COMPACT NET SEMANTICS FOR PROCESS ALGEBRAS

Marco Bernardo, Marina Ribaudo
Università di Torino, Dipartimento di Informatica
Corso Svizzera 185, 10149 Torino, Italy
bernardo, marina@di.unito.it

Nadia Busi
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
busi@cs.unibo.it

Abstract Process algebras and Petri nets are two well known formal methods. In the literature, several mappings from process algebras to Petri nets have been proposed to provide a truly concurrent semantic framework to concurrent programming languages. From an applicative point of view, such mappings facilitate the integration and the cross fertilization between the two formalisms, making it possible the development of a multiparadigm methodology for the modeling and analysis of concurrent systems since the early stages of their design. In this paper we define a new Petri net semantics for process algebras, we demonstrate that it improves the previous proposals with respect to the size of the resulting nets, and we illustrate the usefulness of the net semantics by showing how to reinterpret at the process algebra level results efficiently obtainable at the Petri net level.

1. INTRODUCTION

The complexity of modern computer and communication systems calls for the development of suitable description techniques for their modeling and analysis. The objective is to formally describe the relevant aspects of the systems under investigation since the early stages of their design, so that malfunctionings and inefficiency can be timely detected. Among such formal description techniques, in this paper we focus our attention on process algebras and Petri nets, which have been extensively investigated in the last four decades (see, e.g., [9, 12]) not only in isolation but also jointly. Initially, the purpose was that of providing a truly concurrent semantic framework to concurrent programming languages [8]. This was accomplished by means of semantic mappings from process algebras to Petri nets. In this paper we concentrate on operational (as opposed to denotational) semantic mappings, which have been defined according to two different approaches.

In the *location oriented approach*, the Petri net corresponding to a process term contains a place for every position of the sequential subterms w.r.t. the static operators. As a consequence, the resulting nets turn out to be 1-safe. All the information about the syntactical structure of terms is encoded within places so that the relationships among sequential terms are smoothly preserved. This is exploited to define net transitions by means of inductive rules similar to those for the interleaving semantics for the process algebra, as shown in [7, 10].

In the *label oriented approach*, instead, each net place corresponds to a sequential subterm independently of its position w.r.t. static operators. As a consequence, instances of the same sequential term occurring in different positions w.r.t. static operators can be represented by multiple tokens within the same place. This is possible because the syntactical structure of the process term w.r.t. static operators is no longer fully retained within net places associated with sequential subterms. Therefore, the correspondence between the inductive rules for generating net transitions and the inductive rules of the interleaving semantics for the process algebra no longer holds. In this approach net transitions are defined by axioms. For instance, in [4] five axioms are employed to derive net transitions, with the resulting nets including places with outgoing inhibitor arcs to model scope extrusion and unguarded choice. As another example, in [1] only one axiom is employed, the price to be paid being the introduction of places with outgoing inhibitor and contextual arcs in which information about the syntactical structure is kept. The advantage of the label oriented approach is that the resulting nets are not necessarily 1-safe, so they can be more compact than those obtained with the location oriented approach. Furthermore some terms, for which an infinite net is generated in the location oriented approach, give rise to a finite net in the label oriented approach. This fact has been exploited e.g. in [5], where the finiteness of the generated nets is used to show an expressiveness gap bewteen two different semantics for a process algebra based on Linda coordination primitives.

More recently, it has been recognized that the net semantics for process algebras have an applicative relevance, as they facilitate the integration of the two formalisms making it possible the exchanging of modeling capabilities and analysis techniques between them. This is especially important since the two formalisms are characterized by complementary strengths and weaknesses. On the one hand, process algebras offer a compositional linguistic support to system modeling together with notions of equivalence and preorder. On the other hand, Petri nets provide a truly concurrent framework equipped with analysis techniques that avoid the construction of the underlying state space.

Given the applicative relevance of the net semantics for process algebras, the label oriented approach seems to be the approach of choice due to the compactness of the resulting nets. The weakness of this approach is that additional places with outgoing inhibitor and contextual arcs are employed to handle static operators. Actually, in [1] it has been proved that these places and arcs can be removed a posteriori; this makes the nets simpler but requires an additional computational step. In this paper we further elaborate on the label oriented net semantics of [1] by avoiding the introduction of additional places with inhibitor and contextual arcs altogether. This will be achieved by suitably decorating actions within the sequential terms associated with net places in order to keep track of the occurrences of static operators, without falling in the excess of information of the location oriented approach.

This paper is organized as follows. In Sect. 2 and 3 we present process algebras and Petri nets, respectively. In Sect. 4 we define the new label oriented net semantics for our example process algebra and we prove its correctness by showing that the interleaving semantics for the process algebra can be retrieved from it. In Sect. 5 we discuss the finiteness and the compactness of the generated nets, the ability to detect some kinds of symmetry, and the reinterpretation at the process algebraic level of results efficiently obtainable at the net level. Finally, Sect. 6 concludes the paper with ongoing and future work. Due to lack of space, the reader is referred to [2] for proofs and additional examples.

2. PROCESS ALGEBRA

The process algebra we shall use throughout the paper is composed of a set *Act* of actions, ranged over by a, b, c, d and including the invisible action τ, and the standard operators. Let *Const* be a set of constants ranged over by A, B, C and let $Rel = \{\varphi : Act \longrightarrow Act \mid \varphi^{-1}(\tau) = \{\tau\}\}$ be a set of action relabeling functions preserving action observability.

Definition 1 The set $\mathcal{L}$ of process terms is generated by the grammar

$$E ::= \sum_{i \in I} a_i.E_i \mid E/L \mid E[\varphi] \mid E \parallel_S E \mid A$$

where I is a finite set of indices and $L, S \subseteq Act - \{\tau\}$. We denote by *sort*$(E)$ the set of actions occurring in E and by $\mathcal{G}$ the set of closed and guarded terms of $\mathcal{L}$. ■

Thus we have standard operators such as the guarded alternative composition operator (denoted by $\underline{0}$ whenever $I = \emptyset$), the hiding operator, the relabeling operator, the parallel composition operator with CSP like synchronization, and the constant invocation. We recall that the hiding, relabeling and parallel composition operators are said static operators.

$\sum_{i \in I} a_i.E_i$ expresses a nondeterministic choice among several alternatives. If action a_k, $k \in I$, is executed first, then the derivative term is E_k and the other summands are discarded. E/L behaves as term E except that each executed action is turned into τ whenever it belongs to L. $E[\varphi]$ behaves as term E except that each executed action a becomes $\varphi(a)$. $E_1 \,\|_S\, E_2$ asynchronously executes actions of E_1 or E_2 which do not belong to S and synchronously executes identical actions of E_1 and E_2 which belong to S. Finally, each constant A comes equipped with an equation of the form $A \stackrel{\Delta}{=} E$ defining its behavior.

$$\sum_{i \in I} a_i.E_i \xrightarrow{a_i} E_i \quad i \in I$$

$$\frac{E \xrightarrow{a} E'}{E/L \xrightarrow{a} E'/L} \ \text{if } a \notin L \qquad\qquad \frac{E \xrightarrow{a} E'}{E/L \xrightarrow{\tau} E'/L} \ \text{if } a \in L$$

$$\frac{E \xrightarrow{a} E'}{E[\varphi] \xrightarrow{\varphi(a)} E'[\varphi]} \qquad\qquad \frac{E \xrightarrow{a} E'}{A \xrightarrow{a} E'} \ \text{if } A \stackrel{\Delta}{=} E$$

$$\frac{E_1 \xrightarrow{a} E_1'}{E_1 \,\|_S\, E_2 \xrightarrow{a} E_1' \,\|_S\, E_2 \quad E_2 \,\|_S\, E_1 \xrightarrow{a} E_2 \,\|_S\, E_1'} \ \text{if } a \notin S$$

$$\frac{E_1 \xrightarrow{a} E_1' \quad E_2 \xrightarrow{a} E_2'}{E_1 \,\|_S\, E_2 \xrightarrow{a} E_1' \,\|_S\, E_2'} \ \text{if } a \in S$$

Table 1 Operational interleaving semantics for $\mathcal{G}$

The interleaving semantics for $\mathcal{G}$ is represented by a labeled transition system (LTS for short) where states are in correspondence with terms and labels are actions. Such a LTS is generated according to the inductive rules shown in Table 1.

Definition 2 The interleaving semantics of $E \in \mathcal{G}$ is the LTS $\mathcal{I}[\![E]\!] = (S_{E,\mathcal{I}}, Act, \longrightarrow_{E,\mathcal{I}}, E)$ where $S_{E,\mathcal{I}}$ is the least subset of $\mathcal{G}$ reachable from E and $\longrightarrow_{E,\mathcal{I}}$ is $\longrightarrow$ restricted to $S_{E,\mathcal{I}} \times Act \times S_{E,\mathcal{I}}$. ■

3. PETRI NETS

A Petri net [12] is a bipartite graph whose classes of nodes are called places (representing system conditions and resources) and transitions (representing system activities), respectively. Unlike LTSs, Petri nets support a distributed notion of state given by the marking of places with tokens, which is formalized through a multiset. We use m as metavariable for multisets, "$\{\!|$" and "$|\!\}$" as brackets for multisets, "$\_ \oplus \_$" ("$\_ \ominus \_$") to denote multiset union (difference), and $\mathcal{M}(S)$ ($\mathcal{P}(S)$) to denote the collection of multisets over (subsets of) set S.

Definition 3 A *Petri net* is a tuple (P, Act, T, M_0) where P is a set of places, $T \subseteq \mathcal{M}(P) \times Act \times \mathcal{M}(P)$ is a set of transitions, and $M_0 \in \mathcal{M}(P)$ is the initial marking. The set $\mathcal{M}(P)$ of possible markings will be ranged over by M. ■

In the graphical representation of a Petri net, places are drawn as circles containing black dots called tokens (which describe the current marking) while transitions are drawn as boxes with the appropriate labels. Each transition t can be written as $({}^\bullet t, a, t^\bullet)$ where ${}^\bullet t$ is the weighted preset of t (places where tokens are consumed) and $t^\bullet$ is the weighted postset of t (places where tokens are produced). Given a transition t, we draw an arc from each place in ${}^\bullet t$ to t as well as from t to each place in $t^\bullet$, where each arc is labeled with the multiplicity of the related place.

Definition 4 Let $N = (P, Act, T, M_0)$ be a Petri net.

- $t \in T$ is enabled at marking $M \in \mathcal{M}(P)$ if and only if ${}^\bullet t \subseteq M$. The firing of t produces marking $M' = (M \ominus {}^\bullet t) \oplus t^\bullet$, written $M\,[a\rangle\, M'$.
- The reachability graph (RG for short) of N is the LTS $\mathcal{RG}[\![N]\!] = (RS(M_0), Act, [\rangle, M_0)$ where $RS(M_0)$ is the least subset of $\mathcal{M}(P)$ reachable from M_0. ■

4. NET SEMANTICS

4.1. NET PLACES

We present in three steps our improved net semantics for $\mathcal{G}$. The first step consists of introducing a suitable set of places as well as defining a function which decomposes terms into their sequential components and maps them onto places.

Since sequential terms are represented by guarded alternative compositions, these are used to formalize places. In order to keep track of the static operators, we suitably decorate actions within net places thus avoiding the introduction of inhibitor and contextual arcs and the excess of information of the location oriented approach. Each action a becomes $a^{\theta,R,\sigma}$. Decoration $\theta \in \{\tau, \varepsilon\}$ is used to remember to hide certain actions. Decoration R is a set of conflicts employed to avoid clashes within the scope of a relabeling operator. Its presence is required by the fact that actions will be directly relabeled within net places. We define *Conf* as an infinite set of conflicts (ranged over by r) on which the operation $^- : \mathit{Conf} \longrightarrow \mathit{Conf}, \bar{\bar{r}} = r$ is defined. Finally, decoration σ is a string of combinators used to correctly handle synchronizations. We define *Comb* as an infinite set of combinators (ranged over by k) and we consider the set Comb^* of combinator strings (ranged over by σ), on which the following two operations are defined (i) $^- : \mathit{Comb} \longrightarrow \mathit{Comb}$, $\bar{\bar{k}} = k$ and (ii) $\odot : \mathit{Comb}^* \times \mathit{Comb}^* \,{-\!\!\!\ominus\!\!\!\rightarrow}\, \mathit{Comb}^*$, $\sigma k \odot \sigma \bar{k} = \sigma$. The latter is extended to multisets over Comb^* as follows:

$$\bigodot m = \begin{cases} \bigodot(\{\!|\, \sigma \,|\!\} \oplus m') & \text{if } \exists \sigma, k, m'. \, m = \{\!|\, \sigma k, \sigma \bar{k} \,|\!\} \oplus m' \\ m & \text{otherwise} \end{cases}$$

Definition 5 The set of places is defined by $\mathcal{V} = \{\sum_{i \in I} a_i^{\theta_i, R_i, \sigma_i}.E_i \mid \theta_i \in \{\tau, \varepsilon\} \wedge R_i \in \mathcal{P}(\mathit{Conf}) \wedge \sigma_i \in \mathit{Comb}^* \wedge E_i \in \mathcal{G}'\}$ where the summation is associative and commutative and $\mathcal{G}'$ is the set of closed and guarded terms defined on the same syntax as $\mathcal{G}$ with *Act* replaced by $\mathit{Act}' = \mathit{Act} \times \{\tau, \varepsilon\} \times \mathcal{P}(\mathit{Conf}) \times \mathit{Comb}^*$, (a, θ, R, σ) denoted $a^{\theta,R,\sigma}$, and $a^{\varepsilon,\emptyset,\varepsilon}$ denoted a. $\mathcal{V}$ will be ranged over by V while $\mathcal{M}(\mathcal{V})$ will be ranged over by Q. ■

Decorations will be introduced by the decomposition function through substitutions. In order to avoid undesired substitutions, we identify binders arising from the static operators. In E/L, all the actions in L are bound in E. In $E[\varphi]$, all the actions in $\mathit{Dom}(\varphi) = \{c \in \mathit{Act} \mid \varphi(c) \neq c\}$ are bound in E. For operational convenience, we rewrite the synchronization set S of $E_1 \,\|_S\, E_2$ as the function $s = \{(a, a) \mid a \in S\}$. In $E_1 \,\|_s\, E_2$, all the actions in $\pi_1(s) = \{a \mid \exists b. \, (a, b) \in s\}$ are bound in E_1 and E_2. When dealing with substitutions, we make explicit the actions occurring in the body E of each constant definition $A \stackrel{\Delta}{=} E$ by rewriting the definition itself as $A(a_1, \ldots, a_n) \stackrel{\Delta}{=} E$, where $\{a_1, \ldots, a_n\} = \mathit{sort}(E)$. The semantic rule for constants in Table 1 is accordingly modified.

Definition 6 Let us assume a total ordering $\preceq$ be defined over *Act*. The decomposition function $dec : \mathcal{G}' \longrightarrow \mathcal{M}(\mathcal{V})$ is defined by structural induction as follows (assuming $A(a_1, \ldots, a_n) \overset{\Delta}{=} E$):

$$
\begin{array}{l}
dec(\sum_{i\in I} a_i^{\theta_i,R_i,\sigma_i}.E_i) = \{\!|\ \sum_{i\in I} a_i^{\theta_i,R_i,\sigma_i}.E_i\ |\!\} \\
dec(E/L) = dec(E\{a^{\tau,\emptyset,\varepsilon}/a \mid a \in L\}) \\
dec(E[\varphi]) = dec(E\{d^{\theta,R\cup R_a,\sigma}/a \mid a \in sort(E) \wedge (a, d^{\theta,R,\sigma}) \in \varphi \wedge \\
\qquad\qquad\qquad\qquad R_a = \{r_{ca} \mid \varphi(c) = \varphi(a) \wedge c \prec a \wedge r_{ca} \text{ fresh}\} \cup \\
\qquad\qquad\qquad\qquad\quad \{\bar{r}_{ac} \mid \varphi(c) = \varphi(a) \wedge a \prec c \wedge r_{ac} \text{ fresh}\}\}) \\
dec(E_1 \parallel_s E_2) = dec(E_1\{d^{\theta,R,\sigma k_a}/a \mid (a, d^{\theta,R,\sigma}) \in s \wedge k_a \text{ fresh}\}) \oplus \\
\qquad\qquad\qquad\quad dec(E_2\{d^{\theta,R,\sigma\bar{k}_a}/a \mid (a, d^{\theta,R,\sigma}) \in s \wedge k_a \text{ fresh}\}) \\
dec(A(b_1^{\theta_1,R_1,\sigma_1}, \ldots, b_n^{\theta_n,R_n,\sigma_n})) = dec(E\{b_i^{\theta_i,R_i,\sigma_i}/a_i \mid 1 \le i \le n\})
\end{array}
$$

with the syntactical substitutions on $\mathcal{G}'$ defined by structural induction as follows:

$$
\begin{array}{l}
(\sum_{i\in I} a_i^{\theta_i,R_i,\sigma_i}.E_i)\{b^{\theta,R,\sigma}/a\} = \sum_{i\in I} c_i^{\theta'_i,R'_i,\sigma'_i}.E_i\{b^{\theta,R,\sigma}/a\} \\
E/L\{b^{\theta,R,\sigma}/a\} = \begin{cases} E\{b^{\theta,R,\sigma}/a\}/L & \text{if } a \notin L \\ E/L & \text{if } a \in L \end{cases} \\
E[\varphi]\{b^{\theta,R,\sigma}/a\} = \begin{cases} E\{b^{\theta,R,\sigma}/a\}[\varphi'] & \text{if } a \notin Dom(\varphi) \\ E[\varphi'] & \text{if } a \in Dom(\varphi) \end{cases} \\
(E_1 \parallel_s E_2)\{b^{\theta,R,\sigma}/a\} = \begin{cases} E_1\{b^{\theta,R,\sigma}/a\} \parallel_s E_2\{b^{\theta,R,\sigma}/a\} & \text{if } a \notin \pi_1(s) \\ E_1 \parallel_{s'} E_2 & \text{if } a \in \pi_1(s) \end{cases} \\
A(a_1^{\theta_1,R_1,\sigma_1}, \ldots, a_n^{\theta_n,R_n,\sigma_n})\{b^{\theta,R,\sigma}/a\} = A(c_1^{\theta'_1,R'_1,\sigma'_1}, \ldots, c_n^{\theta'_n,R'_n,\sigma'_n})
\end{array}
$$

where:

$$
\begin{array}{rcl}
\varphi' & = & \{(d_i, c_i^{\theta'_i,R'_i,\sigma'_i}) \mid \exists a_i^{\theta_i,R_i,\sigma_i}.\,(d_i, a_i^{\theta_i,R_i,\sigma_i}) \in \varphi\} \\
s' & = & \{(d_i, c_i^{\theta'_i,R'_i,\sigma'_i}) \mid \exists a_i^{\theta_i,R_i,\sigma_i}.\,(d_i, a_i^{\theta_i,R_i,\sigma_i}) \in s\}
\end{array}
$$

and:

$$
c_i^{\theta'_i,R'_i,\sigma'_i} = \begin{cases} b^{\theta_i\theta,R_i\cup R,\sigma_i\sigma} & \text{if } a_i = a \\ a_i^{\theta_i,R_i,\sigma_i} & \text{if } a_i \neq a \end{cases}
$$

■

It is worth recalling that, in the clause for the parallel composition of the location oriented approach, an explicit track of the operator would be kept by decomposing $E_1 \parallel_S E_2$ into $dec_{\text{loc}}(E_1) \parallel_S id \oplus id \parallel_S dec_{\text{loc}}(E_2)$. Here, instead, according to the label oriented approach, we do not explicitly keep track of the parallel composition operator. As a consequence, instances of the same sequential term occurring within the scope of a parallel composition operator can be mapped onto the same place; e.g., term $a.\underline{0} \parallel_\emptyset a.\underline{0}$ is mapped onto a single place $a.\underline{0}$ with multiplicity two.

Let us see some examples showing how the decomposition works. In the case of hiding, the decomposition function maps term $(a.\underline{0})/\{a\}$ onto place $a^{\tau,\emptyset,\varepsilon}.\underline{0}$. This decoration provides the information that action a is hidden and this information will be subsequently used when generating net transitions.

In the case of relabeling, term $(a.\underline{0} + b.\underline{0})[\varphi]$, where $\varphi(a) = \varphi(b) = b$, is mapped onto place $b^{\varepsilon,\{r\},\emptyset}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\emptyset}.\underline{0}$. In this way, the two actions are kept distinct within the scope of the relabeling operator by their complementary conflicts r and $\bar{r}$. Such conflicts are necessary whenever parallel composition operators occur within the scope of the relabeling operator, as we shall see shortly. Again, this information will be used when generating net transitions.

In the case of parallel composition, term $a.\underline{0} \parallel_{\{a\}} a.\underline{0}$ is mapped onto the set of places $\{a^{\varepsilon,\emptyset,k}.\underline{0}, a^{\varepsilon,\emptyset,\bar{k}}.\underline{0}\}$ which can synchronize because they have complementary combinators k and $\bar{k}$.

As far as the interplay of parallel composition and hiding is concerned, binders are exploited to avoid the synchronization of hidden actions. As an example, in the term $(a.\underline{0})/\{a\} \parallel_{\{a\}} a.\underline{0}$ the synchronization of the two a actions is not possible because the left hand one is hidden. This is reflected at the net level by the fact that the generated places $a^{\tau,\emptyset,\varepsilon}.\underline{0}$ and $a^{\varepsilon,\emptyset,\bar{k}}.\underline{0}$ cannot synchronize because their combinators do not match. In particular, the combinator of the left hand a action is ε instead of k because that a action occurs within the scope of binder $/\{a\}$.

As far as the interplay of parallel composition and relabeling is concerned, the related decorations are used in a combined way in order to avoid the synchronization between different actions that collapse to the same action after their relabeling. As an example, in the term $(a.\underline{0} \parallel_{\{b\}} b.\underline{0})[\varphi]$, where $\varphi(a) = \varphi(b) = b$, no synchronization on b is possible because the parallel composition is within the scope of the relabeling and not the other way around. Again, this is reflected at the net level by the fact that the generated places $b^{\varepsilon,\{r\},k}.\underline{0}$ and $b^{\varepsilon,\{\bar{r}\},\bar{k}}.\underline{0}$ cannot synchronize, despite the fact that they have complementary combinators k and $\bar{k}$, because of the presence of complementary conflicts r and $\bar{r}$.

We observe that the identification of binders to avoid undesired substitutions is necessary mainly because, in the case of the relabeling operator, actions are directly relabeled within net places. As an example, consider term $(a.\underline{0} + b.\underline{0})/\{a\}[\varphi]$, where $\varphi(a) = \varphi(b) = b$, which can execute action τ or action b. When decomposing this term, there are two possible translations causing errors if $/\{a\}$ is not correctly taken into account. In the former case, one may think of relabeling actions occurring in $/\{a\}$ as well, thus erroneously obtaining term $(b^{\varepsilon,\{r\},\varepsilon}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0})/\{b\}$ which gives rise to place $b^{\tau,\{r\},\varepsilon}.\underline{0} + b^{\tau,\{\bar{r}\},\varepsilon}.\underline{0}$ from which only a transition labeled with τ can be generated. In the latter case, one may think of not considering $/\{a\}$ at all, thereby erroneously obtaining term $(b^{\varepsilon,\{r\},\varepsilon}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0})/\{a\}$ which gives rise to place $b^{\varepsilon,\{r\},\varepsilon}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0}$ from which only a transition labeled with b can be generated. In conclusion, actions occurring in binders can be neither relabeled nor ignored. For this reason, we decompose the term above into place $a^{\tau,\emptyset,\varepsilon}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0}$, where we note that the a action is not relabeled to b because it occurs in $/\{a\}$, from which two transitions labeled with τ resp. b can be generated, as expected. As another example, consider term $(a.\underline{0} + b.\underline{0}) \parallel_{\{a\}} (a.\underline{0} + b.\underline{0})[\varphi]$, where $\varphi(a) = \varphi(b) = b$, which can synchronously execute an a action which is turned into b or asynchronously execute two b actions. Similarly to the previous example, if we relabel $\parallel_{\{a\}}$ as $\parallel_{\{b\}}$ we erroneously introduce an additional synchronization between the two original b actions, while if we do not consider $\parallel_{\{a\}}$ at all we get rid of the synchronization between the two original a actions. In order to avoid the two cases above, we preliminarily rewrite the term as $(a.\underline{0}+b.\underline{0}) \parallel_{\{(a,a)\}} (a.\underline{0}+b.\underline{0})[\varphi]$. This gives rise to $(a.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0}) \parallel_{\{(a,b^{\varepsilon,\{r\},\varepsilon})\}} (a.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0})$ which results in the set of places $\{b^{\varepsilon,\{r\},k}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0}, b^{\varepsilon,\{r\},\bar{k}}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0}\}$ from which three transitions labeled with b can be generated, as expected. As usual, the application of each syntactical substitution may be preceded by alpha conversion to avoid binding free names. This is necessary e.g. when decomposing term $(a.\underline{0})/\{b\}[\varphi]$ where $\varphi(a) = \varphi(b) = b$.

Example 7 Let us a consider a simple system composed of two identical processes accessing the same memory module in mutual exclusion. This system can be described in $\mathcal{G}$ as follows:

$$\begin{aligned} \mathit{System} &\triangleq ((\mathit{Proc} \parallel_{\emptyset} \mathit{Proc}) \parallel_{\{\mathit{acq},\mathit{rel}\}} \mathit{Mem})/\{\mathit{comp}\} \\ \mathit{Proc} &\triangleq \mathit{comp}.\mathit{acq}.\mathit{use}.\mathit{rel}.\mathit{Proc} \\ \mathit{Mem} &\triangleq \mathit{acq}.\mathit{rel}.\mathit{Mem} \end{aligned}$$

where *comp* denotes a local computation (which is therefore hidden) while *acq*, *use*, and *rel* denote the acquisition, use, and release of the memory module, respectively. The decomposition of *System* computed by applying the rules in Def. 6 is $\{\!|\, \mathit{comp}^{\tau,\emptyset,\varepsilon}.\mathit{acq}^{\varepsilon,\emptyset,k_1}.\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}$, $\mathit{comp}^{\tau,\emptyset,\varepsilon}.\mathit{acq}^{\varepsilon,\emptyset,k_1}.\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}$, $\mathit{acq}^{\varepsilon,\emptyset,\bar{k}_1}.\mathit{rel}^{\varepsilon,\emptyset,\bar{k}_2}.\mathit{Mem} \,|\!\}$. Note that one place appears twice in the multiset, correctly capturing the presence of two independent replicas of *Proc*. ■

4.2. NET TRANSITIONS

The second step consists of connecting net places by means of net transitions. Unlike the location oriented approach, we will not have a net transition generating rule for each algebraic operator because static operators are not fully retained within places. This lack of information is however compensated for by action decorations introduced within places.

Let us reconsider the examples at the end of Sect. 4.1. Term $(a.\underline{0})/\{a\}$ is mapped onto place $a^{\tau,\emptyset,\varepsilon}.\underline{0}$ which becomes the preset of a transition labeled with τ by exploiting the information stored in the decoration. Term $(a.\underline{0} + b.\underline{0})[\varphi]$, where $\varphi(a) = \varphi(b) = b$, is mapped onto place $b^{\varepsilon,\{r\},\varepsilon}.\underline{0} + b^{\varepsilon,\{\bar{r}\},\varepsilon}.\underline{0}$ which becomes the preset of two (collapsing) transitions both labeled with b. Term $a.\underline{0} \,\|_{\{a\}}\, a.\underline{0}$ is mapped onto the set of places $\{a^{\varepsilon,\emptyset,k}.\underline{0}, a^{\varepsilon,\emptyset,\bar{k}}.\underline{0}\}$ which becomes the preset of a transition labeled with a using the combinators k and $\bar{k}$ stored in the decorations.

Formally, the set of transitions $\longrightarrow_{\mathcal{N}}$ is defined as the least subset of $\mathcal{P}(\mathcal{V}) \times Act \times \mathcal{M}(\mathcal{V})$ generated by the axiom:

$$\{\!|\sum_{h=1}^{n_i} a_{i,h}^{\theta_{i,h},R_{i,h},\sigma_{i,h}}.E_{i,h} \mid 1 \le i \le n \,|\!\} \xrightarrow{b}_{\mathcal{N}} \bigoplus_{i=1}^{n} \{\!|\, dec(E_{i,h_i}) \mid 1 \le h_i \le n_i\} \,|\!\}$$

where $n \in \mathbb{N}_+$ and $n_i \in \mathbb{N}_+$ for all $i = 1, \dots, n$. The preset of the transition is a finite nonempty multiset of places, from each of which a summand is selected (the one with index i, h_i). If the preset contains more than one place, then the generated transition results from the synchronization of the actions of the selected summands. The postset of the transition is the multiset composed of the places representing the decomposition of the derivative terms of the selected summands in the preset. The axiom is subject to the following conditions which are explained below:

1 For all $i, i' = 1, \dots, n$, $a_{i,h_i} = a_{i',h_{i'}}$ and $\theta_{i,h_i} = \theta_{i',h_{i'}}$.

2 For all $i, i' = 1, \dots, n$, it is not the case that $\{r, \bar{r}\} \subseteq R_{i,h_i} \cap R_{i',h_{i'}}$ whenever $i \neq i'$.

3 $\bigodot_{i=1}^{n} \sigma_i = \{\!|\, \epsilon \,|\!\}$.

4 $b = \begin{cases} a & \text{if } \theta_{1,h_1} = \epsilon \\ \tau & \text{if } \theta_{1,h_1} = \tau \end{cases}$

Condition 1 guarantees that the selected summands in the preset have the same action at the same visibility level. Conditions 2 and 3 ensure that the selected summands in the preset correctly synchronize by considering the decorations of the related actions. More precisely, condition 2 avoids the generation of undesired synchronization transitions which are due to a relabeling operator. Condition 3, instead, checks for the correct reduction of combinators to the empty string, which means that a synchronization is possible. Finally, condition 4 determines the action labeling the transition according to the hiding related decoration of one of the selected summands.

Example 8 Let us a consider again the simple system of Ex. 7. Starting from the two places $V_1 = \mathit{comp}^{\tau,\emptyset,\varepsilon}.\mathit{acq}^{\varepsilon,\emptyset,k_1}.\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}$ and $V_2 = \mathit{acq}^{\varepsilon,\emptyset,\bar{k}_1}.\mathit{rel}^{\varepsilon,\emptyset,\bar{k}_2}.\mathit{Mem}$ forming its decomposition, a single transition labeled with τ can be generated whose preset is V_1 and whose postset is given by V_3 obtained from $\mathit{dec}(\mathit{acq}^{\varepsilon,\emptyset,k_1}.\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}) = \{\!|\, \mathit{acq}^{\varepsilon,\emptyset,k_1}.\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc} \,|\!\}$. On the contrary, no transition can be generated having as preset the place V_2 because combinator k_1 does not reduce to the empty string. Such a reduction is instead possible when considering V_3 and V_2 together. They constitute the preset of a transition labeled with acq, whose postset is composed of places V_4 obtained from $\mathit{dec}(\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}) = \{\!|\, \mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc} \,|\!\}$ and V_5 obtained from $\mathit{dec}(\mathit{rel}^{\varepsilon,\emptyset,\bar{k}_2}.\mathit{Mem}) = \{\!|\, \mathit{rel}^{\varepsilon,\emptyset,\bar{k}_2}.\mathit{Mem} \,|\!\}$. By proceeding in this way, we can generate all the transitions in the net corresponding to *System*. ■

4.3. NETS ASSOCIATED WITH TERMS

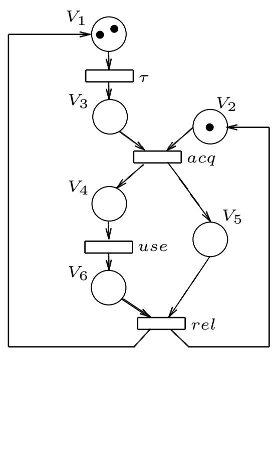
The third step consists of associating with each term an appropriate Petri net by exploiting the previous two steps.

Definition 9 The net semantics of $E \in \mathcal{G}$ is the net $\mathcal{N}[\![E]\!] = (P_{E,\mathcal{N}}, \mathit{Act}, \longrightarrow_{E,\mathcal{N}}, \mathit{dec}(E))$ where $P_{E,\mathcal{N}}$ is the least subset of $\mathcal{V}$ such that:

- $\mathit{dec}(E) \subseteq P_{E,\mathcal{N}}$;
- if $Q_1 \subseteq P_{E,\mathcal{N}}$ and $Q_1 \xrightarrow{a}_{\mathcal{N}} Q_2$, then $Q_2 \subseteq P_{E,\mathcal{N}}$,

and $\longrightarrow_{E,\mathcal{N}}$ is $\longrightarrow_{\mathcal{N}}$ restricted to $\mathcal{P}(P_{E,\mathcal{N}}) \times \mathit{Act} \times \mathcal{M}(P_{E,\mathcal{N}})$. ■

Example 10 Let us a consider again the simple system of Ex. 7 and 8. The corresponding Petri net is shown in Fig. 1. ■



$$\begin{aligned}
V_1 &\equiv \mathit{comp}^{\tau,\emptyset,\varepsilon}.\mathit{acq}^{\varepsilon,\emptyset,k_1}.\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}\\
V_2 &\equiv \mathit{acq}^{\varepsilon,\emptyset,\bar{k}_1}.\mathit{rel}^{\varepsilon,\emptyset,\bar{k}_2}.\mathit{Mem}\\
V_3 &\equiv \mathit{acq}^{\varepsilon,\emptyset,k_1}.\mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}\\
V_4 &\equiv \mathit{use}.\mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}\\
V_5 &\equiv \mathit{rel}^{\varepsilon,\emptyset,\bar{k}_2}.\mathit{Mem}\\
V_6 &\equiv \mathit{rel}^{\varepsilon,\emptyset,k_2}.\mathit{Proc}
\end{aligned}$$

Figure 1 $\mathcal{N}[\![\mathit{System}]\!]$

4.4. RETRIEVABILITY RESULT

We now show that the net semantics for $\mathcal{G}$ is correct w.r.t. the interleaving semantics, which means that the net semantics and the interleaving semantics of a given term represent the same system. Following [10], this is achieved by proving for all $E \in \mathcal{G}$ that $\mathcal{RG}[\![\mathcal{N}[\![E]\!]]\!]$ is strongly bisimilar [9] to $\mathcal{I}[\![E]\!]$. Note that these two LTSs are not isomorphic in general. For instance, Fig. 2 shows both $\mathcal{RG}[\![\mathcal{N}[\![System]\!]]\!]$ and $\mathcal{I}[\![System]\!]$. As it can be noted, there is no 1-1 correspondence between the states in the two LTSs, but a strong bisimulation can be recognized in which several states in $\mathcal{I}[\![System]\!]$ correspond to a single state in $\mathcal{RG}[\![\mathcal{N}[\![System]\!]]\!]$. This correspondence is represented through the subscripts in the name of the states; e.g., s_1 and s'_1 are bisimilar to M_1.

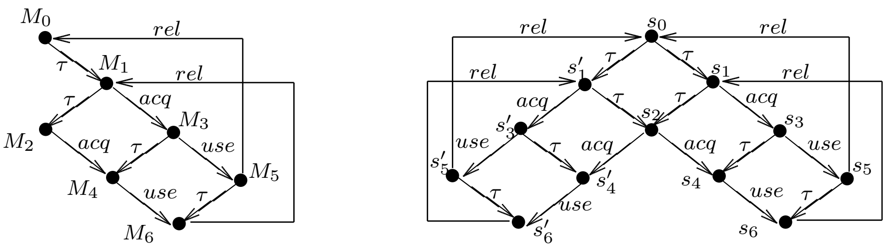


Figure 2 $\mathcal{RG}[\![\mathcal{N}[\![System]\!]]\!]$ and $\mathcal{I}[\![System]\!]$

Theorem 11 *Let $\mathcal{B} = \{(E, Q) \in \mathcal{G}' \times \mathcal{M}(\mathcal{V}) \mid Q = dec(E)\}$. Then $\mathcal{B}$ is a strong bisimulation.* ■

5. PROPERTIES OF THE SEMANTICS

5.1. COMPARISON AT THE NET LEVEL

In the location oriented approach of [7, 10] the resulting nets are 1-safe. As explained in Sect. 4.1, this is a consequence of the decomposition clause $dec_{\text{loc}}(E_1 \,\|_S\, E_2) = dec_{\text{loc}}(E_1) \,\|_S\, id \oplus id \,\|_S\, dec_{\text{loc}}(E_2)$ for the parallel composition operator, which always keeps distinct the two subnets corresponding to the two operand terms of the parallel composition operator. In our approach, instead, the two subnets can collapse into a single one depending on E_1, E_2, and S.

Theorem 12 *Let $E_1, E_2 \in \mathcal{G}$ be two sequential terms. Then $|dec(E_1 \,\|_S\, E_2)| \leq 2 = |dec_{\text{loc}}(E_1 \,\|_S\, E_2)|$. Moreover, if $E_1 \equiv E_2$ and $(sort(E_1) \cup sort(E_2)) \cap S = \emptyset$, then $|dec(E_1 \,\|_S\, E_2)| = 1$.* ■

The theorem above straightforwardly generalizes to an arbitrary number of sequential terms and gives an idea of the compactness that can be gained using the proposed label oriented net semantics. This justifies why the label oriented approach should be preferred to the location oriented approach from an applicative point of view, especially when dealing with large systems in which some parts are replicated.

5.2. COMPARISON AT THE STATE LEVEL

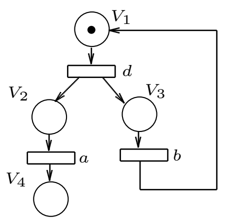
As shown in Fig. 2, $\mathcal{RG}[\![\mathcal{N}[\![System]\!]]\!]$ is smaller than $\mathcal{I}[\![System]\!]$. Actually, with an easy adaptation of the proof of the retrievability result, it can be shown that the reverse never happens.

Theorem 13 *Let $E \in \mathcal{G}$ be a finite state term and let $state(Z)$ be the set of states of LTS Z. Then $|state(\mathcal{RG}[\![\mathcal{N}[\![E]\!]]\!])| \leq |state(\mathcal{I}[\![E]\!])|$. In particular, if $E \equiv E_1 \,\|_S\, E_2$, then $|state(\mathcal{RG}[\![\mathcal{N}[\![E]\!]]\!])| < |state(\mathcal{I}[\![E]\!])|$ whenever $E_1 \equiv E_2$ and $(sort(E_1) \cup sort(E_2)) \cap S = \emptyset$.* ■

The second part of the theorem above straightforwardly generalizes to an arbitrary number of terms. Again, this is particularly important when dealing with large systems in which some parts are replicated since aggregated (w.r.t. the strong bisimulation equivalence) state spaces are obtained without applying minimization algorithms such as [11].

5.3. FINITENESS RESULT

Another nice property of our net semantics is that, in certain cases, it yields finite Petri nets for infinite state terms. As an example, consider $A \stackrel{\Delta}{=} d.(a.\underline{0} \,\|_\emptyset\, b.A)$. Then $\mathcal{N}[\![A]\!]$ is the finite net depicted in Fig. 3, while $\mathcal{N}_{\rm loc}[\![A]\!]$ has infinitely many places and $\mathcal{I}[\![A]\!]$ has infinitely many states.



$$\begin{array}{lcl} V_1 & \equiv & d.(a.\underline{0} \,\|_\emptyset\, b.A) \\ V_2 & \equiv & a.\underline{0} \\ V_3 & \equiv & b.A \\ V_4 & \equiv & \underline{0} \end{array}$$

Figure 3 Finite net semantics of the infinite state term $A \stackrel{\Delta}{=} d.(a.\underline{0} \,\|_\emptyset\, b.A)$

Theorem 14 *Let $E \in \mathcal{G}$ be an infinite state term. If the three following conditions hold:*

- *For each hiding operator $\_/L$ occurring within a recursive constant definition, $sort(E) \cap L = \emptyset$.*
- *For each relabeling operator $\_[\varphi]$ occurring within a recursive constant definition, $sort(E) \cap Dom(\varphi) = \emptyset$.*

- *For each parallel composition operator* $\_ \|_S \_$ *occurring within a recursive constant definition,* $sort(E) \cap S = \emptyset$.

then $\mathcal{N}[\![E]\!]$ *is finite while* $\mathcal{N}_{\rm loc}[\![E]\!]$ *has infinitely many places and* $\mathcal{I}[\![E]\!]$ *has infinitely many states.* ■

The theorem above shows that, for the considered class of infinite state terms, a finite hence analyzable representation can be obtained only with our label oriented net semantics.

5.4. STRUCTURAL REINTERPRETATION

For Petri nets there are several structural analysis techniques which can be used to derive system properties avoiding the generation of the underlying state space. The most important structural technique is the one based on the computation of P-invariants and T-invariants [12].

Definition 15 Let $N = (P, Act, T, M_0)$ be a Petri net such that $|P| = m$ and $|T| = n$. The *incidence matrix* of N is the matrix $\mathbf{A} = [a_{i,j}] \in \mathbb{Z}^{\,n \times m}$ where $a_{i,j} = i^\bullet(j) - {}^\bullet i(j)$. A *P-invariant* is an integer solution of $\mathbf{A} \cdot \mathbf{y} = \mathbf{0}$. A *T-invariant* is an integer solution of $\mathbf{A}^{\rm T} \cdot \mathbf{x} = \mathbf{0}$. ■

P-invariants single out places that do not change their token count during transition firings, whereas T-invariants indicate how often each transition has to fire in order to reproduce a given marking. P-invariants and T-invariants are computed starting from the incidence matrix and are independent from any initial marking, which is only instrumental for their interpretation. As an example of system properties that can be proved by exploiting invariants, it can be shown that a Petri net is bounded if for each of its places there exists a P-invariant associating a positive value with that place, while a necessary condition in order for a Petri net to be live and bounded is that for each of its transitions there exists a T-invariant associating a positive value with that transition.

A desirable property of a net semantics is that of allowing system properties derived from a Petri net to be reinterpreted at the level of the algebraic specification mapped onto that net. In this way one can take advantage of the efficiency of the structural techniques developed for Petri nets, thus importing such techniques in a formalism that does not directly support them. As far as invariants are concerned, our net semantics possesses this property because net places are in correspondence with (decorated) sequential subterms of the algebraic specification and net transitions are labeled with actions occurring in the algebraic specification.

Example 16 Let us consider again the system introduced in Ex. 7, whose net semantics is depicted in Fig. 1. Its incidence matrix $\mathbf{A}$ is

	V_1	V_2	V_3	V_4	V_5	V_6
τ	-1	0	1	0	0	0
acq	0	-1	-1	1	1	0
use	0	0	0	-1	0	1
rel	1	1	0	0	-1	-1

The net has four P-invariants $\mathbf{y}_1^{\mathrm{T}} = [\ 1\ 0\ 1\ 1\ 0\ 1\ ]$, $\mathbf{y}_2^{\mathrm{T}} = [\ 1\ 0\ 1\ 0\ 1\ 0\ ]$, $\mathbf{y}_3^{\mathrm{T}} = [\ 0\ 1\ 0\ 0\ 1\ 0\ ]$, and $\mathbf{y}_4^{\mathrm{T}} = [\ 0\ 1\ 0\ 1\ 0\ 1\ ]$ and one T-invariant $\mathbf{x}^{\mathrm{T}} = [\ 1\ 1\ 1\ 1\ ]$.

By exploiting the P-invariants above, we are able to prove that the memory is used in a mutually exclusive way. Since each P-invariant identifies a subset of the places of the net whose token count is invariant under transition firing, we know that

$$\begin{array}{rcl} M(V_1) + M(V_3) + M(V_4) + M(V_6) & = & 2 \\ M(V_1) + M(V_3) + M(V_5) & = & 2 \\ M(V_2) + M(V_5) & = & 1 \\ M(V_2) + M(V_4) + M(V_6) & = & 1 \end{array}$$

where M denotes the current marking while the constant occurring in each equation is determined by the initial marking. To prove mutual exclusion, we have to demonstrate that there is no state in which both processes can execute action *use*. At the net level, $V_4 \equiv \mathit{use}._{k_2}\mathit{rel}.\mathit{Proc}$ is the place representing a process ready to perform action *use*, so we have to demonstrate that there is no marking in which place V_4 contains more than one token. This can obviously be done by constructing the reachability graph of the net, but it can be also done more efficiently by exploiting the information provided by the fourth P-invariant, as it establishes that, for each reachable marking, the sum of the number of tokens in places V_2, V_4, and V_6 is one.

By exploiting the only T-invariant, instead, we are able to prove that the initial state of the algebraic specification is a home state, because the initial marking of the net enables the transition sequence τ, *acq*, *use*, *rel* whose transition count vector coincides with the T-invariant. ■

6. CONCLUSION

In this paper we have presented (and proved correct) a label oriented net semantics for a process algebra, which improves previous net semantics w.r.t. the size of the generated nets and allows system properties efficiently derived at the net level to be reinterpreted at the algebraic level. Because of its practical advantages, such a label oriented net semantics has been recently implemented to integrate the stochastically timed process algebra EMPA based software tool TwoTowers [3] with the generalized stochastic Petri net based software tool GreatSPN [6], in order to create a multiparadigm tool for the analysis of concurrent and distributed systems which benefits from the complementary strengths of process algebras and Petri nets.

As far as future work is concerned, we plan to investigate the possibility of extending our approach to a symbolic framework where value passing process algebras and colored Petri nets are considered.

References

[1] M. Bernardo, N. Busi, R. Gorrieri, *"A Distributed Semantics for EMPA Based on Stochastic Contextual Nets"*, in Computer Journal 38:492-509, 1995

[2] M. Bernardo, N. Busi, M. Ribaudo, *"Compact Net Semantics for Process Algebras"*, Tech. Rep. UBLCS-2000-02, University of Bologna (Italy), 2000

[3] M. Bernardo, W.R. Cleaveland, S.T. Sims, W.J. Stewart, *"TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems"*, in Proc. of FORTE/PSTV '98, Kluwer, pp. 457-467, 1998

[4] N. Busi, R. Gorrieri, *"A Petri Net Semantics for π-Calculus"*, in Proc. of CONCUR '95, LNCS 962:145-159, 1995

[5] N. Busi, R. Gorrieri, G. Zavattaro, *"On the Expressiveness of Linda Coordination Primitives"*, in Information and Computation 156:90-121, 2000

[6] G. Chiola, G. Franceschinis, R. Gaeta, M. Ribaudo, *"GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets"*, in Performance Evaluation 24:47-68, 1995

[7] P. Degano, R. De Nicola, U. Montanari, *"A Distributed Operational Semantics for CCS Based on Condition/Event Systems"*, in Acta Informatica 26:59-91, 1988

[8] U. Goltz, *"On Representing CCS Programs by Finite Petri Nets"*, in Proc. of MFCS '88, LNCS 324:339-350, 1988

[9] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989

[10] E.-R. Olderog, *"Nets, Terms and Formulas"*, Cambridge University Press, 1991

[11] R. Paige, R.E. Tarjan, *"Three Partition Refinement Algorithms"*, in SIAM Journal of Computing 16:973-989, 1987

[12] W. Reisig, *"Petri Nets: An Introduction"*, Springer-Verlag, 1985