Concurrent and Located Synchronizations in π-Calculus*

Ivan Lanese

Computer Science Department, University of Bologna, Italy
lanese@cs.unibo.it

Abstract. We present two novel semantics for π-calculus. The first allows one to observe on which channel a synchronization is performed, while the second allows concurrent actions, provided that they do not compete for resources. We present both a reduction and a labeled semantics, and show that they induce the same behavioral equivalence. As our main result we show that bisimilarity is a congruence for the concurrent semantics. This important property fails for the standard semantics.

1 Introduction

Recent years have seen a strong effort in the field of process calculi, trying to find the best suited primitives and tools for describing different properties of concurrent interacting systems. One of the most successful among these calculi is the π-calculus [8], which allows one to model mobility, an interesting feature of modern systems, in a natural way. Different extensions have been considered to describe, for instance, concurrency aspects and locations [11,9,14,3].

Concurrency is usually obtained via mappings to models which are equipped with concepts of causality and independence, such as graph transformation systems [9], Petri nets [3] or event structures [14]. This allows one to reason about concurrency issues, but this makes harder or even prevents the use of standard process calculi tools based on labeled transition systems (LTSs). We examine which concurrency aspects can be modeled in process calculi using a standard LTS. Clearly, labels of this LTS will be richer than standard labels. In particular, we allow the execution of many actions inside the same transition, and the label will contain all of them. While some actions do not interfere with each other, others may compete for resources. In real concurrent systems, in fact, actions usually require exclusive access to the communication medium. As a very simple example, you cannot telephone if the line is busy: you need to use another line. This is modeled in π-calculus by requiring concurrent actions to be performed on different channels. This can be done easily for inputs and outputs, but not for synchronizations. In fact, in the standard π-calculus semantics, the label of any complete synchronization is τ, and this does not contain any information on the used channel. This information is necessary for our semantics. Thus, to have a gradual presentation, first we analyze the effects of adding the location of

* Research supported by the Project FET-GC II IST 16004 Sensoria.

Jan van Leeuwen et al. (Eds.): SOFSEM 2007, LNCS 4362, pp. 388–399, 2007.

the synchronization to the label in the standard interleaving scenario, and then we move to the concurrent one. The interleaving case is a necessary step, but it may be useful also by itself. In fact, different channels may not be equivalent, for instance since they may be under different accounting policies.

We analyze the properties of the interleaving and the concurrent semantics both at the level of LTS and of the induced behavioral equivalence. In particular, in both the cases we consider a reduction and a labeled semantics and we show that they induce the same bisimilarity relation. We concentrate on the strong semantics, and give some insights on how the results can be extended to the weak case. An important property of the concurrent semantics is compositionality: the induced bisimilarity is a congruence w.r.t. the operators of process composition, while this is not the case for the standard semantics. This property allows one to compute the behavior of large complex systems from the behavior of their components, making analysis techniques scalable.

Structure of the paper. In Section 2 we recall the standard (early) semantics of π-calculus. Section 3 introduces locations in the interleaving setting, while Section 4 moves to the concurrent one. Section 5 describes some comparisons with similar approaches, while Section 6 outlines the weak semantics. Finally, Section 7 presents some conclusions and traces for future work. A version of the paper with full proofs is available at [4].

2 Background

In this section we present the syntax and the standard (early) semantics of π-calculus (we consider only the monadic π-calculus, but the extension to the polyadic case is straightforward). See, e.g., [13] for a more detailed presentation.

Processes, in π-calculus, communicate by exchanging channel names, using channels themselves as communication medium. Therefore we assume a countable set of channel names ranged over by $a, b, x, \ldots$.

Definition 1 (Syntax)

$$P ::= \overline{a}b.P_1 \mid a(x).P_1 \mid P_1|P_2 \mid P_1 + P_2 \mid \nu a\ P_1 \mid !P_1 \mid 0$$

In the above definition $\overline{a}b.P_1$ is a process that outputs the name b on channel a, while $a(x).P_1$ accepts an input on channel a, and, after receiving b, it behaves as $P_1\{b/x\}$. Both $\overline{a}b$ and $a(x)$ are called prefixes. Also, $P_1|P_2$ is the parallel composition of P_1 and P_2, $P_1 + P_2$ the process that can behave as P_1 or as P_2, $\nu a\ P_1$ is like process P_1, but the scope of channel a has been restricted to P_1, $!P_1$ stands for an unbounded number of copies of P_1 executing in parallel and 0 is the idle process. We restrict our attention to prefix-guarded processes, i.e., in $P_1 + P_2$ both P_1 and P_2 must be either prefixed processes or sums (or 0).

Name x is bound in $a(x).P_1$ and name a is bound in $\nu a\ P_1$. The functions $\mathrm{fn}(P)$, $\mathrm{bn}(P)$ and $\mathrm{n}(P)$ computing the sets of free names, bound names and all the names in process P respectively are defined as usual. We consider processes

Table 1. Standard reduction semantics

$$\text{react-S} \quad (a(x).P+M)|(\overline{a}b.Q+N) \to_S P\{b/x\}|Q$$

$$\text{par-S} \quad \frac{P \to_S P'}{P|Q \to_S P'|Q} \qquad \text{res-S} \quad \frac{P \to_S P'}{\nu a\ P \to_S \nu a\ P'}$$

$$\text{congr-S} \quad \frac{P_1 \equiv P_2 \to_S P_2' \equiv P_1'}{P_1 \to_S P_1'}$$

up to α-conversion of bound names, i.e., we always suppose that all the bound names are different and different from the free names. We will write a instead of $a(x)$ and $\overline{a}$ instead of $\overline{a}b$ if x and b are not important, and if π is a prefix we write π for $\pi.0$.

We first describe the allowed transitions, then the behavioral equivalence. We consider both the reduction semantics, analyzing the behavior of the system in isolation, and the labeled semantics, analyzing its interactions with the environment. In the following sections we will show how these semantics must be changed to handle located and concurrent synchronizations.

To simplify the presentation of the reduction semantics we exploit a structural congruence equating processes that we never want to distinguish.

Definition 2. *The* structural congruence $\equiv$ *is the least congruence satisfying the monoid laws for parallel composition and summation (with* 0 *as unit), the replication law* $P|!P \equiv P$ *and the laws for restriction* $\nu a\, \nu b\, P \equiv \nu b\, \nu a\, P$, $\nu a\, 0 \equiv 0$ *and* $\nu a(P_1|P_2) \equiv P_1|\nu a\, P_2$ *if* $a \notin \mathrm{fn}(P_1)$.

Definition 3 (Reduction semantics). *The* reduction semantics *of* π*-calculus is the set of unlabeled transitions generated by the rules in Table 1.*

The subscript S (for standard) is used to distinguish the standard semantics from the ones we will present later. Also, we will use uppercase letters for reduction semantics and lowercase ones for labeled semantics, thus standard labeled semantics is identified by subscript s.

Definition 4 (Labeled semantics). *The* labeled semantics *is the LTS defined in Table 2. We have as labels input* ab*, output* $\overline{a}b$*, bound output* $\overline{a}(b)$ *(where* b *is bound) and internal action* τ*. We use* α *as metavariable to denote labels.*

The subject $\mathrm{subj}(\alpha)$ and the object $\mathrm{obj}(\alpha)$ of an action α are a and b respectively if α is ab, $\overline{a}b$ or $\overline{a}(b)$, while $\mathrm{subj}(\tau)$ and $\mathrm{obj}(\tau)$ are both undefined.

We now define the behavioral equivalence for our processes. The same definition will be applied also to the LTSs that we will define in the following sections. Subscripts will always clarify which underlying LTS is used.

Definition 5 (Bisimilarity). *Let* t *be a LTS. A* bisimulation *is a relation* $\mathcal{R}$ *such that* $P\,\mathcal{R}\,Q$ *implies:*

Table 2. Standard labeled semantics (rules with * have also a symmetric counterpart)

$$\text{out-s}\quad \overline{a}b.P \xrightarrow{\overline{a}b}_s P \qquad\qquad \text{inp-s}\quad a(x).P \xrightarrow{ab}_s P\{b/x\}$$

$$\text{sum-s}^*\ \frac{P \xrightarrow{\alpha}_s P'}{P+Q \xrightarrow{\alpha}_s P'} \qquad \text{par-s}^*\ \frac{P \xrightarrow{\alpha}_s P' \qquad \mathrm{bn}(\alpha)\cap\mathrm{fn}(Q)=\emptyset}{P|Q \xrightarrow{\alpha}_s P'|Q}$$

$$\text{com-s}^*\ \frac{P \xrightarrow{ab}_s P' \qquad Q \xrightarrow{\overline{a}b}_s Q'}{P|Q \xrightarrow{\tau}_s P'|Q'} \quad \text{close-s}^*\ \frac{P \xrightarrow{ab}_s P' \qquad Q \xrightarrow{\overline{a}(b)}_s Q' \qquad b \notin \mathrm{fn}(P)}{P|Q \xrightarrow{\tau}_s \nu b(P'|Q')}$$

$$\text{res-s}\ \frac{P \xrightarrow{\alpha}_s P' \qquad a \notin \mathrm{n}(\alpha)}{\nu a\, P \xrightarrow{\alpha}_s \nu a\, P'} \qquad \text{open-s}\ \frac{P \xrightarrow{\overline{a}b}_s P' \qquad a \neq b}{\nu b\, P \xrightarrow{\overline{a}(b)}_s P'}$$

$$\text{rep-s}\ \frac{P|!P \xrightarrow{\alpha}_s P'}{!P \xrightarrow{\alpha}_s P'}$$

– $P \xrightarrow{\alpha}_t P'$ *with* $\mathrm{bn}(\alpha)\cap\mathrm{fn}(Q)=\emptyset$ *implies* $Q \xrightarrow{\alpha}_t Q' \wedge P' \mathcal{R}\, Q'$,
– *vice versa.*

A full bisimulation is a substitution closed bisimulation. We denote with $\approx_t$ *(resp.* $\sim_t$*) the maximal bisimulation (resp. full bisimulation), called bisimilarity (resp. full bisimilarity).*

3 Observing Locations in the Interleaving Setting

In this section we present both a reduction semantics and a labeled semantics for π-calculus, based on the idea that synchronizations performed on different channels must be distinguished. We also show that they produce the same behavioral equivalence. Those semantics will be identified by subscripts L and l respectively.

The information about localities must be added to labels, thus there are (simple) labels (denoted by S) also in the reduction semantics. However, when speaking about labeled semantics, we refer to the other style of semantics. Reductions are labeled by sets of channel names containing the free names on which a synchronization is performed. Thus we may have a singleton if the reduction is performed by synchronizing on a free name, and the empty set otherwise. Synchronization on local channels cannot be observed, since the restriction operator completely hides the channel. This also follows the intuition that any effect of the channel usage, included for instance its accounting, must be performed before restricting it. See [2] for an example on introducing accounting on a channel.

Definition 6 (Located interleaving reduction semantics). *The* located interleaving reduction semantics *of* π*-calculus is the LTS generated by the rules in Table 3.*

This semantics strictly follows the structure of standard reduction semantics (Table 1). Actually, the only difference is the introduction of labels.

Table 3. Located interleaving reduction semantics

$$\text{react-L} \quad (a(x).P+M)|(\overline{a}b.Q+N) \xrightarrow{\{a\}}_L P\{b/x\}|Q$$

$$\text{par-L} \quad \frac{P \xrightarrow{S}_L P'}{P|Q \xrightarrow{S}_L P'|Q} \qquad \text{res-L} \quad \frac{P \xrightarrow{S}_L P'}{\nu a\ P \xrightarrow{S\setminus\{a\}}_L \nu a\ P'}$$

$$\text{congr-L} \quad \frac{P_1 \equiv P_2 \xrightarrow{S}_L P_2' \equiv P_1'}{P_1 \xrightarrow{S}_L P_1'}$$

We now present the labeled semantics, by extending the one in Table 2. Technically, the main difference is that we have different labels denoting a complete synchronization instead of just τ. More precisely, we denote a synchronization at a free name a with $a\tau$, while τ is used to denote a synchronization on a restricted channel. The located semantics is obtained by substituting rules com-s and close-s (and their symmetric) with:

$$\text{com-l} \quad \frac{P \xrightarrow{ab}_l P' \qquad Q \xrightarrow{\overline{a}b}_l Q'}{P|Q \xrightarrow{a\tau}_l P'|Q'}$$

$$\text{close-l} \quad \frac{P \xrightarrow{ab}_l P' \qquad Q \xrightarrow{\overline{a}(b)}_l Q' \qquad b \notin \text{fn}(P)}{P|Q \xrightarrow{a\tau}_l \nu b\ (P'|Q')}$$

and adding the new rule tau-l:

$$\text{tau-l} \quad \frac{P \xrightarrow{a\tau}_l P'}{\nu a\ P \xrightarrow{\tau}_l \nu a\ P'}$$

We extend the definition of $\text{subj}(\alpha)$ and $\text{obj}(\alpha)$ by defining $\text{subj}(a\tau) = a$, while $\text{obj}(a\tau)$ is undefined.

The next lemma characterizes the correspondence between standard labeled semantics and located labeled semantics.

Lemma 1 (Operational correspondence). *$P \xrightarrow{\alpha}_l P'$ iff:*

- *either $\alpha \in \{ab, \overline{a}b, \overline{a}(b), \tau\}$ and $P \xrightarrow{\alpha}_s P'$,*
- *or $\alpha = a\tau$ for some $a \in \text{fn}(P)$ and $P \xrightarrow{\tau}_s P'$.*

Note that the states of the located and of the standard LTS coincide, but located labels carry more information. Thus located (full) bisimilarity implies the standard one.

Corollary 1. *$P \approx_l P' \Rightarrow P \approx_s P'$ and $P \sim_l P' \Rightarrow P \sim_s P'$.*

The converse of the previous corollary does not hold.

Counterexample 1 (Located vs standard (full) bisimilarity)
$\nu b\,(\overline{a}+\overline{b})|(a+b) \approx_s \overline{a}|a$ but not $\nu b\,(\overline{a}+\overline{b})|(a+b) \approx_l \overline{a}|a$.

The only difference between the two processes is that the left one can also perform a τ action on the hidden channel b. In the standard semantics this is indistinguishable w.r.t. the synchronization on a, while they are different under the located semantics. The same counterexample holds also for full bisimilarity.

We now analyze the relationships between the reduction and the labeled semantics. First of all we show that the reduction semantics fully captures all the transitions of the labeled semantics that do not require interactions with the environment. We denote with $S\tau$ the label $a\tau$ if $S = \{a\}$ and τ if $S = \emptyset$.

Theorem 1. *$P \xrightarrow{S}_L P'$ iff $P \xrightarrow{S\tau}_l P''$ with $P'' \equiv P'$.*

More interestingly, two processes are bisimilar in any context under the reduction semantics iff they are full bisimilar according to the labeled one.

Definition 7 (Context). *A context $C[\bullet]$ is obtained when a $\bullet$ replaces an occurrence of 0 in a process. We denote as $C[P]$ the process obtained by replacing $\bullet$ with P in $C[\bullet]$, if it is well-formed.*

Theorem 2. *$P \sim_l Q$ iff $C[P] \approx_L C[Q]$ for each context $C[\bullet]$.*

This result proves the complete correspondence between the two semantics.

4 Concurrent Synchronizations

We extend the located semantics presented in the previous section to allow the contemporary execution of many actions, provided that they are performed on different channels. This is justified by the observation that in a system with real parallelism, such as distributed systems, different components can interact at the same time, provided that they do not compete for resources. However, the system is not fully synchronous, thus actions can occur also in isolation, or, in other terms, some components may stay idle during a transition. More parallel scenarios, where the communication medium can be shared by different actions, will be subject of future work. We use subscripts C and c to identify the reduction and the labeled semantics respectively. We start by presenting the reduction semantics.

Definition 8 (Concurrent located reduction semantics). *The* concurrent located reduction semantics *of π-calculus is the LTS generated by the rules in Table 3 and by the rule:*

$$\textit{comp-C} \quad \frac{P \xrightarrow{S_1}_C P' \qquad Q \xrightarrow{S_2}_C Q' \qquad S_1 \cap S_2 = \emptyset}{P|Q \xrightarrow{S_1 \cup S_2}_C P'|Q'}$$

Here labels are used to check that concurrent reductions use different resources. The added rule allows in fact parallel processes to concurrently reduce, by synchronizing on different channels. For instance, $\overline{a}b|a(x).\overline{x}c|\overline{c}|c \xrightarrow{\{a,c\}}_C \overline{b}c$.

The following theorem shows the relation between the concurrent and the interleaving semantics.

Theorem 3. *$P \xrightarrow{S}_C P'$ implies $P = P_1 \xrightarrow{S_1}_L P_2 \xrightarrow{S_2}_L \dots \xrightarrow{S_n}_L P_{n+1} = P'$ with $\bigcup_{i\in\{1,\dots,n\}} S_i = S$.*

We now consider the labeled semantics. Technically, labels are essentially multisets of located labels. Indeed, they are exactly that when there are no restricted names. Restricted names appear in the label when they are extruded, such as b in $\nu b\,\overline{a}b \xrightarrow{\overline{a}(b)}_l 0$. However many outputs may extrude the same name concurrently. Thus the set of extruded names must be attached to the whole label and not to single outputs.

Thus we use labels of the form $(Y)act$ where Y is the set of extruded names and act is a multiset of basic actions α of the form ab, $\overline{a}b$, $a\tau$ or τ. We use μ as metavariable for those labels, and we write $\alpha \in \mu$ if either α has the form ab, $\overline{a}b$ (with $b \notin Y$), $a\tau$ or τ and it belongs to act, or if $\alpha = \overline{a}(b)$, $\overline{a}b \in act$ and $b \in Y$. We use $[\alpha_1, \alpha_2, \dots, \alpha_n]$ to denote a multiset containing the elements $\alpha_1, \alpha_2, \dots, \alpha_n$, and we use the operators $\cup$, $\subseteq$, $\setminus$, ... on multisets with the obvious meaning. We extend the notation to deal with labels, where the operators are applied to both the multiset part and the set of extruded names (but, if a name does not occur in the multiset, then it is removed also from the set of extruded names). We call sequential label any label whose multiset part is a singleton, and sequential transition any transition with a sequential label.

We define $\mathrm{subj}(\mu) = \bigcup_{\alpha\in\mu} \mathrm{subj}(\alpha)$ and similarly $\mathrm{obj}(\mu) = \bigcup_{\alpha\in\mu} \mathrm{obj}(\alpha)$. Also $\mathrm{tau}(\mu)$ is the largest submultiset of μ containing only actions τ (non located).

A label $\mu = (Y)act$ is well-formed if $[\alpha_1, \alpha_2] \subseteq \mu$ implies $\mathrm{subj}(\alpha_1) \neq \mathrm{subj}(\alpha_2)$ (if both the actions have a subject) and $y \in Y$ implies $y \in \mathrm{obj}(\mu)$ and $y \notin \mathrm{subj}(\mu)$. We denote as $\mathrm{act}_a(\mu)$ the unique action $\alpha \in \mu$ such that $\mathrm{subj}(\alpha) = a$, if it exists.

In order to define the semantics we introduce two auxiliary operators to deal with labels: @ and $\setminus$, corresponding intuitively to label composition and label restriction. The label $\mu_1 @ \mu_2$ is defined only if, whenever $x \in \mathrm{subj}(\mu_1)$ and $x \in \mathrm{subj}(\mu_2)$, $\mathrm{act}_x(\mu_1)$ and $\mathrm{act}_x(\mu_2)$ are an input and an output (possibly bound) with equal subjects and objects. In that case $\mu_1 @ \mu_2 = (Y)act$ with:

$$
\begin{aligned}
act = {} & \mathrm{tau}(\mu_1) \cup \mathrm{tau}(\mu_2) \cup \\
& \bigcup_{a\in \mathrm{subj}(\mu_1)\cup\mathrm{subj}(\mu_2)} \begin{cases} [a\tau] & \text{if } a \in \mathrm{subj}(\mu_1) \cap \mathrm{subj}(\mu_2) \\ \mathrm{act}_a(\mu_i) & \text{if } a \in \mathrm{subj}(\mu_i) \setminus \mathrm{subj}(\mu_{3-i}), i \in \{1,2\} \end{cases}
\end{aligned}
$$

Also, $Y = (\mathrm{bn}(\mu_1) \cup \mathrm{bn}(\mu_2)) \cap \mathrm{obj}(act)$.

Similarly, $\mu \setminus a$ is defined only if all the occurrences of a in μ (if any) are as object of a free output or as subject of $a\tau$. In the last case $a\tau$ is replaced by τ. Other actions are preserved. If $a \in \mathrm{obj}(\mu)$, then a is added to Y, otherwise Y is unchanged.

We use νA as shortcut for $\nu a_1 \nu a_2 \dots \nu a_n$ where $A = \{a_1, a_2, \dots, a_n\}$.

Definition 9 (Concurrent located labeled semantics). *The* concurrent located labeled semantics *of π-calculus is the LTS defined in Table 4.*

Table 4. Concurrent located labeled semantics (rules with * have also a symmetric counterpart)

$$\text{out-c} \quad \overline{a}b.P \xrightarrow{[\overline{a}b]}_c P \qquad\qquad \text{inp-c} \quad a(x).P \xrightarrow{[ab]}_c P\{b/x\}$$

$$\text{sum-c}^* \quad \frac{P \xrightarrow{\mu}_c P'}{P+Q \xrightarrow{\mu}_c P'} \qquad\qquad \text{par-c}^* \quad \frac{P \xrightarrow{\mu}_c P' \qquad \mathrm{bn}(\mu)\cap\mathrm{fn}(Q)=\emptyset}{P|Q \xrightarrow{\mu}_c P'|Q}$$

$$\text{com-c}^* \quad \frac{P \xrightarrow{\mu_1}_c P' \qquad Q \xrightarrow{\mu_2}_c Q' \qquad \Phi}{P|Q \xrightarrow{\mu_1 \,@\, \mu_2}_c \nu Z\, P'|Q'} \qquad\qquad \text{res-c} \quad \frac{P \xrightarrow{\mu}_c P' \qquad \Phi'}{\nu a\, P \xrightarrow{\mu \setminus a}_c \nu A\, P'}$$

$$\text{rep-c} \quad \frac{P|!P \xrightarrow{\mu}_c P'}{!P \xrightarrow{\mu}_c P'}$$

where Φ requires $\mathrm{bn}(\mu_1)\cap\mathrm{fn}(Q) = \mathrm{bn}(\mu_2)\cap\mathrm{fn}(P) = \mathrm{bn}(\mu_1)\cap\mathrm{bn}(\mu_2) = \emptyset$ and defines $Z = (\mathrm{bn}(\mu_1)\cup\mathrm{bn}(\mu_2)) \setminus \mathrm{bn}(\mu_1 \,@\, \mu_2)$ and Φ' defines $A = \{a\}$ if $a \notin \mathrm{obj}(\mu)$ and $A = \emptyset$ otherwise.

The following theorem shows that the concurrent LTS includes the interleaving one. Moreover, when moving to the concurrent framework, no sequential transitions are added.

Theorem 4. *$P \xrightarrow{\alpha}_l P'$ iff*

- *$\alpha \neq \overline{a}(b)$ and $P \xrightarrow{[\alpha]}_c P'$;*
- *$\alpha = \overline{a}(b)$ and $P \xrightarrow{(b)[\overline{a}b]}_c P'$.*

As an obvious consequence the concurrent bisimilarity implies the located (and the standard) one.

Corollary 2. *$P \approx_c P' \Rightarrow P \approx_l P' \Rightarrow P \approx_s P'$.*

The following counterexample shows that the concurrent bisimilarity is strictly finer.

Counterexample 2 (Concurrent vs located bisimilarity)
$a|\overline{b} \approx_l a.\overline{b} + \overline{b}.a$ but not $a|\overline{b} \approx_c a.\overline{b} + \overline{b}.a$.

The two processes are bisimilar under the located semantics, but not under the concurrent one where $a|\overline{b} \xrightarrow{[a,\overline{b}]}_c 0$, a transition that cannot be matched by $a.\overline{b} + \overline{b}.a$. This shows that the concurrent semantics highlights the degree of parallelism of a process, distinguishing between concurrency and nondeterminism. This is the same counterexample used to prove that $\approx_s$ is not a congruence, since the two terms have different transitions when placed in a context that merges a and b: the first one can perform a τ action while the second one cannot. This counterexample essentially exploits the fact that the expansion law is no longer valid. However some instances of the expansion law hold, for instance when actions are on the same channel: $\overline{a}x|\overline{a}y \approx_c \overline{a}x.\overline{a}y + \overline{a}y.\overline{a}x$. Also, the ability to perform actions in parallel includes the ability to perform the same actions sequentially, thus $c.(a|b) \approx_c c.(a|b) + c.a.b$.

The above counterexample suggests that bisimilarity may be a congruence. This is indeed the case, as proved by the following theorem.

Theorem 5. *$\approx_c$ is a congruence w.r.t. all the operators in the calculus and w.r.t. substitutions.*

While referring to [4] for the whole proof, we want to highlight here some important points. First of all this theorem suggests that observing concurrency aspects is important to have good compositionality properties. This happens also in some similar cases (see [6]). Interestingly, adding a smaller amount of concurrency is enough to get this property, in fact it is enough to allow the concurrent execution of one input and one output on different channels. This alone would yield, however, a semantics that lacks, in our opinion, a clear intuitive meaning.

Building the concurrent semantics on top of the located semantics is fundamental for the congruence result. In fact, consider a concurrent semantics with only normal τ actions. Then the two terms $\nu b\,(a+b)|(\overline{a}+\overline{b})$ and $a|\overline{a}$ are bisimilar, but when they are inserted into context $a|\bullet$ the first one can perform $[a,\tau]$ going to 0 while the second one cannot. Notice that closure under substitutions implies $\approx_c = \sim_c$.

We now show that a concurrent transition can always be decomposed in a computation including only sequential transitions, thus generalizing Theorem 3.

Given a label μ and a sequential label α we define the operation $\alpha\,;\mu$ only if $\mathrm{bn}(\mu) \cap \mathrm{n}(\alpha) = \emptyset$ and the union of the two action parts is well-formed. In that case $\alpha\,;\mu$ is computed by making the union both on the multisets of actions and on the sets of extruded names.

Theorem 6. *If $P \xrightarrow{\alpha\,;\,\mu}_c P'$ then $P \xrightarrow{\alpha}_c P'' \xrightarrow{\mu}_c P'$.*

The recursive application of the theorem allows one to decompose a concurrent transition in a sequential computation. In fact, any non sequential label can be written as $\alpha;\mu$ for suitable α and μ.

Results similar to those in Theorem 1 and in Theorem 2 can be proved also for the concurrent scenario. However there is a little mismatch between the labeled and the reduction semantics we have presented, which are the most direct generalizations of the interleaving ones. The labeled semantics distinguishes between $[\tau]$ and $[\tau,\tau]$, while in the reduction one they both correspond to $\emptyset$. One can either add the missing information to the reduction labels, or remove it from the labeled setting. We analyze here the second case, but the first one is analogous. We use *nc* as subscript for this semantics (modifying the rules is trivial: actually it is enough to modify the operator of label restriction).

We extend the notation $S\tau$, denoting by it the multiset containing one action $a\tau$ for each $a \in S$.

Theorem 7. *$P \xrightarrow{S}_C P'$ iff $P \xrightarrow{S\tau}_{nc} P''$ with $P'' \equiv P'$.*

Theorem 8. *$P \approx_{nc} Q$ iff $C[P] \approx_C C[Q]$ for each context $C[\bullet]$.*

5 Related Work

Many semantics for π-calculus have been proposed in the literature, focusing on different aspects. We present a comparison with the ones more related to ours.

First of all, we take the inspiration for this work from a concurrent semantics for Fusion Calculus [10] derived in [7] using a mapping from Fusion Calculus into a graph transformation framework called SHR [5]. The intrinsic concurrent nature of SHR and the fact that actions there are naturally located on nodes make the main semantic aspects discussed in this paper emerge spontaneously. The semantics presented in [7] however preserved many of the particularities of SHR synchronization, such as the fact that extrusions are not observed, and processes are always allowed to perform idle transitions to themselves. Because of the first difference, processes $\nu x\,\overline{u}x$ and $\nu x\,\overline{u}x + \overline{u}z$ were bisimilar, while they are not even bisimilar with the standard semantics of π-calculus. On the other side, idle transitions allowed the observation of the free names of a process, thus $\nu x\,\overline{x}u$ and 0 were not bisimilar. Furthermore the semantics in [7] is derived via a mapping from an LTS which is quite different w.r.t standard process calculi LTSs, while our work presents a similar semantics in a direct and standard way.

Some related semantics for π-calculus are described below.

Net semantics [3]: this semantics is obtained via a mapping into Petri nets, and is quite related to ours. The main differences are that actions can use the same channel concurrently, thus $a.a$ and $a|a$ are distinguished, but two outputs cannot extrude the same name at the same time, thus $\nu y\,\overline{x}y.\overline{z}y + \overline{z}y.\overline{x}y$ and $\nu y\,\overline{x}y|\overline{z}y$ are equivalent.

Open bisimilarity [12]: open bisimilarity instantiates processes throughout the bisimulation game, but it uses distinctions to keep track of which names can never be merged. Open bisimilarity is less distinguishing than concurrent bisimilarity. The inclusion follows easily from the closure under arbitrary substitutions of concurrent bisimilarity. The inclusion is strict since $a|b$ and $a.b + b.a$ are open bisimilar but not concurrent bisimilar. Notice that open bisimilarity is a congruence, but it has no direct coinductive characterization.

Causal bisimilarity [1]: this semantics traces the causal dependencies among actions. It is not comparable with the concurrent semantics, since $\nu b\,(a+b)|(\overline{a}+\overline{b})$ and $a|\overline{a}$ are causally bisimilar (there are no dependencies) but not concurrent bisimilar. Conversely $a|a$ and $a.a$ are concurrent bisimilar but not causally bisimilar.

If we add located τ actions to causal semantics we get a bisimilarity finer than the concurrent one. The inclusion follows since if two actions are independent, then they can be executed concurrently. Thus from the interleaving transitions and the causal dependencies one can compute the concurrent transitions. The inclusion is strict since $!a$ and $!a.a$ are causally different (in the second one there are causal dependencies between different occurrences of a, while in the first one there are not), but concurrent bisimilar.

Similar statements can be made for the (mixed-order) concurrent semantics in [9], which has a causal flavor.

Located bisimilarity [11]: in this bisimilarity a location is associated to each sequential process, thus actions performed by different sequential processes are distinguished. This concept of localities is completely different from ours, and even if it tracks sequential processes this bisimilarity is not comparable with the concurrent one. In fact, $\nu b\ a.b.c|e.\overline{b}$ and $\nu b\ a.b|e.\overline{b}.c$ are concurrent bisimilar, but they are not located bisimilar since in the first one c is executed in the same component of a while in the second one it is not. On the other hand: $\nu b \nu c\ b|\overline{b}|c|\overline{c}$ and $\nu b \nu c\ b|\overline{b}.(c|\overline{c})$ are located bisimilar since τ actions do not exhibit locations, while they are not concurrent bisimilar since in the first one the two τ actions can be executed in parallel, while in the second one they cannot.

6 Weak Semantics

In this section we outline the main features of the weak bisimilarities based on the labeled semantics we have introduced in this paper. Usually weak bisimilarity (see [13] for the precise definition) abstracts from internal activities, i.e. from τ actions. However, in our setting, we have two kinds of τ actions: $a\tau$ performed on free name a and τ performed on a hidden name. While one must surely abstract from the latter ones, abstracting also located synchronizations may lose too much information. We call semiweak bisimilarity the one that abstracts only from τ (or $\emptyset$ in the reduction semantics), and weak the one that abstracts also from $a\tau$ (or all the labels in the reduction semantics). Semiweak bisimilarity is midway between the strong and the weak semantics.

If we consider semiweak bisimilarity most of the results shown in the previous sections are still valid. The only notable difference is that the semiweak semantics is not image finite (i.e., a process may have an infinite number of one step derivatives), even up to bisimilarity, but theorems 2 and 8 can be proved only for processes that are image finite up to bisimilarity. However, the same hypothesis is required to prove the corresponding property of the standard weak semantics.

If we consider weak bisimilarity instead (based on the labeled semantics), the located and the standard bisimilarities collapse. The concurrent bisimilarity is still strictly finer (Counterexample 2 is still valid) than the standard one, but there is no simple relation with a reduction semantics. In fact, in the reduction semantics labels should be completely abstracted away, thus one must have some other way to observe process behavior. The usual approach of using barbs [13], that is observing the capabilities to produce inputs or outputs, is not sufficient. For instance, $a.b + b.a$ and $a|b$ are barbed bisimilar in the concurrent scenario (both of them can react when put in a context containing either $\overline{a}$ or $\overline{b}$ or both).

7 Conclusions and Future Work

We have presented two semantics for π-calculus, highlighting important information about which channels are used by a synchronization and which actions can be executed concurrently. We have analyzed the semantics both at the level of LTS and of induced behavioral equivalence. As our main result we have shown that bisimilarity is a congruence for the concurrent located semantics, and this guarantees compositionality. Note that all the shown results hold also for CCS, since mobility is not exploited in the used constructions.

As future work we plan to apply the same ideas to other calculi. In particular, preliminary analysis show that similar results can be obtained for Fusion Calculus [10], but more care is required to deal with fusions. Also, we want to study the semantic effect of allowing concurrent actions on the same channel. Preliminary results show that this has a strong impact, for instance the direct generalization of Theorem 8 fails.

Acknowledgments. The author would like to strongly acknowledge Davide Sangiorgi for many useful discussions and comments and Ugo Montanari for some early discussions.

References

1. Boreale, M. and Sangiorgi, D.: Some Congruence Properties of pi-Calculus Bisimilarities. Theoret. Comput. Sci. **198** 1-2 (1998) 159–176
2. Bruni, R. and Lanese, I.: PRISMA: A Mobile Calculus with Parametric Synchronization. In Proc. of TGC'06, Springer, LNCS (2006) to appear
3. Busi, N. and Gorrieri, R.: A Petri Net Semantics for pi-Calculus. In Proc. of CONCUR'95, Springer, LNCS **962** (1995) 145–159
4. Concurrent and Located Synchronizations in π-Calculus, extended version. http://www.cs.unibo.it/~lanese/publications/fulltext/sofsem2007ext.pdf.gz.
5. Ferrari, G.L., Montanari, U., and Tuosto, E.: A LTS Semantics of Ambients via Graph Synchronization with Mobility. In Proc. of ICTCS'01, Springer, LNCS **2202** (2001) 1–16
6. Lanese, I.: Synchronization Strategies for Global Computing Models. PhD Thesis, Computer Science Department, University of Pisa, Pisa, Italy (2006)
7. Lanese, I. and Montanari, U.: A Graphical Fusion Calculus. In Proceedings of the Workshop of the COMETA Project on Computational Metamodels, Elsevier Science, ENTCS **104** (2004) 199–215
8. Milner, R., Parrow, J., and Walker, J.: A Calculus of Mobile Processes, I and II. Inform. and Comput. **100** 1 (1992) 1–40, 41–77
9. Montanari, U. and Pistore, M.: Concurrent Semantics for the pi-Calculus. In Proc. of MFPS'95, Elsevier Science, ENTCS **1** (1995)
10. Parrow, J. and Victor, B.: The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In Proc. of LICS'98, IEEE Computer Society Press (1998) 176–185
11. Sangiorgi, D.: Locality and Interleaving Semantics in Calculi for Mobile Processes. Theoret. Comput. Sci. **155** 1 (1996) 39–83
12. Sangiorgi, D.: A Theory of Bisimulation for the pi-Calculus. Acta Inf. **33** 1 (1996) 69–97
13. Sangiorgi, D. and Walker, D.: Pi-Calculus: A Theory of Mobile Processes. Cambridge University Press (2001)
14. Varacca, D. and Yoshida, N.: Typed Event Structures and the pi-Calculus. In Proc. of MFPS'06, Elsevier Science, ENTCS **158** (2006) 373–397