}\langle b\rangle, \overline{a}(n)\}$. The syntax of labels shows that the object of an input is always a variable, whereas the object of a free output is either a variable (e.g. $b(x)$ or $\overline{a}\langle x\rangle$) or a name. On the other hand, the object of a bound output is always a name, since it must occur under a restriction. Moreover, thanks to the Basic Assumption, the side conditions in rules (PAR) and (RES) are not needed anymore.

3 Event structures

This section reviews basic definitions of prime event structures [DDNM88,NPW81,Win87].

Definition 1 (Labelled Event Structure). *Let L be a set of labels. A labelled event structure is a tuple $\mathcal{E} = \langle E, \leq, \smile, \lambda \rangle$ s.t.*

- *E is a countable set of* events*;*
- *$\langle E, \leq \rangle$ is a partial order, called the* causal order*;*
- *for every $e \in E$, the set $\lfloor e) := \{e' \mid e' < e\}$, called the* enabling set *of e, is finite;*
- *$\smile$ is an irreflexive and symmetric relation, called the* conflict relation*, satisfying the following: for every $e_1, e_2, e_3 \in E$ if $e_1 \leq e_2$ and $e_1 \smile e_3$ then $e_2 \smile e_3$.*
- *$\lambda : E \to L$ is a labelling function that associates a label to each event in E.*

Intuitively, labels represent *actions*, and events should be thought of as *occurrences of actions*. Labels allow us to identify events which represent different occurrences of the same action. In addition, labels are essential when composing two event structures in a parallel composition, as they identify which events correctly synchronise.

We say that the conflict $e_2 \smile e_3$ is *inherited* from the conflict $e_1 \smile e_3$, when $e_1 < e_2$. If a conflict $e_1 \smile e_2$ is not inherited from any other conflict we say that it is *immediate*. If two events are not causally related nor in conflict they are said to be *concurrent*.

The notion of computation is usually captured in event structures in terms of configurations. A *configuration* C of an event structure $\mathcal{E}$ is a conflict free downward closed subset of E, i.e. a subset C of E satisfying: (1) if $e \in C$ then $\lfloor e) \subseteq C$ and (2) for every $e, e' \in C$, it is not the case that $e \smile e'$, that is e and e' are either causally dependent or concurrent. In other words, a configuration represents a run of an event structure, where events are partially ordered. The set of configurations of $\mathcal{E}$, partially ordered by inclusion, is denoted as $\mathcal{L}(\mathcal{E})$. Another notion of computation can be defined in terms of the following notion of labelled transition systems over event structures. We will use this definition to easily prove that the computational steps of a π-calculus process are reflected into its event structure semantics.

Definition 2 (LTS of event structures). *Let $\mathcal{E} = \langle E, \leq, \smile, \lambda \rangle$ be a labelled event structure and let e be one of its minimal events such that $\lambda(e) = \beta$. Then we write $\mathcal{E} \xrightarrow{\beta} \mathcal{E}\lfloor e$, where $\mathcal{E}\lfloor e$ is the event structure $\langle E', \leq_{|E'}, \smile_{|E'}, \lambda_{E'} \rangle$ with $E' = \{e' \in E \mid e' \neq e$ and $e' \not\smile e\}$.*

Roughly speaking, $\mathcal{E}\lfloor e$ is $\mathcal{E}$ minus the event e, and minus all events that are in conflict with e. The reachable LTS with initial state $\mathcal{E}$ corresponds to the computations over $\mathcal{E}$.

Event structures have been shown to be the class of objects of a category [WN95]. Moreover, it is easily shown that an isomorphism in this category is a label-preserving bijective function that preserves and reflects causality and conflict. We denote by $\mathcal{E}_1 \cong \mathcal{E}_2$ the fact that there is an isomorphism between $\mathcal{E}_1$ and $\mathcal{E}_2$.

We review here an informal description of several operations on labelled event structures, that we are going to use in the next section. See [Win87] for more details.

- *Prefixing $a.\mathcal{E}$.* This operation adds to the event structure a new minimal element, labelled by a, below every other event in $\mathcal{E}$.

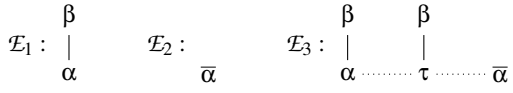
- *Prefixed sum* $\sum_{i\in I} a_i.\mathcal{E}_i$. This is obtained as the disjoint union of copies of the event structures $a_i.\mathcal{E}_i$. The conflict relation is extended by putting in conflict every pair of events belonging to two different copies of $a_i.\mathcal{E}_i$.
- *Restriction* (or *Hiding*) $\mathcal{E} \setminus X$ where $X \subseteq L$ is a set of labels. This is obtained by removing from E all events with label in X and all events that are above (i.e., causally depend on) one of those.
- *Relabelling* $\mathcal{E}[f]$ where L and L' are two sets of labels and $f : L \to L'$. This operation just consists in composing the labelling function λ of $\mathcal{E}$ with the function f. The new event structure is labelled over L' and its labelling function is $f \circ \lambda$.

The parallel composition of two event structures $\mathcal{E}_1$ and $\mathcal{E}_2$ gives a new event structure $\mathcal{E}'$ whose events model the parallel occurrence of pairs of events $e_1 \in E_1$ and $e_2 \in E_2$. In particular, when the labels of e_1 and e_2 match according to an underlying synchronisation model, $\mathcal{E}'$ records (with an event $e' \in E'$) that a synchronisation between e_1 and e_2 is possible, and deals with the causal effects of such a synchronisation.

Technically, the parallel composition is defined as the categorical product followed by restriction and relabelling [WN95]. The categorical product represents all conceivable synchronisations, relabelling implements a synchronisation model by expressing which events are allowed to synchronise, and hiding removes synchronisations that are not permitted. The synchronisation model underlying the relabelling operation is formalised by the notion of *synchronisation algebra*.A synchronisation algebra S is a partial binary operation $\bullet_S$ defined on $L_* := L \uplus \{*\}$ where $*$ is a distinguished label. If α_i are the labels of events $e_i \in E_i$, then the event $e' \in E'$ representing the synchronisation of e_1 and e_2 is labelled by $\alpha_1 \bullet_S \alpha_2$. When $\alpha_1 \bullet_S \alpha_2$ is undefined, the synchronisation event e' is given a distinguished label bad, indicating that this event is not allowed and should be deleted.

Definition 3 (Parallel Composition of Event Structures). *Let $\mathcal{E}_1, \mathcal{E}_2$ two event structures labelled over L, let S be a synchronisation algebra, and let $f_S : L_* \to L' = L_* \cup \{\mathsf{bad}\}$ be a function defined as $f_S(\alpha_1, \alpha_2) = \alpha_1 \bullet_S \alpha_2$, if S is defined on (α_1, α_2), and $f_S(\alpha_1, \alpha_2) = \mathsf{bad}$ otherwise. The parallel composition $\mathcal{E}_1 \|_S \mathcal{E}_2$ is defined as the categorical product followed by relabelling and restriction:$\mathcal{E}_1 \|_S \mathcal{E}_2 = (\mathcal{E}_1 \times \mathcal{E}_2)[f_S] \setminus \{\mathsf{bad}\}$. The subscripts S are omitted when the synchronisation algebra is clear from the context.*

Example 4. We show a simple example of parallel composition. Consider the set of labels $L = \{\alpha, \beta, \overline{\alpha}, \tau\}$ and the synchronisation algebra obtained as the symmetric closure of the following rules: $\alpha \bullet \overline{\alpha} = \tau$, $\alpha \bullet * = \alpha$, $\overline{\alpha} \bullet * = \overline{\alpha}$, $\beta \bullet * = \beta$ and undefined otherwise. Consider the two event structures $\mathcal{E}_1, \mathcal{E}_2$, where $E_1 = \{a, b\}, E_2 = \{a'\}$, with $a \leq_1 b$ and $\lambda_1(a) = \alpha, \lambda_1(b) = \beta, \lambda_2(a') = \overline{\alpha}$. The event structures are represented as follows:



where dotted lines represent immediate conflict, while the causal order proceeds upwards along the straight lines. Then $\mathcal{E}_3 := \mathcal{E}_1 \| \mathcal{E}_2$ is the event structure $\langle E_3, \leq, \smile, \lambda \rangle$ where $E_3 = \{e{:=}(\emptyset, a, *), e'{:=}(\emptyset, *, a'), e''{:=}(\emptyset, a, a'), d{:=}(\{e\}, a', *), d''{:=}(\{e''\}, a', *)\}$, and the ordering, immediate conflict and the labelling are as in the picture above.

We say that an event structure $\mathcal{E}$ is a *prefix* of an event structure $\mathcal{E}'$, denoted $\mathcal{E} \leq \mathcal{E}'$ if there exists $\mathcal{E}'' \cong \mathcal{E}'$ such that $E \subseteq E''$ and no event in $E'' \setminus E$ is below any event of E. Winskel [Win82] has shown that the class of event structures with the prefix order is a large CPO, and thus the limits of countable increasing chains exist. Moreover all operators on event structures are continuous. We will use this fact to define the semantics of the recursive definitions.

4 Free Name Passing

We present the event structure semantics of the full π-calculus in two phases, dealing separately with the two main issues of the calculus. We start in this section discussing free name passing, and we postpone to the next section the treatment of scope extrusion.

The core of a compositional semantics of a process calculus is parallel composition. When a process P is put in parallel with another process, new synchronizations can be triggered. Hence the semantics of P must also account for "potential" synchronizations that might be activated by parallel compositions. In Winskel's event structure semantics of CCS [Win82], the parallel composition is defined as a product in a suitable category followed by relabelling and hiding, as we have presented in Section 3. For the semantics of the π-calculus, when the parallel composition of two event structures is computed, synchronisation events that involve input and output on different channels cannot be hidden straight away. If at least one of the two channels is a variable, then it is possible that, after prefixing and parallel composition, the two channels will be made equal.

We then resort to a technique similar to the one used in [CVY07]: we consider a generalized notion of relabelling that takes into account the history of a (synchronization) event. Such a relabelling is defined according to the following ideas:

- each pair $(a(x), \overline{a}\langle b \rangle)$ made of two equal names or two equal variables is relabelled $\tau_{x \to b}$, indicating that it represents a legal synchronization where b is substituted for x. Moreover, such a substitution must be propagated in all the events that causally depend on this synchronization. Anyway, after all substitution have taken place, there is no need to remember the extra information carried by the τ action, than the subscripts of the τ events are erased.
- Synchronisations pairs, like $(a(x), \overline{b}\langle c \rangle)$, that involve different channels (at least) one of which is a variable, is relabelled $(a(x), \overline{b}\langle c \rangle)_{x \to c}$, postponing the decision whether they represent a correct synchronization or not.
- Each pair $(n(x), \overline{m}\langle b \rangle)$ made of two different names is relabelled bad to denote a synchronization that is not allowed.

Definition 5 (Generalised Relabelling). *Let L and L' be two sets of labels, and let $Pom(L')$ be a pomset (i.e., partially ordered multiset) of labels in L'. Given an event structure $\mathcal{E} = \langle E, \leq, \smile, \lambda \rangle$ over the set of labels L, and a function $f : Pom(L') \times L \longrightarrow L'$, we define the relabelling operation $\mathcal{E}[f]$ as the event structure $\mathcal{E}' = \langle E, \leq, \smile, \lambda' \rangle$ with labels in L', where $\lambda' : E \longrightarrow L'$ is defined as follows by induction on the height of an element of E: if $h(e) = 0$ then $\lambda'(e) = f(\emptyset, \lambda(e))$, if $h(e) = n+1$ then $\lambda'(e) = f(\lambda'([e)), \lambda(e))$.*

In words, an event e is relabelled with a label $\lambda'(e)$ that depends on the (pomset of) labels of the events belonging to its causal history $[e)$.

In the case of π-calculus with free names, let $L = \{a(x), \overline{a}\langle b\rangle, \tau \mid a,b \in \mathit{Names} \cup \mathit{Variables},\ x \in \mathit{Variables}\}$ be the set of labels used in the LTS of π-calculus without restriction. We define the relabelling function needed by the parallel composition operation around the extended set of labels $L' = L \cup \{(\alpha,\beta)_{x\to b} \mid \alpha,\beta \in L\} \cup \{\tau_{x\to b},\ \mathsf{bad}\}$, where bad is a distinguished label. The relabelling function $f_\pi : \mathit{Pom}(L') \times (L' \uplus \{*\} \times L' \uplus \{*\}) \longrightarrow L'$ is defined as follows (we omit the symmetric clauses):

$$f_\pi(X, \langle a(y), \overline{a}\langle b\rangle\rangle) = \tau_{y\to b} \qquad f_\pi(X, \langle a(x), \overline{y}\langle c\rangle\rangle) = \begin{cases} \tau_{x\to c} & \text{if } \alpha_{y\to a} \in X \\ (a(x), \overline{y}\langle c\rangle)_{x\to c} & \text{otherwise} \end{cases}$$

$$f_\pi(X, \langle n(y), \overline{m}\langle b\rangle\rangle) = \mathsf{bad} \qquad f_\pi(X, \langle y(x), \overline{a}\langle n\rangle\rangle) = \begin{cases} \tau_{x\to n} & \text{if } \alpha_{y\to a} \in X \\ (y(x), \overline{a}\langle n\rangle)_{x\to n} & \text{otherwise} \end{cases}$$

$$f_\pi(X, \langle y(x), *\rangle) = \begin{cases} a(x) & \text{if } \alpha_{y\to a} \in X \\ y(x) & \text{otherwise} \end{cases} \qquad f_\pi(X, \langle \overline{y}\langle b\rangle, *\rangle) = \begin{cases} \overline{a}\langle b\rangle & \text{if } \alpha_{y\to a} \in X \\ \overline{y}\langle b\rangle & \text{otherwise} \end{cases}$$

$$f_\pi(X, \langle \alpha, *\rangle) = \alpha \qquad f_\pi(X, \langle \alpha, \beta\rangle) = \mathsf{bad} \quad \text{otherwise}$$

The extra information carried by the τ-actions, differently from that of "incomplete synchronization" events, is only necessary in order to *define* the relabelling, but there is no need to keep it after the synchronization has been completed. Hence we apply a second relabelling *er* that simply erases the subscript of τ actions.
The semantics of the π-calculus is then defined as follows by induction on processes, where the parallel composition of event structure is defined by

$$\mathcal{E}_1 \|_\pi \mathcal{E}_2 = ((\mathcal{E}_1 \times \mathcal{E}_2)\ [f_\pi][er]) \setminus \{\mathsf{bad}\}$$

To deal with recursive definitions, we use an index k to denote the level of unfolding.

$$\{\!|\, \mathbf{0} \,|\!\}_k = \emptyset \qquad \{\!|\, \textstyle\sum_{i\in I} \pi_i.P_i \,|\!\}_k = \sum_{i\in I} \pi_i.\{\!|\, P_i \,|\!\}_k \qquad \{\!|\, P \mid Q \,|\!\}_k = \{\!|\, P \,|\!\}_k \parallel_\pi \{\!|\, Q \,|\!\}_k$$

$$\{\!|\, A\langle \tilde{y}, \tilde{q} \mid \mathbf{w}, \mathbf{m}\rangle \,|\!\}_0 = \emptyset \qquad \{\!|\, A\langle \tilde{y}, \tilde{q} \mid \mathbf{w}, \mathbf{m}\rangle \,|\!\}_{k+1} = \{\!|\, P_A\{^{\tilde{y},\tilde{q}}/_{\tilde{x},\tilde{p}}\}\{^{\mathbf{w},\mathbf{m}}/_{\mathbf{z},\mathbf{n}}\} \,|\!\}_k$$

Recall that all operators on event structures are continuous with respect to the prefix order. It is thus easy to show that, for any k, $\{\!|\, P \,|\!\}_k \leq \{\!|\, P \,|\!\}_{k+1}$. We define $\{\!|\, P \,|\!\}$ to be the limit of the increasing chain $...\{\!|\, P \,|\!\}_k \leq \{\!|\, P \,|\!\}_{k+1} \leq \{\!|\, P \,|\!\}_{k+2}...$, that is $\{\!|\, P \,|\!\} = \sup_{k\in\mathcal{N}} \{\!|\, P \,|\!\}_k$
Since all operators are continuous w.r.t. the prefix order we have the following result:

Theorem 6 (Compositionality). *The semantics $\{\!|\, P \,|\!\}$ is compositional, i.e. $\{\!|\, P \mid Q \,|\!\} = \{\!|\, P \,|\!\} \parallel_\pi \{\!|\, Q \,|\!\}$, $\{\!|\, \sum_{i\in I} \pi_i.P_i \,|\!\} = \sum_{i\in I} \pi_i.\{\!|\, P_i \,|\!\}$,*

Example 7. As an example, consider the process $P = n(z).(\overline{z}\langle a\rangle \mid m(x))$. The synchronization along the channel m can be only performed if the previous synchronization along n substitutes the variable z with the name m. Accordingly, the semantics of the process P is the leftmost event structure in Figure 2, denoted by $\mathcal{E}_P$. Moreover, the rightmost structure in Figure 2 corresponds to the semantics of the process $P \mid \overline{n}\langle m\rangle \mid \overline{n}\langle p\rangle$.

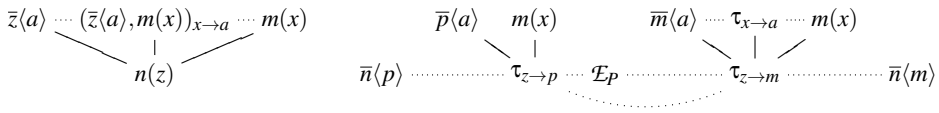


Fig. 2.

The following theorem shows that the event structure semantics is operationally correct. Indeed, given a process P, the computational steps of P in the LTS of Section 2 are reflected by the semantics $\{\!|\, P \,|\!\}$.

Theorem 8 (Operational Adequacy). *Let* $\beta \in L = \{a(x), \overline{a}\langle b\rangle, \tau\}$. *Suppose* $P \xrightarrow{\beta} P'$ *in the* π*-calculus. Then* $\{\!|\, P \,|\!\} \xrightarrow{\beta} \cong \{\!|\, P' \,|\!\}$. *Conversely, suppose* $\{\!|\, P \,|\!\} \xrightarrow{\beta} \mathcal{E}'$. *Then there exists* P' *such that* $P \xrightarrow{\beta} P'$ *and* $\{\!|\, P' \,|\!\} \cong \mathcal{E}'$.

Note that the correspondence holds for those labels that appear in the LTS of the calculus. Labels that identify "incomplete synchronizations" have been introduced in the event structure semantics for the sake of compositionality, but they are not considered in the theorem above since they do not correspond to any operational step.

5 Scope extrusion

In this section we show how the causal dependencies introduced by scope extrusion can be captured by event structure-based models. As we discussed in Section 1, the communication of bound names implies that any action with a bound subject causally depends on a dynamic set of possible extruders of that bound subject. Hence dealing with scope extrusion requires modelling some form of disjunctive causality. Prime event structures are stable models that represent an action α that can be caused either by the action β_1 or the action β_2 as two different events e, e' that are both labelled α but e causally depends on the event labeled β_1 while e' is caused by the event labelled β_2. In order to avoid the proliferation of events representing the same action with different extruders, we follow here a different approach, postponing to the next section a more detailed discussion on the use of prime event structures.

5.1 Event structure with bound names

We define the semantics of the full π-calculus in terms of pairs $(\mathcal{E}, X)$, where $\mathcal{E}$ is a prime event structure, and is a X a set of names. We call such a pair an *event structure with bound names*. Intuitively, the causal relation of $\mathcal{E}$ encodes the structural causality of a process, while the set X records bound names. The names in the set X affect the notion of computation on $\mathcal{E}$: for instance, if a minimal event in $\mathcal{E}$ has a label whose subject is in X, then such a minimal event cannot be executed since it requires a previous extrusion. In other terms, given a pair $(\mathcal{E}, X)$ we define a notion of *permitted configurations*, ensuring that any computation that contains an action whose subject is a bound name, also contains a previous extrusion of that name. Objective causality is then implicitly captured by permitted configurations.

Definition 9 (Semantics). *The semantics of the full π-calculus is inductively defined as follows, where k denote the level of unfolding of recursive definitions, and we write $\mathcal{E}_P^k$, resp. X_P^k, for the first, resp. the second, projection of the pair $\{\!|\, P \,|\!\}_k$*

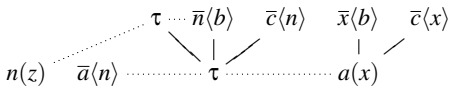
$$
\begin{aligned}
\{\!|\, \mathbf{0} \,|\!\}_k &= (\emptyset, \emptyset) \\
\{\!|\, \textstyle\sum_{i\in I} \pi_i.P_i \,|\!\}_k &= (\ \textstyle\sum_{i\in I} \pi_i.\mathcal{E}_{P_i}^k\ ,\ \ \biguplus_{i\in I} X_{P_i}^k\ ) \\
\{\!|\, P \mid Q \,|\!\}_k &= (\ \mathcal{E}_P^k \parallel_\pi \mathcal{E}_Q^k\ ,\ \ X_P^k \uplus X_Q^k\ ) \\
\{\!|\, (\nu n)P \,|\!\}_k &= (\ \mathcal{E}_P^k\ ,\ \ X_P^k \uplus \{n\}\ ) \\
\{\!|\, A\langle \tilde{y}, \tilde{q} \mid \mathbf{w}, \mathbf{m} \rangle \,|\!\}_0 &= (\emptyset, \{\mathbf{w}(\mathbb{N})\}) \\
\{\!|\, A\langle \tilde{y}, \tilde{q} \mid \mathbf{w}, \mathbf{m} \rangle \,|\!\}_{k+1} &= (\ \mathcal{E}^k_{P_A\{\tilde{y},\tilde{q}/\tilde{x},\tilde{p}\}\{\mathbf{w},\mathbf{m}/\mathbf{z},\mathbf{n}\}},\ \ \{\mathbf{w}(\mathbb{N})\}\ )
\end{aligned}
$$

It is easy to show that, for any k, $\mathcal{E}_P^k \leq \mathcal{E}_P^{k+1}$ and $X_P^k = X_P^{k+1} = X_P$. Then the semantics of a process P is defined as the following limit: $\{\!|\, P \,|\!\} = (\ \sup_{k\in\mathcal{N}} \mathcal{E}_P^k\ ,\ X_P\ )$.

This semantics is surprisingly simple: a restricted process like $(\nu n)P$ is represented by a prime event structure that encodes the process P where the scope of n has been opened, and we collect the name n in the set of bound names. As for parallel composition, the semantics $\{\!|\, P \mid Q \,|\!\}$ is a pair $(\mathcal{E}, X)$ where X collects the bound names of both $\{\!|\, P \,|\!\}$ and $\{\!|\, Q \,|\!\}$ (remind that we assumed that bound names are pairwise different), while the event structure $\mathcal{E}$ is obtained exactly as in the previous sections. This is since the event structures that get composed correspond to the processes P and Q where the scope of any bound name has been opened. It is immediate to prove the following property.

Proposition 10. *Let P, Q be two processes, then $\{\!|\, (\nu n)P \mid Q \,|\!\} = \{\!|\, (\nu n)(P \mid Q) \,|\!\}$.*

Example 11. Consider the process $P = (\nu n)(\overline{a}\langle n\rangle \mid n(z)) \mid a(x).(\overline{x}\langle b\rangle \mid \overline{c}\langle x\rangle)$, whose first synchronization produces a new extruder $\overline{c}\langle n\rangle$ for the bound name n. The semantics of P is the pair $(\mathcal{E}_P, \{n\})$, where $\mathcal{E}_P$ is the following e.s.:



In order to study the operational correspondence between the LTS semantics of the π-calculus and the event structure semantics above, we first need to adapt the notion of computational steps of the pairs $(\mathcal{E}, X)$. The definition of labelled transitions between prime event structures, i.e., Definition 2, is generalized as follows.

Definition 12 (Permitted Transitions). *Let $(\mathcal{E}, X)$ be a labelled event structure with bound names. Let e be a minimal event of $\mathcal{E}$ with $\lambda(e) = \beta$. We define the following permitted labelled transitions:*

- $(\mathcal{E}, X) \xrightarrow{\beta} (\mathcal{E}\lfloor e,\ X)$, *if* $\beta \in \{\tau, a(x), \overline{a}\langle b\rangle\}$ *with* $a, b \notin X$.
- $(\mathcal{E}, X) \xrightarrow{\overline{a}(n)} (\mathcal{E}\lfloor e,\ X\backslash\{n\})$, *if* $\beta = \overline{a}\langle n\rangle$ *with* $a \notin X$ *and* $n \in X$.

According to this definition, the set of bound names constrains the set of transitions that can be performed. In particular, no transition whose label has a bound subject is allowed. On the other hand, when a minimal event labeled $\overline{a}\langle n\rangle$ is consumed, if the name n is bound, the transition's labels records that this event is indeed a bound output. Moreover, in this case we record that the scope of n is opened by removing n from the set of bound names of the target pair. Finally, observe that the previous definition only allows transitions whose labels are in the set $L = \{\tau, a(x), \overline{a}\langle b\rangle, \overline{a}(n)\}$, which is exactly the sets of labels in the LTS of Section 2.

Theorem 13 (Operational Adequacy). *Let $\beta \in L = \{a(x), \overline{a}\langle b\rangle, \overline{a}(n), \tau\}$. Suppose $P \xrightarrow{\beta} P'$ in the π-calculus. Then $\{\!|P|\!\} \xrightarrow{\beta} \cong \{\!|P'|\!\}$. Conversely, suppose $\{\!|P|\!\} \xrightarrow{\beta} \mathcal{E}'$. Then there exists P' such that $P \xrightarrow{\beta} P'$ and $\{\!|P'|\!\} \cong \mathcal{E}'$.*

5.2 Subjective and Objective Causality

Given an event structure with bound names $(\mathcal{E}, X)$, Definition 12 shows that some configuration of $\mathcal{E}$ is no longer allowed. For instance, if e is minimal but its label has a subject that is a name in X, e.g, $\lambda(e) = n(x)$ with $n \in X$, then the configuration $\{e\}$ is no longer allowed since the event e requires a previous extrusion of the name n.

Definition 14 (Permitted Configuration). *Let $(\mathcal{E}, X)$ be an event structure with bound names. Given a configuration C of $\mathcal{E}$, we say that C is* permitted *in $(\mathcal{E}, X)$ whenever, for any $e \in C$ whose label has subject n with $n \in X$,*

- *$C \setminus \{e\}$ is permitted, and*
- *$C \setminus \{e\}$ contains an event e' whose label is an output action with object n.*

The first item of the definition above is used to avoid circular definitions that would allow wrong configurations like $\{\overline{n}\langle m\rangle, \overline{m}\langle n\rangle\}$ with $X = \{n, m\}$. Now, the two forms of causality of the π-calculus can be defined using event structures with bound names and permitted configurations.

Definition 15 (Subjective and Objective Causality). *Let P be a process of the π-calculus, and $\{\!|P|\!\} = (\mathcal{E}_p, X_P)$ be its semantics. Let be $e_1, e_2 \in E_P$, then*

- *e_2 has a* subjective dependence *on e_1 if $e_1 \leq_{\mathcal{E}_P} e_2$;*
- *e_2 has a* objective dependence *on e_1 if (i) the label of e_1 is the output of a name in X which is also the subject of the label of e_2, and if (ii) there exists a configuration C that is permitted in $(\mathcal{E}, X)$ and that contains both e_1 and e_2.*

Example 16. Let consider again the process P in Example 11. The configurations $C_1 = \{\overline{a}\langle n\rangle, n(z)\}$ and $C_2 = \{\tau, \overline{c}\langle n\rangle, n(z)\}$ are both permitted by $\{\!|P|\!\}$, and they witness the fact that the action $n(z)$ has an objective dependence on $\overline{a}\langle n\rangle$ and[4] on $\overline{c}\langle n\rangle$.

Example 17. Let be $P = (\nu n)(\overline{a}\langle n\rangle \mid \overline{b}\langle n\rangle \mid n(x))$, then $\{\!|P|\!\} = (\mathcal{E}_P, \{n\})$ where $\mathcal{E}_P$ has three concurrent events. In this process there is no subjective causality, however the action $n(x)$ has an objective dependence on $\overline{a}\langle n\rangle$ and on $\overline{b}\langle n\rangle$ since both $C_1 = \{\overline{a}\langle n\rangle, n(x)\}$ and $C_2 = \{\overline{b}\langle n\rangle, n(x)\}$ are permitted configurations.

[4] We could also say that $n(z)$ objectively depends *either* on $\overline{a}\langle n\rangle$ *or* on $\overline{c}\langle n\rangle$.

Example 18. Let be $P = (\nu n)(\overline{a}\langle n\rangle.\overline{b}\langle n\rangle.n(x))$, then $\{\!|P|\!\} = (\mathcal{E}_P, \{n\})$ where $\mathcal{E}_P$ is a chain of three events. According to the causal relation of $\mathcal{E}_P$, the action $n(x)$ has a structural dependence on both the outputs. Moreover, the permitted configuration $C = \{\overline{a}\langle n\rangle, \overline{b}\langle n\rangle, n(x)\}$ shows that $n(x)$ has an objective dependence on $\overline{a}\langle n\rangle$ and on $\overline{b}\langle n\rangle$[5].

5.3 The meaning of labelled causality

In this paper we focus on compositional semantics, studying a true concurrent semantics that operationally matches the LTS semantics of the π-calculus. Alternatively, one could take as primitive the reduction semantics of the π-calculus, taking the perspective that only τ-events are "real" computational steps of a concurrent system. Therefore one could argue that the concept of causal dependency makes only sense between τ events. In this perspective, we propose to interpret the causal relation between non-τ events as an anticipation of the causal relations involving the synchronizations they will take part in. In other terms, non-τ events (from now on simply called labelled events) represent "incomplete" events, that are waiting for a synchronization or a substitution to be completed. Hence we can prove that in our semantics two labelled events e_1 and e_2 are causally dependent *if and only if* the τ-events they can take part in are causally dependent. This property is expressed by the following theorem in terms of permitted configurations. Recall that the parallel composition of two event structures is obtained by first constructing the cartesian product. Therefore there are projection morphisms π_1, π_2 on the two composing structures. Let call τ-configuration a configuration whose events are all τ-events. Note that every τ-configuration is permitted.

Theorem 19. *Let P be a process. A configuration C is permitted in $\{\!|P|\!\}$ if and only if there exists a process Q and a τ-configuration C' in $\{\!|P \mid Q|\!\}$ such that $\pi_1(C') = C$.*

Let e_1, e_2 be two labelled events of $\{\!|P|\!\} = (\mathcal{E}_p, X_p)$. If e_1, e_2 are structurally dependent, i.e., $e_1 \leq_{\mathcal{E}_P} e_2$, then such a structural dependence is preserved and reflected in the τ-actions they are involved in because of the way the parallel composition of event structures is defined. On the other hand, let be e_1, e_2 objectively dependent. Consider the parallel composition $(\mathcal{E}_P || _\pi \mathcal{E}_Q, X_P \cup X_Q)$ for some Q such that there is a τ events e'_2 in $\mathcal{E}_P \,||_\pi \mathcal{E}_Q$ with $\pi_1(e'_2) = e_2$ and $[e'_2)$ is a τ-configuration. Then there must be an event $e'_1 \in [e'_2)$ such that $\pi_1(e'_1) = e_1$.

6 Disjunctive causality

As we discussed in Section 1, objective causality introduced by scope extrusion requires for the π-calculus a semantic model that is able to express some form of disjunctive causality. In the previous section we followed an approach that just ensures that some extruder (causally) precedes any action with a bound subject. However, we could alternatively take the effort of tracing the identity of the actual extruders. We could do it by duplicating the events corresponding to actions with bound subject and letting different copies depend on different, alternative, extruders. Such a duplication allows to use

[5] In this case we do not know which of the two outputs really extruded the bound name, accordingly to the inclusive disjunctive causality approach we are following.

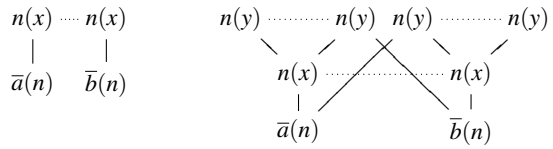
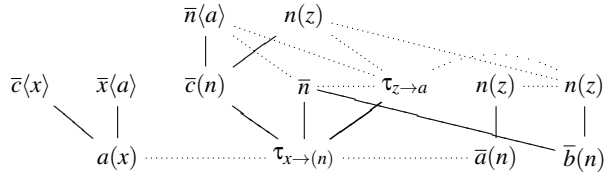


Fig. 3.

prime event structures as semantics models. In this section we discuss this alternative approach showing to what extent it can be pursued.

As a first example, the semantics of the process $P = (\nu n)(\overline{a}\langle n\rangle \mid \overline{b}\langle n\rangle \mid n(x))$, containing two possible extruders for the action $n(x)$, can be represented by left-most prime event structure in Figure 3. When more than a single action use as subject the same bound name, each one of these actions must depend on one of the possible extruders. Then the causality of the process $(\nu n)(\overline{a}\langle n\rangle \mid \overline{b}\langle n\rangle \mid n(x).n(y))$, represented by the right-most event structure in Figure 3, shows that the two read actions might depend either on the same extruder or on two different extrusions.

Things get more complicate when dealing with the dynamic addition of new extruders by means of communications. In order to guarantee that there are distinct copies of any event with a bound subject that causally depend on different extruders, we have to consider the objective causalities generated by a communication. More precisely, when the variable x is substituted with a bound name n by effect of a synchronization, (i) any action with subject x appearing in the reading thread becomes an action that requires a previous scope extrusion, and (ii) the outputs with object x become new extruders for any action with subject n or x. To exemplify, consider the process $P' = (\nu n)(\overline{a}\langle n\rangle \mid \overline{b}\langle n\rangle \mid n(z)) \mid a(x).(\overline{x}\langle a\rangle \mid \overline{c}\langle x\rangle)$ that initially contains two extruders for n, and with the synchronization along the channel a evolves to $(\nu n)(\overline{b}\langle n\rangle \mid n(z) \mid \overline{n}\langle a\rangle \mid \overline{c}\langle n\rangle)$. Its causal semantics can be represented with the following prime event structure:



The read action $n(z)$ may depend on one of the two initial extruders $\overline{a}\langle n\rangle$ and $\overline{b}\langle n\rangle$, or on the new extruder $\overline{c}\langle n\rangle$ that is generated by the first communication. Accordingly, three different copies of the event $n(z)$ appear over each of the three extruders. On the other hand, the output action on the bound name n is generated by the substitution entailed by the communication along the channel a, hence any copy of that action keeps a (structural) dependence on the corresponding τ event. Moreover, since it is an action with bound subject, there must be a copy of it for each of the remaining extruders of n, that is $\overline{b}\langle n\rangle$ and $\overline{c}\langle n\rangle$. To enhance readability, the event structure resulting from the execution of the communication along the channel a is the leftmost e.s. in Figure 4.

So far so good, in particular it seems possible to let the causal relation of prime event structures encode both structural and objective causality of π-processes. However, this is not the case. Consider the process $P = (\nu n)(\overline{a}\langle n\rangle.\overline{b}\langle n\rangle.n(z))$ of Example 18.

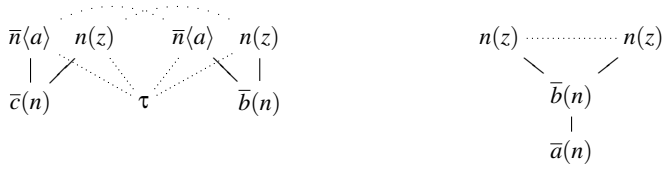


Fig. 4.

If we just duplicate the event $n(z)$ to distinguish the fact that it might depend on an extrusion along a or along b, we obtain the rightmost structure in Figure 4, that we denote $\mathcal{E}_p$. In particular, even if the two copies intends to represent two different objective causalities, nothing distinguishes them since since they both structurally depend on both outputs. This is a problem when we compose the process P in parallel with, e.g., $Q = a(x).\overline{c}\langle x\rangle \mid b(y).\overline{d}\langle y\rangle$. After the two synchronizations we would like to obtain two copies of the read actions on n that depend on the two different remaining extruders $\overline{c}\langle n\rangle$ and $\overline{d}\langle n\rangle$. However, in order to obtain such an event structure as the parallel composition of the semantics of P and the semantics of Q we must be able to record somehow the different objective causality that distinguishes the two copies of $n(z)$ in the semantics of P.

The technical solution would be to enrich the event labels so that the label of an event e also records the identity of the extruder events that e (objectively) causally depends on. A precise account of this approach is technically wired and intractable, so that the intuition on the semantics gets lost. Moreover, we think that this final example sheds light on the fact that structural and objective causality of π-processes cannot be expressed by the sole causal relation of event structures. To conclude, at the price of losing the information about which extruder an event depends on, the approach we developed in the previous section brings a number of benefits: it is technically much simpler, it is operationally adequate, it gives a clearer account of the two forms of causality distinctive of π-processes.

7 Related Work

There are several causal models for the π-calculus, that use different techniques. There exist semantics in terms of labelled transition systems, where the causal relations between transitions are represented by "proofs" which allow to distinguish different occurrences of the same transition [BS98,DP99]. In [CS00], a more abstract approach is followed, which involves indexed transition systems. In [JJ95], a semantics of the π-calculus in terms of pomsets is given, following ideas from dataflow theory. The two papers [BG95,Eng96] present Petri nets semantics of the π-calculus. However, only [MP95] introduces a semantics that allow the parallel extrusion; this is in the framework of graph rewriting.

Previous work on an event structure semantics of the π-calculus are [VY10,CVY07] that study fragments of the calculus, while [BMM06] gives an unlabelled event structure semantics of the full calculus which only corresponds to the *reduction* semantics, hence which is not compositional. Glynn Winskel [Win05] proposes a framework for accounting for new name generation within event structures, but this framework has not yet been applied to the π-calculus.

The main application of our work would be the study of a labelled reversible semantics of the π-calculus that would extend the work of Danos and Krivine [DK04]. Phillips and Ulidowski [PU07] noted the strict correspondence between reversible transition systems and event structures. A first step in this direction is [LMS10], which proposes a reversible semantics of the π-calculus that only considers reductions. It would also be interesting to study which kind of configuration structures [vGP09] can naturally include our definition of permitted configuration.

References

[BG95] N. Busi and R. Gorrieri. A petri net semantics for pi-calculus. In *Proceedings of 6th CONCUR*, volume 962 of *LNCS*, pages 145–159. Springer, 1995.

[BMM06] R. Bruni, H. Melgratti, and U. Montanari. Event structure semantics for nominal calculi. In *Proc.of CONCUR*, volume 4137 of *LNCS*, pages 295–309. Springer, 2006.

[BS98] M. Boreale and D. Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Inf.*, 35(5):353–400, 1998.

[CS00] Gian Luca Cattani and Peter Sewell. Models for name-passing processes: Interleaving and causal. In *Proceedings of 15th LICS*, pages 322–332. IEEE, 2000.

[CVY07] S. Crafa, D. Varacca, and N. Yoshida. Compositional event structure semantics for the internal pi-calculus. In *Proc. of CONCUR*, volume 4703 of *LNCS*, pages 317–332. Springer, 2007.

[DDNM88] P. Degano, R. De Nicola, and U. Montanari. On the consistency of truly concurrent operational and denotational semantics. In *LICS*, pages 133–141. IEEE, 1988.

[DK04] V. Danos and J. Krivine. Reversible communicating systems. In *Proceedings of CONCUR 2004*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004.

[DP99] P. Degano and C. Priami. Non-interleaving semantics for mobile processes. *Theor. Comp. Sci.*, 216(1-2):237–270, 1999.

[Eng96] Joost Engelfriet. A multiset semantics for the pi-calculus with replication. *Theor. Comp. Sci.*, 153(1&2):65–94, 1996.

[JJ95] L. J. Jagadeesan and R. Jagadeesan. Causality and true concurrency: A data-flow analysis of the pi-calculus. In *AMAST*, volume 936 of *LNCS*, pages 277–291, 1995.

[LMS10] I. Lanese, C. A. Mezzina, and J. Stefani. Reversing higher-order pi. In *Proceedings of CONCUR 2010*, volume 6269 of *LNCS*, pages 478–493. Springer, 2010.

[MP95] U. Montanari and M. Pistore. Concurrent semantics for the pi-calculus. *Electr. Notes Theor. Comput. Sci.*, 1:411–429, 1995. Proceedings of MFPS '95.

[NPW81] Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theor. Comp. Sci.*, 13(1):85–108, 1981.

[PU07] I. Phillips and I. Ulidowski. Reversibility and models for concurrency. *Electr. Notes Theor. Comput. Sci.*, 192(1):93–108, 2007. Proceedings of SOS 2007.

[vGP09] Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures, event structures and petri nets. *Theor. Comput. Sci.*, 410(41):4111–4159, 2009.

[VY10] D. Varacca and N. Yoshida. Typed event structures and the linear pi-calculus. *Theor. Comput. Sci.*, 411(19):1949–1973, 2010.

[Win82] G. Winskel. Event structure semantics for CCS and related languages. In *Proceedings of 9th ICALP*, volume 140 of *LNCS*, pages 561–576. Springer, 1982.

[Win87] G. Winskel. Event structures. In *Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.

[Win05] G. Winskel. Name generation and linearity. In *LICS*, pages 301–310. IEEE, 2005.

[WN95] G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of logic in Computer Science*, volume 4. Clarendon Press, 1995.