P_BNDC and Replication*

Annalisa Bossi, Damiano Macedonio, Carla Piazza, and Sabina Rossi

Dipartimento di Informatica, Università Ca' Foscari di Venezia
{bossi,mace,piazza,srossi}@dsi.unive.it

Abstract. Information flow security in a multilevel system aims at guaranteeing that no high level information is revealed to low level users, even in the presence of any possible malicious process. *Persistent_BNDC* (*P_BNDC*, for short) is a information-flow security property for processes in dynamic contexts which guarantees confidentiality. In this work we show that the replication operator preserves *P_BNDC*. Then we define a proof system which provides a very efficient technique for the stepwise development and the verification of recursively defined *P_BNDC* processes.

Keywords: system security, confidentiality, information flow, noninterference, stepwise development.

1 Introduction

The production of large and complex systems that satisfy a given property strongly depends on the ability of dividing the task of the system into subtasks that are solved by system components. It is the classical divide-and-conquer approach, at the basis of any systematic development of complex systems. When security is the property of interest, difficulties can be encountered in applying this approach since secure systems might not be composed only by secure components. Nevertheless it is essential to know how properties of the components behave under composition. Many security conditions are compositional and many nontrivial security properties emerge under composition. General theories of compositionality exist for properties like safety and liveness [38, 1] and compositionality results for information-flow based confidentiality properties have also been developed [24, 27, 39, 22].

The problem of protecting confidential data in a multilevel system is one of the relevant issues in computer security. Information flow security assures confidentiality since it guarantees that no high level (confidential) information is revealed to users running at low levels [17, 25, 13, 31, 37, 34], even in the presence of any possible malicious process. To establish that information does not flow from high to low it is sufficient to establish that high behavior has no effect

* This work has been partially supported by the MURST project "Modelli formali per la sicurezza" and the EU Contract IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems" (MyThS).

on what low level users can observe, i.e., the low level view of the system is independent of high behavior. This notion of information flow security, known as Non-Interference, has been introduced in [18], and subsequently developed by many authors in many different settings [13, 14, 9, 20, 33, 35, 19].

In this paper we consider the security property *Persistent_BNDC* (*P_BNDC*, for short), proposed in [16], and further studied in [15, 4, 3, 5]. *P_BNDC* is a security property suitable to analyze processes in completely dynamic hostile environments, i.e., environments which can be dynamically reconfigured at run-time, changing in unpredictable ways. In [3] we gave an *unwinding* condition that allows us to express *P_BNDC* in terms of a local property of high level actions. As noticed in [21], unwinding conditions are useful for giving efficient proof techniques. Indeed, we use our local characterization to define a proof system which allows us to *statically prove* that a process is *P_BNDC*, i.e., by just inspecting its syntax. State-explosion is avoided by exploiting the compositionality of *P_BNDC* with respect to the parallel operator which is the source of the exponential growing of the number of states in a system. Moreover, the system offers a mean to build processes which are *P_BNDC* by construction in an incremental way. The proof system extends the one given in [23] for finite processes, i.e., processes that may only perform finite sequences of actions. In particular, we show how it is possible to deal also with recursive processes, which may perform unbounded sequences of actions. The process algebra language considered in [3] is SPA, a variation of Milner's CCS [29] where recursive constant definitions can be used to define recursive processes. The proof system had been proved to be correct, even if not complete. It is composed of two layers, a kernel where no constant definitions are considered and a second layer to deal with systems of definitions.

Many authors consider the replication operator, !, instead of constant definitions. The two approaches have the same expressive power in π-calculus [30, 36], but as recently proved in [10], replication cannot supplant recursion in CCS. In this paper we show how by allowing recursion only through replication we obtain a new compositionality result for the class of *P_BNDC* processes and a much simpler proof system which provides a very efficient technique for the stepwise development and the verification of recursively defined *P_BNDC* processes. We also identify a class of constant definitions which can be safely added to our language and treated by an extended proof system. In fact, we show that the encoding proposed in [30] for π-calculus is correct with respect to our class of definitions.

The paper is organized as follows. In Section 2 we introduce the language, recall the definition of *P_BNDC* process and its properties. In Section 3 we prove that *P_BNDC* is compositional with respect to the replication operator. In Section 3.2 we present a proof system which, by exploiting the new compositionality result, extends the kernel presented in [3] by adding recursion through replication in a very simple way. Section 4 is devoted to the (re)-introduction of constant definitions. Finally, in Section 5 we draw some conclusions. The paper

contains all the proofs of the results reported in Section 3, all the other proofs can be found in [6].

2 Basic Notions

2.1 The Language

In this section we report the syntax and semantics of the process algebra language we consider in this paper. It is a variation of Milner's CCS [29], similar to the SPA described in [13], where the set of visible actions is partitioned into high level actions and low level ones in order to specify multilevel systems. Differently from [29], as done also in [12], we start with the *replication* operator instead of the constant definitions. In Section 4 we will reintroduce constant definitions.

The syntax of our process algebra is based on the same elements as CCS that is: a set $\mathcal{L}$ of *visible* actions such that $\mathcal{L} = I \cup O$ where $I = \{a, b, \ldots\}$ is a set of *input* actions and $O = \{\bar{a}, \bar{b}, \ldots\}$ is a set of *output* actions; a special action τ which models internal computations, i.e., not visible outside the system; a complementation function $\bar{\cdot} : \mathcal{L} \to \mathcal{L}$, such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$, and $\bar{\tau} = \tau$; $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. The set of visible actions is partitioned into two sets, H and L, of high and low actions such that $\overline{H} = H$ and $\overline{L} = L$. A *process* E is a term built using the following productions:

$$E ::= \mathbf{0} \mid a.E \mid E + E \mid\mid E|E \mid E \setminus v \mid E[f] \mid !E$$

where $a \in Act$, $v \subseteq \mathcal{L}$, $f : Act \to Act$ is such that $f(L) \subseteq L \cup \{\tau\}$, $f(H) \subseteq H \cup \{\tau\}$, $f(\bar{\alpha}) = \overline{f(\alpha)}$ and $f(\tau) = \tau$.

Intuitively, $\mathbf{0}$ is the empty process that does nothing; $a.E$ is a process that can perform an action a and then behaves as E; $E_1 + E_2$ represents the nondeterministic choice between the two processes E_1 and E_2; $E_1|E_2$ is the parallel composition of E_1 and E_2, where executions are interleaved, possibly synchronized on complementary input/output actions, producing an internal action τ; $E \setminus v$ is a process E prevented from performing actions in v; $E[f]$ is the process E whose actions are renamed *via* the relabelling function f. The process $!E$ (bang E) means $E|E|\ldots$, i.e., the parallel composition of as many copy as needed of the process E.

Given a fixed language $\mathcal{L}$ we denote by $\mathcal{E}^!$ the set of all processes, by $\mathcal{E}^!_H$ the set of all high level processes, i.e., those constructed over $H \cup \{\tau\}$, and by $\mathcal{E}^!_L$ the set of all low level processes, i.e., those constructed over $L \cup \{\tau\}$,

The operational semantics of processes is given in terms of *Labelled Transition Systems* (LTS). A LTS is a triple $(V, A, \to)$ where V is a set of states, A is a set of labels (actions), $\to \subseteq V \times A \times V$ is a set of labelled transitions. The notation $(V_1, a, V_2) \in \to$ (or equivalently $V_1 \xrightarrow{a} V_2$) means that the system can move from the state V_1 to the state V_2 through the action a. The operational semantics of our language is the LTS $(\mathcal{E}^!, Act, \to)$, where the states are the terms of the algebra and the transition relation $\to \subseteq \mathcal{E}^! \times Act \times \mathcal{E}^!$ is defined by structural induction as the least relation generated by the inference rules reported in Figure 1. The

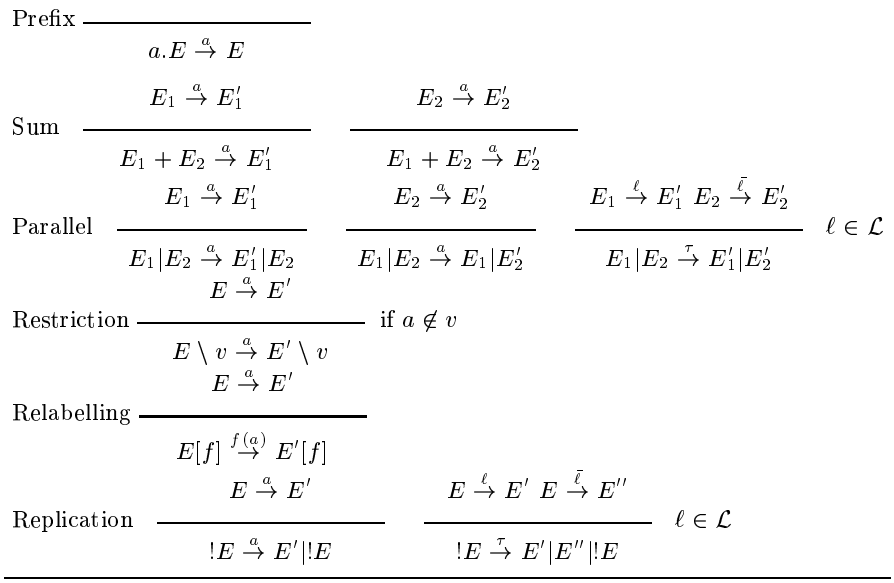

Fig. 1. The operational rules

operational semantics for a process E is the subpart of the LTS reachable from the initial state and we refer to it as $LTS(E) = (V_E, Act, \rightarrow)$.

In the paper we use the following notations. If $t = a_1 \cdots a_n \in Act^*$ and $E \xrightarrow{a_1} \cdots \xrightarrow{a_n} E'$, then we say that E' is reachable from E and write $E \xrightarrow{t} E'$, or simply $E \rightsquigarrow E'$. We also write $E \stackrel{t}{\Longrightarrow} E'$ if $E(\xrightarrow{\tau})^* \xrightarrow{a_1} (\xrightarrow{\tau})^* \cdots (\xrightarrow{\tau})^* \xrightarrow{a_n} (\xrightarrow{\tau})^* E'$ where $(\xrightarrow{\tau})^*$ denotes a (possibly empty) sequence of τ labelled transitions. If $t \in Act^*$, then $\hat{t} \in \mathcal{L}^*$ is the sequence gained by deleting all occurrences of τ from t. As a consequence, $E \stackrel{\hat{a}}{\Longrightarrow} E'$ stands for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\xrightarrow{\tau})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\stackrel{\hat{\tau}}{\Longrightarrow}$ means zero or more τ labelled transitions). Given two processes E, F we write $E \equiv F$ when E and F are syntactically equal.

The concept of *observation equivalence* between two processes is based on the idea that two systems have the same semantics if and only if they cannot be distinguished by an external observer. This is obtained by defining an equivalence relation over $\mathcal{E}^!$. We report the definitions of two observational equivalences known as *weak bisimulation* and *strong bisimulation* [29]. Intuitively, weak bisimulation equates two processes if they mutually simulate their behavior step by step, but it does not care about internal τ actions. So, when P simulates an action of Q, it can also execute some τ actions before or after that action.

Definition 1 (Weak Bisimulation). *A binary relation $\mathcal{R} \subseteq \mathcal{E}^! \times \mathcal{E}^!$ over processes is a* weak bisimulation *if $(E,F) \in \mathcal{R}$ implies, for all $a \in Act$,*

- *if $E \xrightarrow{a} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E',F') \in \mathcal{R}$;*
- *if $F \xrightarrow{a} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow} E'$ and $(E',F') \in \mathcal{R}$.*

Two processes $E, F \in \mathcal{E}^!$ are weakly bisimilar*, denoted by $E \approx F$, if there exists a weak bisimulation $\mathcal{R}$ containing the pair (E,F).*

The relation $\approx$ is the largest weak bisimulation and is an equivalence relation [29].

Strong bisimulation is stronger than weak bisimulation, since it consider the τ actions as all the other actions.

Definition 2 (Strong Bisimulation). *A binary relation $\mathcal{R} \subseteq \mathcal{E}^! \times \mathcal{E}^!$ over processes is a* strong bisimulation *if $(E,F) \in \mathcal{R}$ implies, for all $a \in Act$,*

- *if $E \xrightarrow{a} E'$, then there exists F' such that $F \xrightarrow{a} F'$ and $(E',F') \in \mathcal{R}$;*
- *if $F \xrightarrow{a} F'$, then there exists E' such that $E \xrightarrow{a} E'$ and $(E',F') \in \mathcal{R}$.*

Two processes $E, F \in \mathcal{E}^!$ are strong bisimilar*, denoted by $E \sim F$, if there exists a strong bisimulation $\mathcal{R}$ containing the pair (E,F).*

The relation $\sim$ is the largest weak bisimulation and is an equivalence relation. Moreover, two strongly bisimilar processes are also weakly bisimilar.

2.2 The *P_BNDC* Security Property

In this section we recall the *Persistent Bisimulation-based Non Deducibility on Compositions* (*P_BNDC*, for short) security property (see [16]) and its characterization in terms of unwinding condition (see [3]). We start by introducing an equivalence relation on low actions which allows us to simplify both the definition and the characterization of *P_BNDC*. Weak bisimulation on low actions is a sort of weak bisimulation which consider only the low actions. Hence, when two processes are weak bisimilar on low actions they cannot be distinguished by a low level user.

Definition 3 (Weak Bisimulation on Low Actions). *A binary relation $\mathcal{R} \subseteq \mathcal{E}^! \times \mathcal{E}^!$ over processes is a* weak bisimulation on low actions*, if $(E,F) \in \mathcal{R}$ implies, for all $a \in L \cup \{\tau\}$,*

- *if $E \xrightarrow{a} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E',F') \in \mathcal{R}$;*
- *if $F \xrightarrow{a} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow} E'$ and $(E',F') \in \mathcal{R}$.*

Two processes $E, F \in \mathcal{E}^!$ are weakly bisimilar on low actions*, denoted by $E \approx_l F$, if there exists a weak bisimulation on low actions $\mathcal{R}$ containing the pair (E,F).*

Lemma 1. *The relation $\approx_l$ is the largest weak bisimulation on low actions and it is an equivalence relation.*

Proof. From the definition of $\approx_l$ it follows that $\approx_l$ is the union of all the weak bisimulation on low actions. Hence if we prove that $\approx_l$ is a weak bisimulation on low actions we immediately get that it is the largest. Let $E, F \in \mathcal{E}^!$ be such that $E \approx_l F$. This means that there exists a weak bisimulation on low actions $\mathcal{R}$ such that $(E, F) \in \mathcal{R}$. If $E \xrightarrow{a} E'$, with $a \in L \cup \{\tau\}$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$. Hence, $E' \approx_l F'$. The case $F \xrightarrow{a} F'$ is similar.

Since the $\mathcal{I}d = \{(E, E) \mid E \in \mathcal{E}^!\}$ is a weak bisimulation on low actions, $\approx_l$ is reflexive. If $\mathcal{R}$ is a weak bisimulation on low actions, it is easy to prove that $\mathcal{R}^{-1} = \{(F, E) \mid (E, F) \in \mathcal{R}\}$ is a weak bisimulation on low actions. Hence $\approx_l$ is symmetric. The transitivity of $\approx_l$ follows from the fact that if $\mathcal{R}$ and $\mathcal{S}$ are weak bisimulations on low actions, then $R \cdot S = \{(E, G) \mid (E, F) \in \mathcal{R} \text{ and } (F, G) \in \mathcal{S}\}$ is a weak bisimulation on low actions. □

Lemma 2. *Let $E, F \in \mathcal{E}^!$.*

$$E \approx_l F \text{ iff } E \setminus H \approx F \setminus H.$$

Proof. $\Rightarrow$) It is sufficient to prove that: $\mathcal{R} = \{(E \setminus H, F \setminus H) \mid E \approx_l F\}$ is a weak bisimulation. Let E, F be processes such that $E \approx_l F$. If $E \setminus H \xrightarrow{a} E'$, then $a \in L \cup \{\tau\}$, $E' \equiv E'' \setminus H$ and $E \xrightarrow{a} E''$. Hence, $F \stackrel{\hat{a}}{\Longrightarrow} F''$, with $E'' \approx_l F''$, so we conclude that $F \setminus H \stackrel{\hat{a}}{\Longrightarrow} F'' \setminus H$, by Restriction, and $(E'' \setminus H, F'' \setminus H) \in \mathcal{R}$.The case $F \xrightarrow{a} F'$ is similar.

$\Leftarrow$) We can prove that $\mathcal{R} = \{(E, F) \mid E \setminus H \approx F \setminus H\}$ is a weak bisimulation on low actions. Let E, F be processes such that $E \setminus H \approx F \setminus H$. If $E \xrightarrow{a} E'$, with $a \in L \cup \{\tau\}$, then $E \setminus H \xrightarrow{a} E' \setminus H$, by Restriction, and so $F \setminus H \xrightarrow{a} F' \setminus H$ with $E' \setminus H \approx F' \setminus H$. This means that $(E', F') \in \mathcal{R}$ and we conclude. The case $F \xrightarrow{a} F'$ is similar. □

Using weak bisimulation on low actions we recall the notion of *Bisimulation-based Non Deducibility on Compositions* (*BNDC*, for short) [13] which is at the basis of *P_BNDC*. The *BNDC* security property aims at guaranteeing that no information flow from the high to the low level is possible, even in the presence of malicious processes. It is based on the idea of checking a system against all high level potential interactions, representing all possible high malicious programs. A system E is *BNDC* if for every high process Π a low user cannot distinguish E from $(E|\Pi)$, i.e., if Π cannot interfere [18] with the low level execution of E.

Definition 4 (BNDC). *Let $E \in \mathcal{E}^!$ be a process.*

$$E \in BNDC \text{ iff } \forall\ \Pi \in \mathcal{E}^!{}_H,\ E \approx_l (E|\Pi).$$

In [16] it is shown that *BNDC* is not strong enough for systems in dynamic execution environments. To deal with these situations, the property *P_BNDC* is introduced. Intuitively, a system E is *P_BNDC* if it never reaches insecure states.

Definition 5 (Persistent_BNDC). *Let $E \in \mathcal{E}^!$ be a process.*

$$E \in P\_BNDC \text{ iff } E \rightsquigarrow E' \text{ implies } E' \in BNDC.$$

Although the decidability of *BNDC* is still an open problem, *P_BNDC* is decidable (in polynomial time) as shown in [16] ([15]). In [3] another decidable characterization of *P_BNDC* processes has been proposed. It allows to express *P_BNDC* in terms of a local property of high level actions and it recalls the unwinding conditions proposed in other settings (e.g., [21, 11, 28, 32]). Also if we are using a variation of the SPA, with replications instead of constant definitions, the characterization presented in [3] holds.

Theorem 1 (Unwinding). *Let $E \in \mathcal{E}^!$ be a process.*

$$E \in P\_BNDC$$
$$\textit{iff}$$
$$\textit{if } E \rightsquigarrow E_i \xrightarrow{h} E_j, \textit{ then } E_i \stackrel{\hat{\tau}}{\Longrightarrow} E_k \textit{ and } E_j \approx_l E_k.$$

Proof. $\Leftarrow$) Let E be a process such that for all E_i reachable from E, if $E_i \xrightarrow{h} E_j$ then $E_i \stackrel{\hat{\tau}}{\Longrightarrow} E_k$ and $E_j \approx_l E_k$. Let

$$\mathcal{S} = \{(E_i, (E_i|\Pi)) \mid \Pi \in \mathcal{E}^!{}_H \text{ is a process and } E \rightsquigarrow E_i\}.$$

We prove that $\mathcal{S}$ is a weak bisimulation on low actions up to $\approx_l$. We have to consider the following cases:

- $(E_i|\Pi) \xrightarrow{\tau} (E_i|\Pi_1)$. Since $E_i \stackrel{\hat{\tau}}{\Longrightarrow} E_i$, by definition of $\mathcal{S}$, $(E_i, (E_i|\Pi_1)) \in \mathcal{S}$.
- $(E_i|\Pi) \xrightarrow{a} (E_j|\Pi)$, with $a \in L \cup \{\tau\}$. Hence $E_i \xrightarrow{a} E_j$ and, by definition of $\mathcal{S}$, $(E_j, (E_j|\Pi)) \in \mathcal{S}$.
- $(E_i|\Pi) \xrightarrow{\tau} (E_j|\Pi_1)$ where $E_i \xrightarrow{h} E_j$, with $h \in H$. By hypothesis $E_i \stackrel{\hat{\tau}}{\Longrightarrow} E_k$ and $E_j \approx_l E_k$. Hence, $E_k \approx_l E_j \; \mathcal{S} \; (E_j|\Pi_1))$.
- $E_i \xrightarrow{a} E_j$ with $a \in L \cup \{\tau\}$. Then, $(E_i|\Pi) \xrightarrow{a} (E_j|\Pi)$ and $(E_j, (E_j|\Pi)) \in \mathcal{S}$.

$\Rightarrow$) Let E be *P_BNDC*. Then, for all E_i reachable from E, $E_i \in P\_BNDC$. In particular, for all E_i reachable from E and for all $\Pi \in \mathcal{E}^!{}_H$, $E_i \approx_l (E_i|\Pi)$. Suppose that $E_i \xrightarrow{h} E_j$. Let $\Pi \equiv \bar{h}$. Then $(E_i|\Pi) \xrightarrow{\tau} E_j$. Since $E_i \approx_l (E_i|\Pi)$, $E_i \stackrel{\hat{\tau}}{\Longrightarrow} E_k$ and $E_j \approx_l E_k$. □

In [3] it has been proved that the class of *P_BNDC* processes enjoys the following compositionality properties.

Lemma 3. *The following properties hold:*

1. *if $E \in \mathcal{E}^!_L$, then $E \in P\_BNDC$;*
2. *if $E \in \mathcal{E}^!_H$, then $E \in P\_BNDC$;*
3. *if $E \in P\_BNDC$, then $E \setminus v \in P\_BNDC$;*
4. *if $E \in P\_BNDC$, then $E[f] \in P\_BNDC$;*
5. *if $E, F \in P\_BNDC$, then $E|F \in P\_BNDC$;*
6. *if $E_i, F_j \in P\_BNDC$, $i \in I$ and $j \in J$, then $\sum_{i \in I} a_i.E_i + \sum_{j \in J}(h_j.F_j + \tau.F_j) \in P\_BNDC$, where $a_i \in L$ and $h_j \in H$.*

3 *P_BNDC* and Replications

3.1 Compositionality Result

In this section we extend the compositionality result concerning *P_BNDC* to the replication operator.

We start by observing that the processes reachable from $!E$ can be characterized using the processes reachable from E. In particular, they are nothing but parallel composition of $!E$ with a finite number of processes reachable from E.

Lemma 4. *Let $E \in \mathcal{E}^!$ be a process. If $!E \rightsquigarrow E'$, then there exist $n \geq 0$ and $E_1, \ldots, E_n$ such that $E \rightsquigarrow E_i$, for $i = 1, \ldots, n$ and*

$$E' \equiv E_1|E_2|\ldots|E_n|!E.$$

Proof. By induction on the length ln of $\rightsquigarrow$.

If $ln = 0$, then $E' \equiv !E$, hence we have the thesis with $n = 0$.

Let us assume that we have proved the thesis for all the $ln \leq m$. Let $ln = m + 1$. This means that there exists E'' such that $!E \rightsquigarrow E''$ with m steps and $E'' \stackrel{a}{\rightarrow} E'$. By inductive hypothesis there exist $n \geq 0$ and $E_1, \ldots, E_n$ such that $E \rightsquigarrow E_i$, for $i = 1, \ldots, n$ and $E'' \equiv E_1|E_2|\ldots|E_n|!E$. If the action a is performed by one of the E_i's, say E_1, we have the thesis, since $E \rightsquigarrow E_1 \stackrel{a}{\rightarrow} E_1'$ and $E' \equiv E_1'|E_2|\ldots|E_n|!E$. Similarly we obtain the thesis if $a = \tau$ is a synchronization between two of the E_i's. If the action a is performed by $!E$ applying the first rule of Replication, then $E \stackrel{a}{\rightarrow} E_{n+1}$ and $E' \equiv E_1|E_2|\ldots|E_n|E_{n+1}|!E$. Similarly we obtain the thesis in the remaining two cases, i.e. if a is performed by $!E$ applying the second rule of Replication or if a is a synchronization between one of the E_i's and $!E$. □

There is an interesting connections between the processes reachable from E and the processes reachable from $!E$, when E is *P_BNDC*: if $2n$ processes reachable from a E are pairwise weak bisimilar on low actions, this relation is preserved also on the processes reachable from $!E$ that they characterize.

Lemma 5. *Let E be a* P_BNDC *process, $n \geq 0$ and $\forall i \in \{1, .., n\}$ F_i, G_i be reachable from E. If $\forall i \in \{1, .., n\}$ $F_i \approx_l G_i$ then*

$$F_1|F_2\ldots|F_n|!E \approx_l G_1|G_2\ldots|G_n|!E.$$

Proof. Let E be a *P_BNDC* process and

$$\begin{aligned}\mathcal{R} = \{&(F_1|F_2\ldots|F_n|!E,\ G_1|G_2\ldots|G_n|!P) \mid \\ &n \geq 0, \forall i \in \{1, .., n\}\ F_i, G_i \text{ are reachable from } E \text{ and } F_i \approx_l G_i\}.\end{aligned}$$

We prove that $\mathcal{R}$ is a weak bisimulation on low actions. We have to consider the following three cases.

- No synchronization. Let $a \in L \cup \{\tau\}$ and assume $F_1|F_2\ldots|F_n|!E \stackrel{a}{\longrightarrow} E'$. There are three subcases.

- The action is performed by one of the F_i's, say F_1.
  Hence $E' \equiv F_1'|F_2 \ldots |F_n|!E$ where $F_i \xrightarrow{a} F_i'$. Since $F_1 \approx_l G_1$ there exists G_1' such that $G_1 \stackrel{\hat{a}}{\Longrightarrow} G_1'$ and $G_1' \approx_l G_1'$. Thus $G_1|G_2 \ldots |G_n|!E \stackrel{\hat{a}}{\Longrightarrow} G_1'|G_2 \ldots |G_n|!E$ and $(F_1'|F_2 \ldots |F_n|!E,\ G_1'|G_2 \ldots |G_n|!E) \in \mathcal{R}$.
- The action is performed by $!E$. If $a \in L$, then $E' \equiv F_1|F_2 \ldots |F_n|F_{n+1}|!E$ where $E \xrightarrow{a} F_{n+1}$. Clearly $G_1|G_2 \ldots |G_n|!E \stackrel{\hat{a}}{\Longrightarrow} G_1|G_2 \ldots |G_n|F_{n+1}|!E$ and $(F_1|F_2 \ldots |F_n|F_{n+1}|!E,\ G_1|G_2 \ldots |G_n|F_{n+1}|!E) \in \mathcal{R}$. If $a = \tau$, we similarly get the thesis applying the second rule for the semantics of the replication.

– Synchronization on low actions. There are two subcases.
- Two of the F_i's synchronize. Without loss of generality we can assume that $F_1|F_2 \ldots |F_n|!E \xrightarrow{\tau} F_1'|F_2' \ldots |F_n|!E$ where $F_1 \xrightarrow{a} F_1'$ and $F_2 \xrightarrow{\bar{a}} F_2'$. Since $F_1 \approx_l G_1$ and $F_2 \approx_l G_2$ there exist G_1', G_2' such that $G_1 \stackrel{\hat{a}}{\Longrightarrow} G_1',\ G_1' \approx_l F_1',\ G_2 \stackrel{\hat{\bar{a}}}{\Longrightarrow} G_2',\ G_2' \approx_l F_2'$. Then $G_1|G_2 \ldots |G_n|!E \stackrel{\hat{\tau}}{\Longrightarrow} G_1'|G_2' \ldots |G_n|!E$ and $(F_1'|F_2' \ldots |F_n|!E,\ G_1'|G_2' \ldots |G_n|!E) \in \mathcal{R}$.
- One of the F_i's synchronizes with $!E$. Similar to the previous case.

– Synchronization on high actions. There are two subcases.
- Two of the F_i's synchronize. Without loss of generality we can assume that $F_1|F_2 \ldots |F_n|!E \xrightarrow{\tau} F_1'|F_2' \ldots |F_n|!E$ where $F_1 \xrightarrow{h} F_1'$ and $F_2 \xrightarrow{\bar{h}} F_2'$, $h \in H$. Since E is *P_BNDC* and both F_1 and F_2 are reachable from E, by the unwinding Theorem 1 there exist G_1'' and G_2'' such that

  $$F_i \stackrel{\hat{\tau}}{\Longrightarrow} G_i'',\ F_i' \approx_l G_i'',\ i = 1, 2.$$

  Since $F_i \approx_l G_i$, $i = 1, 2$, there exist G_1' and G_2' such that

  $$G_i \stackrel{\hat{\tau}}{\Longrightarrow} G_i',\ G_i'' \approx_l G_i', i = 1, 2.$$

  Hence $F_i' \approx_l G_i'$, $i = 1, 2$, and $G_1|G_2 \ldots |G_n|!E \stackrel{\hat{\tau}}{\Longrightarrow} G_1'|G_2' \ldots |G_n|!E$ where $(F_1'|F_2' \ldots |F_n|!E,\ G_1'|G_2' \ldots |G_n|!E) \in \mathcal{R}$.
- one of the F_i's synchronizes with $!E$. Without loss of generality we can assume that $F_1|F_2 \ldots |F_n|!E \xrightarrow{\tau} F_1'|F_2 \ldots |F_n|E'|!E$ where $F_1 \xrightarrow{h} F_1'$ and $!E \xrightarrow{\bar{h}} E'|!E$, $h \in H$. Hence $E \xrightarrow{\bar{h}} E'$. Since E is *P_BNDC* and both F_1 and E' are reachable from E, by the unwinding Theorem 1 there exist G_1'' and E'' such that $F_1 \stackrel{\hat{\tau}}{\Longrightarrow} G_1'',\ F_1' \approx_l G_1''$ and $E \stackrel{\hat{\tau}}{\Longrightarrow} E'',\ E' \approx_l E''$. Since $F_1 \approx_l G_1$, there exists G_1' such that $G_1 \stackrel{\hat{\tau}}{\Longrightarrow} G_1',\ G_1'' \approx_l G_1'$. Hence $F_1' \approx_l G_1'$, and $G_1|G_2 \ldots |G_n|!E \stackrel{\hat{\tau}}{\Longrightarrow} G_1'|G_2 \ldots |E''|!E$ where it holds $(F_1'|F_2 \ldots |F_n|E'|!E,\ G_1'|G_2 \ldots |E''|!E) \in \mathcal{R}$.

□

We can now prove that *P_BNDC* is compositional with respect to the replication operator.

Theorem 2. *Let $E \in \mathcal{E}^!$ be a process. If $E \in$ P_BNDC, then $!E \in$ P_BNDC.*

Proof. We prove that if F is reachable from $!E$, then F is *P_BNDC*. We prove that F satisfies the unwinding condition (see Theorem 1). By Lemma 4 we have that $F \equiv F_1|F_2 \ldots |F_n|!E$. Let $F_1|F_2 \ldots |F_n|!E \xrightarrow{h} F'$. If the action is performed by one of the F_i's, say F_1, then $F' \equiv F_1'|F_2 \ldots |F_n|!E$ where $F_1 \xrightarrow{h} F_1'$. By the unwinding condition on F_1, there exists G_1' such that $F_1 \stackrel{\hat{\tau}}{\Longrightarrow} G_1'$, $G_1' \approx_l F_1'$. Hence $F_1|F_2 \ldots |F_n|!E \stackrel{\hat{\tau}}{\Longrightarrow} G_1'|F_2 \ldots |F_n|!E$ and, by Lemma 5, $F_1'|F_2 \ldots |F_n|!E \approx_l G_1'|F_2 \ldots |F_n|!E$. If the action is performed by $!E$, then $F' \equiv F_1|F_2 \ldots |F_n|E'|!E$ and $E \xrightarrow{h} E'$. By the unwinding condition on E, there exists E'' such that $E \stackrel{\hat{\tau}}{\Longrightarrow} E''$, $E'' \approx_l E'$. Hence $F_1|F_2 \ldots |F_n|!E \stackrel{\hat{\tau}}{\Longrightarrow} F_1|F_2 \ldots |F_n|E''|!E$ and, by Lemma 5, $F_1|F_2 \ldots |F_n|E'|!E \approx_l F_1|F_2 \ldots |F_n|E''|!E$. □

3.2 A Proof System for Processes with Replications

In [3] it has been presented a proof system which allows us to build *P_BNDC* processes in an incremental way. The proof system is composed by a set of rules whose conclusions are in the form $E \in \mathcal{HP}[A]$, where A is a set of constants. The intended meaning of the judgment is that E is a *P_BNDC* process provided that all the constants in A are *P_BNDC*. The set A plays the role of a set of assumptions: if it is empty then E is *P_BNDC* otherwise we are still working on our construction under open hypothesis. It is immediate to observe that the system described in [3] is correct also using set of processes (meta-variables), instead of set of constants, as assumptions. To this end we extend the standard notation $E[X]$, where X denotes a variable to $E[G]$, where G denotes a process whose occurrences in E can be syntactically and unambiguously identified. Consequently, we use $E[F/G]$ to denote the process we obtain by replacing all the occurrences of G in E with F. Hence, in this section the meaning of $E \in \mathcal{HP}[A]$ is that E is a *P_BNDC* process provided that all the processes in A are *P_BNDC*. In this section we show how to exploit Lemma 3 and Theorem 2 in order to extend the system to the case of processes with replication.

Definition 6 (*P_BNDC − System*)**.** *P_BNDC − System is the proof system containing the following rules.*

$$\frac{}{E \in \mathcal{HP}[\{E\}]}\ E \text{ is a process} \quad (Proc)$$

$$\frac{}{E \in \mathcal{HP}[\emptyset]}\ E \in \mathcal{E}_L^! \quad (Low)$$

$$\frac{}{E \in \mathcal{HP}[\emptyset]}\ E \in \mathcal{E}_H^! \quad (High)$$

$$\frac{E \in \mathcal{HP}[A]}{E \setminus v \in \mathcal{HP}[A]}\ (Rest)$$

$$\frac{E \in \mathcal{HP}[A]}{E[f] \in \mathcal{HP}[A]}\ (Label)$$

$$\frac{E \in \mathcal{HP}[A] \quad F \in \mathcal{HP}[B]}{E|F \in \mathcal{HP}[A \cup B]} \ (Par)$$

$$\frac{E_i \in \mathcal{HP}[A_i] \quad F_j \in \mathcal{HP}[B_j]}{\sum_{i\in I} a_i.E_i + \sum_{j\in J}(h_j.F_j + \tau.F_j) \in \mathcal{HP}[\cup_{i\in I} A_i \cup \cup_{j\in J} B_j]} \ \begin{array}{l} a_i \in L \cup \{\tau\}, h_j \in H \\ (Choice) \end{array}$$

$$\frac{E \in \mathcal{HP}[A]}{!E \in \mathcal{HP}[A]} \ (Repl)$$

$$\frac{E[G] \in \mathcal{HP}[A] \quad F \in \mathcal{HP}[B]}{E[F/G] \in \mathcal{HP}[(A \setminus \{F\}) \cup B]} \ (Subst)$$

Theorem 3 (Correctness). *The system P_BNDC − System is correct, i.e., if there exists a proof in P_BNDC − System which ends with $E \in \mathcal{HP}[A]$, then E is P_BNDC provided that all the processes in A are P_BNDC.*

Proof. Rule $(Proc)$ is trivially correct.

By using Lemma 3 we obtain the correctness of rules (Low), $(High)$ $(Rest)$, $(Label)$, (Par), $(Choice)$ in the case in which $A = \emptyset$. The general case follows immediately by the definition of $\mathcal{HP}[A]$.

The correctness of rule $(Repl)$ follows from Theorem 2 in the case in which $A = \emptyset$. The general case follows again by the definition of $\mathcal{HP}[A]$.

The correctness of rule $(Subst)$ is provable by induction on depth of the derivations and by using Lemma 3 and Theorem 2.

□

Next corollary is an immediate consequence of Theorem 3.

Corollary 1. *Let $E \in \mathcal{E}^!$. If there exists a proof of $E \in \mathcal{HP}[\emptyset]$, then E is P_BNDC.*

Example 1. Consider the process CH defined as

$$\begin{array}{l} CH \equiv ((in_0.(\overline{out_0}.\overline{\sigma}.\mathbf{0} + \tau.\overline{\sigma}.\mathbf{0}) + in_1.(\overline{out_1}.\overline{\sigma}.\mathbf{0} + \tau.\overline{\sigma}.\mathbf{0}))| \\ \qquad !(\sigma.(in_0.(\overline{out_0}.\overline{\sigma}.\mathbf{0} + \tau.\overline{\sigma}.\mathbf{0}) + in_1.(\overline{out_1}.\overline{\sigma}.\mathbf{0} + \tau.\overline{\sigma}.\mathbf{0})))) \setminus \{\sigma, \overline{\sigma}\} \end{array}$$

where $in_0, in_1, \sigma, \overline{\sigma} \in L$ and $\overline{out_0}, \overline{out_1} \in H$. This process CH is a channel which may accept a value 0 (or 1) through the low level input in_0 (or in_1). When it holds a value, it may deliver it through a high level output $\overline{out_0}$ (or $\overline{out_1}$). The channel can transmit values infinitely many times. In fact, when the $\overline{\sigma}$ action is reached the process resets itself and recursively repeats the sequence of actions.

This process is nothing but the channel described in [29]. This correspondence will be clarified in the next section.

It is easy to see that we can derive the judgement $CH \in \mathcal{HP}[\emptyset]$ in P_BNDC-system. Both $\overline{out_0}.\overline{\sigma}.\mathbf{0}+\tau.\overline{\sigma}.\mathbf{0} \in \mathcal{HP}[\emptyset]$ and $\overline{out_1}.\overline{\sigma}.\mathbf{0}+\tau.\overline{\sigma}.\mathbf{0} \in \mathcal{HP}[\emptyset]$ are derivable applying the rule $(Choice)$. Then, using the rules $(Choice)$, (Par), $(Repl)$, and $(Rest)$ in a straightforward way we get the thesis.

4 Adding Constant Definitions

In this section we add some constant definitions to our language. Then, exploiting the compositionality of *P_BNDC* with respect to the replication operator, we prove a compositionality result for *P_BNDC* with respect to the constant definitions we consider.

Note that we do not add all constant definitions, since in CCS, differently from π-calculus [36], replication is not expressive enough to represent all constant definitions [10].

4.1 Definitions using Replications

In standard CCS [29] we find that complex recursive systems are defined parametrically, as

$$Z \stackrel{\text{def}}{=} E[Z],$$

where Z is a process identifier and $E[Z]$ a process expression which may contain "calls" to Z and eventually other parametric processes.

Example 2. Consider the process Z recursively defined as

$$Z \stackrel{\text{def}}{=} a.Z + b.\mathbf{0}$$

Intuitively this process can perform either an action a and return in its initial state or an action b and terminate. Similarly it is possible to consider two mutually defined processes X and Y defined as

$$\begin{cases} X \stackrel{\text{def}}{=} a.Y \\ Y \stackrel{\text{def}}{=} b.X \end{cases}$$

In this case X performs an action a, then it calls Y. Similarly Y performs an action b and calls X.

This way of defining recursive processes was taken as basic in [13] and in other previous works on *P_BNDC* (see [3–5]). In the context of the π-calculus in [30], an encoding is defined which eliminates a finite number of constant definitions using replication. As already noticed in [36], the same encoding applied to full CCS does not work (see also Remark 1). In what follows we identify a fragment of CCS on which the encoding is correct.

Let $Act = \mathcal{L} \cup \{\tau\}$ be a set of actions, with $\mathcal{L}$ partitioned into the two sets H and L, as described in Section 2.1. Let $\mathcal{C}$ be a finite set of constants. Consider all the processes D which can be obtained using the following productions:

$$D ::= \mathbf{0} \mid a.D \mid D + D \mid D|D \mid Z$$

where $Z \in \mathcal{C}$ is a constant which must be associated to a definition $Z \stackrel{\text{def}}{=} D$. Let $\mathcal{E}^{\text{def}}$ be the set of processes defined with this syntax. Given a process D, $const(D)$ denotes all the constants which occur in D. We say that a process D is *constant-free* if $const(D) = \emptyset$.

Example 3. The process Z informally presented in Example 2 is a process of $\mathcal{E}^{\mathrm{def}}$. In particular Z is a constant to which we associate the definition $Z \stackrel{\mathrm{def}}{=} a.Z + b.\mathbf{0}$.

Similarly the processes X and Y of Example 2 are two constants whose definitions are mutually recursive.

In order to define the semantics of the processes in $\mathcal{E}^{\mathrm{def}}$ we add to the rules of Figure 1 the following rule to deal with constant definitions.

$$\text{Constant} \ \frac{}{Z \stackrel{\tau}{\rightarrow} D} \ \text{if } Z \stackrel{\mathrm{def}}{=} D$$

This rule tells us that Z performs a τ transition and then behaves as D.

Example 4. Let Z be the constant defined in Example 2. By applying once the rule Constant we obtain that $Z \stackrel{\tau}{\rightarrow} a.Z + b.\mathbf{0}$ then $a.Z + b.\mathbf{0} \stackrel{b}{\rightarrow} \mathbf{0}$ or $a.Z + b.\mathbf{0} \stackrel{a}{\rightarrow} Z$. In the second case we can apply again the rule Constant.

Let X and Y be the constants defined in the second part of Example 2. We have that $X \stackrel{\tau}{\rightarrow} a.Y \stackrel{a}{\rightarrow} Y$ and $Y \stackrel{\tau}{\rightarrow} b.X \stackrel{b}{\rightarrow} X$.

All the processes in $\mathcal{E}^{\mathrm{def}}$ can be translated into an equivalent (bisimilar) process of the language $\mathcal{E}^{!}$ with restriction and replication and without constant definition presented in Section 2.1.

We briefly recall how the encoding which removes the constant definitions works. Let $Z_1, \ldots, Z_n$ be n constants defined as

$$Z_i \stackrel{\mathrm{def}}{=} D_i,$$

where for all $i = 1, \ldots, n$ $const(D_i) \subseteq \{Z_1, \ldots, Z_n\}$. Let $S = \{\sigma_1, \overline{\sigma_1}, \ldots, \sigma_n, \overline{\sigma_n}\}$ be a new set of actions disjoint from Act. We associate to the constant Z_i the action σ_i and we introduce the notation [1]:

$$\widehat{Z}_i \ \equiv !(\sigma_i.D_i[\overline{\sigma_1}.\mathbf{0}/Z_1, \ldots, \overline{\sigma_n}.\mathbf{0}/Z_n]),$$

where in D_i the constants $Z_1, \ldots, Z_n$ have been substituted by the constant-free expressions $\overline{\sigma_1}.\mathbf{0}, \ldots, \overline{\sigma_n}.\mathbf{0}$. Since $const(D_i) \subseteq \{Z_1, \ldots, Z_n\}$, $\widehat{Z}_i$ is a constant-free expression.

Definition 7 (Encoding of $\mathcal{E}^{\mathrm{def}}$). *Given a process D with $const(D) \subseteq \{Z_1, \ldots, Z_n\}$ its* encoding $[\![D]\!]$ *is the constant-free process*

$$[\![D]\!] \ \equiv \ (D[\overline{\sigma_1}.\mathbf{0}/Z_1, \ldots, \overline{\sigma_n}.\mathbf{0}/Z_n] | \widehat{Z}_1 | \ldots | \widehat{Z}_n) \setminus S.$$

In particular, when D is one of the Z_i's we obtain

$$[\![Z_i]\!] \ \equiv \ (\overline{\sigma_i}.\mathbf{0} | \widehat{Z}_1 | \ldots | \widehat{Z}_n) \setminus \{\sigma_1, \ldots, \sigma_n\}.$$

[1] We use the notation $D[Z_1, \ldots, Z_n]$ when we want to stress the fact that the constants $Z_1, \ldots, Z_n$ can occur in D.

Example 5. Let Z the constant defined in Example 2. We obtain that the encoding of Z is

$$[\![Z]\!] \equiv (\overline{\sigma}.\mathbf{0} | !(\sigma.\mathbf{0}.(a.\overline{\sigma}.\mathbf{0} + b.\mathbf{0}))) \setminus S$$

Notice that $\widehat{Z}$ which is

$$\widehat{Z} \equiv !(\sigma.\mathbf{0}.(a.\overline{\sigma}.\mathbf{0} + b.\mathbf{0}))$$

is different from $[\![Z]\!]$.

Remark 1. In the encoding the action $\overline{\sigma_i}$ is used to make a "call to the procedure" Z_i which is represented by $\widehat{Z}_i$. The encoding does not work in the full CCS, since the scope of the restrictions and renamings is not enlarged to the $\widehat{Z}_i$. Consider for instance a constant Z defined as

$$Z \stackrel{\text{def}}{=} a.Z$$

and the process $E \equiv (Z) \setminus \{a\}$. The process E can only perform a τ action, then it terminates. If apply our encoding to E we obtain

$$[\![E]\!] \equiv ((\overline{\sigma}.\mathbf{0}) \setminus \{a\} | !(\sigma.a.\overline{\sigma}.\mathbf{0})) \setminus S.$$

The process $[\![E]\!]$ performs a τ, and then it is able to perform an action a, since in $\widehat{Z}$ the action a is allowed. We will see that in this case we are able to overcome the problem. A more complex example of a process which cannot be encoded can be obtained using two mutual recursive constant definitions

$$\begin{cases} X \stackrel{\text{def}}{=} (a.Y) \setminus \{b\} \\ Y \stackrel{\text{def}}{=} (b.X) \setminus \{a\} \end{cases}$$

The process $F \equiv X$ performs a τ transition, followed by an a transition, then it performs another τ transition and it terminates. Its encoding would be

$$[\![F]\!] \equiv (\overline{\sigma_X}.\mathbf{0} | !(\sigma_X.(a.\overline{\sigma_Y}.\mathbf{0}) \setminus \{b\}) | !(\sigma_Y.(b.\overline{\sigma_X}.\mathbf{0}) \setminus \{a\})) \setminus S$$

The process $[\![F]\!]$ does not terminate and it performs an infinite number of b actions. The solution we will apply later to enlarge our encoding cannot be applied to this process.

Remark 2. The encoding is not compositional, for instance, given $D_1, D_2 \in \mathcal{E}^{\text{def}}$, $[\![D_1 \mid D_2]\!] \neq [\![D_1]\!] \mid [\![D_2]\!]$. However we can show that the encoding is compositional with respect to strong bisimulation, as stated in the following lemma.

Lemma 6. *Given $D, D_1, D_2 \in \mathcal{E}^{\text{def}}$ then:*

1. $[\![a.D]\!] \sim a.[\![D]\!]$;
2. $[\![Z]\!] \sim \tau.[\![D_0]\!]$ *(provided that $Z \stackrel{\text{def}}{=} D_0$);*
3. $[\![D_1 + D_2]\!] \sim [\![D_1]\!] + [\![D_2]\!]$;
4. $[\![D_1 \mid D_2]\!] \sim [\![D_1]\!] \mid [\![D_2]\!]$.

Proof. See appendix. □

Moreover we can show that there is a strong correspondence between the actions of D and the actions of $[\![D]\!]$. The following theorem states the observational equivalence between D and $[\![D]\!]$ when D belongs to $\mathcal{E}^{\mathrm{def}}$. Since $D \in \mathcal{E}^{\mathrm{def}}$ and $[\![D]\!] \in \mathcal{E}^{!}$ the bisimulation we establish is a relation on $\mathcal{E}^{\mathrm{def}} \times \mathcal{E}^{!}$.

Theorem 4. *For each $D \in \mathcal{E}^{\mathrm{def}}$ it holds $D \sim [\![D]\!]$.*

Proof. See appendix. □

The actions σ_i's introduced in the encoding are neither high nor low level actions. They are used only in the encoding, in order to obtain constant free-processes, but they are not visible outside because of the outmost restriction. Indeed, they are introduced only to *fire* infinitely many times the actions of the D_i's. Nevertheless, we have to decide how to treat them in the definition of the attackers and in the definition of the low level observational equivalence. We consider this issue in the next section.

Before moving to our security property we show how to apply our encoding to a richer language in which restriction and renaming can be used "outside" the recursive definitions. In particular, consider all the processes E defined by the following productions:

$$E ::= \mathbf{0} \mid a.E \mid E + E \mid E|E \mid E \setminus v \mid E[f] \mid !E \mid Z$$

where $Z \in \mathcal{C}$ is a constant which must be associated to a definition $Z \stackrel{\mathrm{def}}{=} D$, with $D \in \mathcal{E}^{\mathrm{def}}$. Let $\mathcal{E}^{\mathrm{def}!}$ be the set of processes defined with this syntax.

Since the constants are defined using processes in $\mathcal{E}^{\mathrm{def}}$, by Theorem 4, we have that $Z \sim [\![Z]\!]$. Using the fact that $\sim$ is a congruence on our language we immediately get that the following encoding can be applied to the processes in $\mathcal{E}^{\mathrm{def}!}$.

Definition 8 (Encoding of $\mathcal{E}^{\mathrm{def}!}$). *Let $E \in \mathcal{E}^{\mathrm{def}!}$ be a process such that $const(E) \subseteq \{Z_1, \ldots, Z_n\}$ its* encoding $\{\!\{E\}\!\}$ *is the constant-free process*

$$\{\!\{E\}\!\} \equiv E[[\![Z_1]\!]/Z_1, \ldots, [\![Z_n]\!]/Z_n].$$

Corollary 2. *For each $E \in \mathcal{E}^{\mathrm{def}!}$ it holds $E \sim \{\!\{E\}\!\}$.*

Example 6. Consider the constant Z and the process E defined in Remark 1. The process E is in $\mathcal{E}^{\mathrm{def}!}$. Its encoding is

$$\{\!\{E\}\!\} \equiv ((\overline{\sigma}.\mathbf{0}|!(\sigma.a.\overline{\sigma}.\mathbf{0})) \setminus S) \setminus \{a\}$$

Now, correctly, we obtain that E performs a τ transitions, then it terminates.

The constants X and Y of Remark 1 do not belong to $\mathcal{E}^{\mathrm{def}!}$. In fact in order to translate X we would need a correct translation of Y, and this is not possible without a correct translation of X, i.e., we enter in a loop.

4.2 *P_BNDC* and Definitions

Let $Act = L \cup H \cup \{\tau\}$ as defined in Section 2.1. Let S be a new set of (synchronization) actions such that $S \cap Act = \emptyset$ and $\overline{S} = S$, i.e., S is closed with respect to the complementation operation. In what follows we consider as set of actions $Act = L \cup H \cup \{\tau\} \cup S$. Moreover, we require that if f is a relabelling function, then $f(S) \subseteq S \cup \{\tau\}$. As previously observed the actions of S do not represent 'real' actions, but they are only instrumental for the encoding. However, it is convenient to assimilate them to low level actions in dealing with our security notions. Therefore, the high level attacker cannot perform them and the low level user can observe them. In this way we can treat in a compositional way also processes in which these actions occur. In particular, we extend the concept of weak bisimulation on low actions considering the actions in S as the actions in L. With a slight abuse of notation from now on we say that two processes $E, F \in \mathcal{E}^{\text{def!}}$ are *weakly bisimilar on low*, denoted by $E \approx_l F$, if there exists a binary relation $\mathcal{R} \subseteq \mathcal{E}^{\text{def!}} \times \mathcal{E}^{\text{def!}}$ such that if $(E, F) \in \mathcal{R}$, then for all $a \in L \cup S \cup \{\tau\}$,

- if $E \xrightarrow{a} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;
- if $F \xrightarrow{a} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow} E'$ and $(E', F') \in \mathcal{R}$.

Clearly $\approx_l$ is still the largest weak bisimulation on low actions and it is an equivalence relation. Moreover it is still true that $E \approx_l F$ iff $E \setminus H \approx F \setminus H$.

Using this definition of $\approx_l$ the notions of *BNDC* and *P_BNDC* can be consistently transposed. Notice that using these extended definitions Theorem 1 and Theorem 2 continue to hold. As far as Lemma 3 is concerned some trivial changes are necessary. In particular, let $\mathcal{E}^{\text{def!}}_{HS}$ be the set of all processes constructed over $H \cup S \cup \{\tau\}$. Similarly, let $\mathcal{E}^{\text{def!}}_{LS}$ be the set of all processes constructed over $L \cup S \cup \{\tau\}$ and $\mathcal{E}^{\text{def!}}_{HL}$ be the set of all processes constructed over $L \cup H \cup \{\tau\}$. In case 1. of Lemma 3 it is necessary to consider constant-free processes in $\mathcal{E}^{\text{def!}}_{LS}$. Similarly in case 2. the processes have to be constant-free and to belong to $\mathcal{E}^{\text{def!}}_{H}$. In case 6. the actions a_i's can range over $L \cup S \cup \{\tau\}$. Moreover, from Theorem 4 we immediately get the following result.

Corollary 3. *Let $Z_1, \ldots, Z_n$ be n constants defined as $Z_i \stackrel{\text{def}}{=} D_i$, for $i = 1, \ldots, n$. If for all $i = 1, \ldots, n$ it holds $const(D_i) \subseteq \{Z_1, \ldots, Z_n\}$ and $[\![Z_i]\!] \in P\_BNDC$, then then all the Z_i's are P_BNDC.*

4.3 Extension of the Proof System to Processes with Definitions

In order to deal with the language extended with the action in S and with the constant definitions we have to modify some of the rules of the proof system described in Section 3.2 and to add new rules to deal with constant definitions. In particular, we change the rules (Low) and $(Choice)$ by considering $L \cup S$ instead of L and by adding "E is constant-free" to the rules $(High)$ and (Low).

Then we add the following rules to deal with constant definitions

$$\frac{}{E \setminus S \in \mathcal{HP}[\emptyset]}\ E \in \mathcal{E}^{\mathrm{def!}}_{HS},\ E \text{ is constant-free} \quad (High2)$$

$$\frac{[\![X_i]\!] \in \mathcal{HP}[A]}{X_i \in \mathcal{HP}[A]}\ (X_i \stackrel{\mathrm{def}}{=} D_i)_{i=1}^{n} \quad (Const)$$

where we recall that $[\![X_i]\!]$ is a constant-free process.

With a slight abuse of notation we now call $P\_BNDC - System$ this modification/extension of the proof system presented in Section 3.2.

Example 7. Let us consider the channel as defined in [4] (which was derived from [29]).

$$C = in_0.(\overline{out}_0.C + \tau.C) + in_1.(\overline{out}_1.C + \tau.C)$$

Its encoding is

$$[\![C]\!] \equiv (\overline{\sigma}.\mathbf{0} \mid\ !(\sigma.(in_0.(\overline{out_0}.\overline{\sigma}.\mathbf{0} + \tau.\overline{\sigma}.\mathbf{0}) + in_1.(\overline{out_1}.\overline{\sigma}.\mathbf{0} + \tau.\overline{\sigma}.\mathbf{0})))) \setminus S$$

It is easy to see that we can derive $C \in \mathcal{HP}[\emptyset]$ in our extended proof system. Notice that the process CH described in Example 1 is exactly the process we obtain after a τ transition of $[\![C]\!]$. Written in this way, the channel behavior is now certainly clearer. The example shows the advantages of using constant definitions with respect to replication.

The correctness of $P\_BNDC - System$ immediately follows from Corollary 3.

Corollary 4. *Let $E \in \mathcal{E}^{\mathrm{def!}}$ be a process. If there exists a proof of $E \in \mathcal{HP}[\emptyset]$ in $P\_BNDC - System$, then E is P_BNDC.*

It is possible to add to the proof system the following derived rules which allow in some cases to shorten the derivations in which constant definitions are involved

$$\frac{(D_i[\overline{\sigma_1}.\mathbf{0}/X_1, \ldots, \overline{\sigma_n}.\mathbf{0}/X_n])_{i=1}^{n} \in \mathcal{HP}[A]}{X_i \in \mathcal{HP}[A]}\ (X_i \stackrel{\mathrm{def}}{=} D_i)_{i=1}^{n} \quad (ConstDer)$$

$$\frac{\{\!\{E\}\!\} \in \mathcal{HP}[A]}{E \in \mathcal{HP}[A]}\ (Trans)$$

The correctness of this derived rule $(ConstDer)$ is a consequence of the following result.

Lemma 7. *Let $Z_1, \ldots, Z_n$ be n constants defined as $Z_i \stackrel{\mathrm{def}}{=} D_i$, for $i = 1, \ldots, n$. If for all $i = 1, \ldots, n$ it holds $const(D_i) \subseteq \{Z_1, \ldots, Z_n\}$ and*

$$D_i[\overline{\sigma_1}.\mathbf{0}/Z_1, \ldots, \overline{\sigma_n}.\mathbf{0}/Z_n] \in P\_BNDC,$$

then all the Z_i's are P_BNDC.

Proof. By Theorem 4 we have that Z_i is weak bisimilar to

$$[\![Z_i]\!] \equiv (\overline{\sigma_i}.\mathbf{0}|\widehat{Z}_1|\ldots|\widehat{Z}_n) \setminus S.$$

Hence, if we prove that all the $\widehat{Z}_j$'s are P_BNDC, by Lemma 3, we get the thesis. We have that

$$\widehat{Z}_j \equiv !(\sigma_j.D_j[\overline{\sigma_1}.\mathbf{0}/Z_1,\ldots,\overline{\sigma_n}.\mathbf{0}/Z_n]).$$

By hypothesis $D_j[\overline{\sigma_1}.\mathbf{0}/Z_1,\ldots,\overline{\sigma_n}.\mathbf{0}/Z_n]$ is P_BNDC_S, hence applying Lemma 3 we obtain that $\widehat{Z}_j$ is P_BNDC. □

As far as the rule $(Trans)$ is concerned, its correctness is an immediate consequences of Corollary 2.

Example 8. Let X and Y be defined as

$$\begin{cases} X \stackrel{\text{def}}{=} h.Y + \tau.Y \\ Y \stackrel{\text{def}}{=} l.Y \end{cases}$$

By applying rule $(ConstDer)$ it is immediate to prove that X and Y are $\mathcal{HP}[\emptyset]$.

Let Z be defined as

$$Z \stackrel{\text{def}}{=} h.Z$$

Since $[\![Z]\!] \in \mathcal{E}_H^{\text{def}!}$, by rule $(Const)$ and rule $(SHigh)$ we get that Z is $\mathcal{HP}[\emptyset]$.

5 Conclusions

In this paper we study the class of P_BNDC processes written in a variant of SPA language where recursive processes are defined by means of replications instead of constant definitions. The modified language is slightly less powerful than the original one, but the loss of expressive power is largely compensated by the compositionality result we obtain. In fact, we proved that the class of P_BNDC processes is compositional with respect to replication. This result allows us to define a proof system which provides a very efficient technique for the stepwise development and the verification of recursively defined P_BNDC processes. We also identify a class of constants definitions which can be safely added to our language and treated by an extended proof system.

As already noticed in [3], there are many other approaches to the verification of information flow properties. In the literature we found only another example of a proof system for security proposed by Martinelli in [23] which deals only with finite processes. For instance, there are verification techniques for information flow security which are based on types (see, e.g., [37, 35, 19, 7]) and control flow analysis (see, e.g., [2, 8]). However, most of them are concerned with different models, e.g., trace semantics [20, 21, 11, 26].

References

1. M. Abadi and L. Lamport. Conjoining Specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 17(3):507–535, May 1995.
2. C. Bodei, P. Degano, F. Nielson, and H. Nielson. Static Analysis for the π-calculus with Applications to Security. *Information and Computation*, 168(1):68–92, 2001.
3. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. A Proof System for Information Flow Security. In M. Leuschel, editor, *Proc. of Int. Workshop on Logic Based Program Development and Transformation*, LNCS. Springer-Verlag, 2002. To appear.
4. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Transforming Processes to Ensure and Check Information Flow Security. In H. Kirchner and C. Ringeissen, editors, *Int. Conference on Algebraic Methodology and Software Technology (AMAST'02)*, volume 2422 of *LNCS*, pages 271–286. Springer-Verlag, 2002.
5. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Bisimulation and Unwinding for Verifying Possibilistic Security Properties. In L. D. Zuck, P. C. Attie, A. Cortesi, and S. Mukhopadhyay, editors, *Proc. of Int. Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'03)*, volume 2575 of *LNCS*, pages 223–237. Springer-Verlag, 2003.
6. A. Bossi, D. Macedonio, C. Piazza, and S. Rossi. P_BNDC and Replication. Technical Report CS-2003-??, Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy, 2003.
7. G. Boudol and I. Castellani. Noninterference for Concurrent Programs. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Proc. of Int. Colloquium on on Automata, Languages and Programming (ICALP'01)*, volume 2076 of *LNCS*, pages 382–395. Springer-Verlag, 2001.
8. C. Braghin, A. Cortesi, and R. Focardi. Control Flow Analysis of Mobile Ambients with Security Boundaries. In *Proc. Int. Conference on Formal Methods for Open Object-Based Distributed Systems (IFIPM'02)*, pages 197–212. Kluwer, 2002.
9. M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication Interference in Mobile Boxed Ambients. In M. Agrawal and A. Seth, editors, *Proc. of Int. Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'02)*, volume 2556 of *LNCS*, pages 71–84. Springer-Verlag, 2002.
10. N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. Recursive Definitions in Channel Based Calculi. In *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP'03)*. LNCS, 2003.
11. McCullough. D. A Hookup Theorem for Multilevel Security. *IEEE Transactions on Software Engineering*, 16(6):563–568, 1990.
12. P. Degano, F. Gadducci, and C. Priami. A Causal Semantics for CCS via Rewriting Logic. *Theoretical Computer Science*, 275(1-2):259–282, 2002.
13. R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*. Springer-Verlag, 2001.
14. R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 744–755. Springer-Verlag, 2000.
15. R. Focardi, C. Piazza, and S. Rossi. Proof Methods for Bisimulation based Information Flow Security. In A. Cortesi, editor, *Proc. of Int. Workshop on Verification, Model Checking and Abstract Interpretation*, volume 2294 of *LNCS*, pages 16–31. Springer-Verlag, 2002.

16. R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 307–319. IEEE Comp. Soc. Press, 2002.
17. S. N. Foley. A Universal Theory of Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 116–122. IEEE Comp. Soc. Press, 1987.
18. J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Comp. Soc. Press, 1982.
19. M. Hennessy and J. Riely. Information Flow vs. Resource Access in the Asynchronous Pi-calculus. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 24(5):566–591, 2002.
20. H. Mantel. Possibilistic Definitions of Security - An Assebly Kit -. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 185–199. IEEE Comp. Soc. Press, 2000.
21. H. Mantel. Unwinding Possibilistic Security Properties. In *Proc. of the European Symposium on Research in Computer Security*, volume 2895 of *LNCS*, pages 238–254. Springer-Verlag, 2000.
22. H. Mantel. On the Composition of Secure Systems. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 88–101. IEEE Comp. Soc. Press, 2002.
23. F. Martinelli. Partial Model Checking and Theorem Proving for Ensuring Security Properties. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'98)*, pages 44–52. IEEE Comp. Soc. Press, 1998.
24. D. McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 161–166. IEEE Comp. Soc. Press, 1987.
25. J. McLean. Security Models and Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 180–187. IEEE Comp. Soc. Press, 1990.
26. J. McLean. A General Theory of Composition for Trace Sets Closed under Selective Interleaving Functions. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 79–93. IEEE Comp. Soc. Press, 1994.
27. J. McLean. A General Theory of Composition for a Class of "Possibilistic" Security Properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, 1996.
28. J. K. Millen. Unwinding Forward Correctability. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'94)*, pages 2–10. IEEE Comp. Soc. Press, 1994.
29. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
30. R. Milner. The Polyadic Pi-calculus: a tutorial. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993.
31. C. O'Halloran. A Calculus of Information Flow. In *Proc. of the European Symposium on Research in Security and Privacy*, pages 180–187. AFCET, 1990.
32. J. Rushby. Noninterference, Transitivity, and Channel-Control Security Policies. Technical Report CSL-92-02, SRI International, 1992.
33. P. Ryan and S. Schneider. Process Algebra and Non-Interference. *Journal of Computer Security*, 9(1/2):75–103, 2001.
34. A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communication*, 21(1):5–19, 2003.
35. A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 200–215. IEEE Comp. Soc. Press, 2000.

36. D. Sangiorgi and D. Walker. *The π-calculus*. Cambridge University Press, 2001.
37. G. Smith and D. M. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'98)*, pages 355–364. ACM Press, 1998.
38. J. Widom, D. Gries, and F. B. Schneider. Trace-based Network Proof Systems: Expressiveness and Completeness. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 14(3):396–416, 1992.
39. A. Zakinthinos and E. S. Lee. A General Theory of Security Properties. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 74–102. IEEE Comp. Soc. Press, 1997.

Appendix

Proof of Theorem 4

We prove a simpler version of Theorem 4 in the case of a single constant. The general case can be easily proved by extending the following results. So in the rest of this section we assume that the set of constants is $\{Z\}$ and that the environment defines $Z \stackrel{\text{def}}{=} D_0$. In this case $S = \{\bar{\sigma}, \sigma\}$, and Definition 7 reduces to $[\![D]\!] \equiv (\, D[\bar{\sigma}.\mathbf{0}/Z] \,|\, !(\sigma.\mathbf{0}.D_0[\bar{\sigma}.\mathbf{0}/Z]\,)\backslash S$ if Z appears in D, otherwise $[\![D]\!] \equiv D$.

First we prove some results that are instrumental to prove Theorem 4.

Lemma 8. *The rule*

$$\frac{D[\bar{\sigma}.\mathbf{0}/Z] \stackrel{a}{\rightarrow} D'[\bar{\sigma}.\mathbf{0}/Z]}{[\![D]\!] \stackrel{a}{\rightarrow} [\![D']\!]}\ (a \notin S)$$

is derivable by using the rules in Figure 1, where $D \in \mathcal{E}^{\text{def}}$.

Moreover, if $a \notin S \cup \{\tau\}$ then the transition $[\![D]\!] \stackrel{a}{\rightarrow} [\![D']\!]$ can be deduced only form $D[\bar{\sigma}.\mathbf{0}/Z] \stackrel{a}{\rightarrow} D'[\bar{\sigma}.\mathbf{0}/Z]$.

Proof. To derive the rule it is sufficient to use Parallel and Restriction:

$$\frac{\dfrac{\dfrac{D[\bar{\sigma}.\mathbf{0}/Z] \stackrel{a}{\rightarrow} D'[\bar{\sigma}.\mathbf{0}/Z]}{D[\bar{\sigma}.\mathbf{0}/Z] \,|\, \widehat{Z} \stackrel{a}{\rightarrow} D'[\bar{\sigma}.\mathbf{0}/Z] \,|\, \widehat{Z}}}{D[\bar{\sigma}.\mathbf{0}/Z] \,|\, \widehat{Z} \setminus S \stackrel{a}{\rightarrow} D'[\bar{\sigma}.\mathbf{0}/Z] \,|\, \widehat{Z} \setminus S}}{[\![D]\!] \stackrel{a}{\rightarrow} [\![D']\!]}$$

On the other hand, if $a \notin S \cup \{\tau\}$ the only way to to obtain $[\![D]\!] \stackrel{a}{\rightarrow} [\![D']\!]$ is by starting from $D[\bar{\sigma}.\mathbf{0}/Z] \stackrel{a}{\rightarrow} D'[\bar{\sigma}.\mathbf{0}/Z]$ and by applying Parallel and Restriction. □

Lemma 9. *Given $D_1, D_2 \in \mathcal{E}^{\text{def}}$, then $[\![D_1 + D_2]\!] \sim [\![D_1]\!] + [\![D_2]\!]$.*

Proof. We will prove that $\mathcal{R} = \sim \cup \{ (([\![D_1 + D_2]\!], [\![D_1]\!] + [\![D_2]\!]) \ : \ D_1, D_2 \in \mathcal{E}^{\mathrm{def}} \}$ is a strong bisimulation. Consider $([\![D_1 + D_2]\!], [\![D_1]\!] + [\![D_2]\!]) \in \mathcal{R}$. Here we handle only the case in which $const(D_1) = const(D_2) = \{Z\}$, the other cases ($const(D_1) = \emptyset$ or $const(D_2) = \emptyset$) are handled in similar way.

First suppose that $[\![D_1 + D_2]\!] \stackrel{a}{\rightarrow} E$, this means that

$$((D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z]) \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]) \setminus S \stackrel{a}{\rightarrow} E$$

We distinguish two cases: synchronization and no synchronization.

- Synchronization ($a = \tau$). Then the parallel can only synchronize on σ and $\bar{\sigma}$, in fact the process $!\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]$ can only perform the action σ. So we have

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1}{D_1[\bar{\sigma}.\mathbf{0}/Z] \stackrel{\bar{\sigma}}{\rightarrow} E'}}{D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z] \stackrel{\bar{\sigma}}{\rightarrow} E'} \quad \dfrac{\Gamma_2}{\widehat{Z} \stackrel{\sigma}{\rightarrow} E''}}{(D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z]) \mid \widehat{Z} \stackrel{\tau}{\rightarrow} (E'|E'')}}{[\![D_1 + D_2]\!] \stackrel{\tau}{\rightarrow} (E'|E'') \setminus S}$$

  and then we can derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1}{D_1[\bar{\sigma}.\mathbf{0}/Z] \stackrel{\bar{\sigma}}{\rightarrow} E'} \quad \dfrac{\Gamma_2}{\widehat{Z} \stackrel{\sigma}{\rightarrow} E''}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid \widehat{Z} \stackrel{\tau}{\rightarrow} E'|E''}}{[\![D_1]\!] \stackrel{\tau}{\rightarrow} (E'|E'') \setminus S}}{[\![D_1]\!] + [\![D_2]\!] \stackrel{\tau}{\rightarrow} (E'|E'') \setminus S}$$

  with $((E'|E'') \setminus S, (E'|E'') \setminus S) \in \mathcal{R}$.
- No synchronization. Then $a \neq \bar{\sigma}, \sigma$ and it is performed by $D_1[\bar{\sigma}.\mathbf{0}/Z]$ or $D_2[\bar{\sigma}.\mathbf{0}/Z]$; without loss of generality we can assume that a is performed by $D_1[\bar{\sigma}.\mathbf{0}/Z]$, so we have

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \stackrel{a}{\rightarrow} E'|\widehat{Z}}}{(D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z]) \stackrel{a}{\rightarrow} E'|\widehat{Z}}}{(D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z]) \mid \widehat{Z} \stackrel{a}{\rightarrow} E'|\widehat{Z}}}{[\![D_1 + D_2]\!] \stackrel{a}{\rightarrow} (E'|\widehat{Z}) \setminus S}$$

  Then we can derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \stackrel{a}{\rightarrow} E'|\widehat{Z}}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid \widehat{Z} \stackrel{a}{\rightarrow} E'|\widehat{Z}}}{[\![D_1]\!] \stackrel{a}{\rightarrow} E'|\widehat{Z}(E'|\widehat{Z}) \setminus S}}{[\![D_1]\!] + [\![D_2]\!] \stackrel{a}{\rightarrow} (E'|\widehat{Z}) \setminus S}$$

  with $((E'|\widehat{Z}) \setminus S, (E'|\widehat{Z}) \setminus S) \in \mathcal{R}$.

On the other hand, suppose that $[\![D_1]\!] + [\![D_2]\!] \xrightarrow{a} D$. This means that

$$(\, D_1[\bar{\sigma}.\mathbf{0}/Z] \,|\, !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \,) \setminus S + (\, D_2[\bar{\sigma}.\mathbf{0}/Z] \,|\, !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \,) \setminus S \xrightarrow{a} E$$

Then the last rule applied in the derivation is Sum; without loss of generality we can assume that $(\, D_1[\bar{\sigma}.\mathbf{0}/Z] \,|\, !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \,) \setminus S \xrightarrow{a} D$. Yet again, we distinguish two cases: synchronization and no synchronization.

- Synchronization. ($a = \tau$). Then the parallel can only synchronize on σ and $\bar{\sigma}$. We have

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{\sigma}} E'} \quad \dfrac{\Gamma_2}{\widehat{Z} \xrightarrow{\sigma} E''}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \,|\, \widehat{Z} \xrightarrow{\tau} E'|E''}}{[\![D_1]\!] \xrightarrow{\tau} (E'|E'') \setminus S}}{[\![D_1]\!] + [\![D_2]\!] \xrightarrow{\tau} (E'|E'') \setminus S}$$

  Then we can derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{\sigma}} E'}}{D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{\sigma}} E'} \quad \dfrac{\Gamma_2}{\widehat{Z} \xrightarrow{\sigma} E''}}{(D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z]) \,|\, \widehat{Z} \xrightarrow{\tau} (E'|E'')}}{[\![D_1 + D_2]\!] \xrightarrow{\tau} (E'|E'') \setminus S}$$

  with $(\, (E'|E'') \setminus S, (E'|E'') \setminus S \,) \in \mathcal{R}$.
- No synchronization. Then $a \neq \bar{\sigma}, \sigma$ and it is performed by $D_1[\bar{\sigma}.\mathbf{0}/Z]$, so we have

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{a} E'|\widehat{Z}}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \,|\, \widehat{Z} \xrightarrow{a} E'|\widehat{Z}}}{[\![D_1]\!] \xrightarrow{a} (E'|\widehat{Z}) \setminus S}}{[\![D_1]\!] + [\![D_2]\!] \xrightarrow{a} (\, E'|\widehat{Z} \,) \setminus S}$$

  Then we can derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{a} E'|\widehat{Z}}}{(\, D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z] \,) \xrightarrow{a} E'|\widehat{Z}}}{(\, D_1[\bar{\sigma}.\mathbf{0}/Z] + D_2[\bar{\sigma}.\mathbf{0}/Z] \,) \,|\, \widehat{Z} \xrightarrow{a} E'|\widehat{Z}}}{[\![D_1 + D_2]\!] \xrightarrow{a} (\, E'|\widehat{Z} \,) \setminus S}$$

  with $(\, (E'|\widehat{Z}) \setminus S, (\, E'|\widehat{Z} \,) \setminus S \,) \in \mathcal{R}$

We can conclude that $\mathcal{R}$ is a strong bisimulation, hence $[\![D_1 + D_2]\!] \sim [\![D_1]\!] + [\![D_2]\!]$ for any $D_1, D_2 \in \mathcal{E}^{\mathrm{def}}$. □

Lemma 10. *Given* $D_1, D_2 \in \mathcal{E}^{\mathrm{def}}$, *then* $[\![D_1 \mid D_2]\!] \sim [\![D_1]\!] \mid [\![D_2]\!]$.

Proof. We will prove that $\mathcal{R} = \{\, ([\![D_1 \mid D_2]\!], [\![D_1]\!] \mid [\![D_2]\!]) \;:\; D_1, D_2 \in \mathcal{E}^{\mathrm{def}} \,\}$ is a strong bisimulation. Consider $([\![D_1 \mid D_2]\!], [\![D_1]\!] \mid [\![D_2]\!]) \in \mathcal{R}$. Here we handle only the case in which $const(D_1) = const(D_2) = \{Z\}$, the other cases ($const(D_1) = \emptyset$ or $const(D_2) = \emptyset$) are handled in similar way.

First suppose that $[\![D_1 \mid D_2]\!] \xrightarrow{a} E$, this means that

$$(\, D_1[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \,) \setminus S \xrightarrow{a} E$$

We distinguish three cases.

- Synchronization ($a = \tau$) between $D_1[\bar{\sigma}.\mathbf{0}/Z]$ and $D_2[\bar{\sigma}.\mathbf{0}/Z]$. These two processes can synchronize on $b, \bar{b} \notin S$, so we have[2]

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{b} D_1'[\bar{\sigma}.\mathbf{0}/Z]} \quad \dfrac{\Gamma_2}{D_2[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{b}} D_2'[\bar{\sigma}.\mathbf{0}/Z]}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\tau} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_2'[\bar{\sigma}.\mathbf{0}/Z]}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \mid \widehat{Z} \xrightarrow{\tau} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_2'[\bar{\sigma}.\mathbf{0}/Z] \mid \widehat{Z}}}{(\, D_1[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \mid \widehat{Z} \,) \setminus S \xrightarrow{\tau} (\, D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_2'[\bar{\sigma}.\mathbf{0}/Z] \mid \widehat{Z} \,) \setminus S}}{[\![D_1 \mid D_2]\!] \xrightarrow{\tau} [\![D_1' \mid D_2']\!]}$$

  Then we can derive

$$\dfrac{\dfrac{\dfrac{\Gamma_1}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{b} D_1'[\bar{\sigma}.\mathbf{0}/Z]}}{[\![D_1]\!] \xrightarrow{b} [\![D_1']\!]}\ (b \notin S) \quad \dfrac{\dfrac{\Gamma_2}{D_2[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{b}} D_2'[\bar{\sigma}.\mathbf{0}/Z]}}{[\![D_2]\!] \xrightarrow{\bar{b}} [\![D_2']\!]}\ (\bar{b} \notin S)}{[\![D_1]\!] \mid [\![D_2]\!] \xrightarrow{\tau} [\![D_1']\!] \mid [\![D_2']\!]}$$

  whit $(\, [\![D_1' \mid D_2']\!] \,,\, [\![D_1']\!] \mid [\![D_2']\!] \,) \in \mathcal{R}$.
- Synchronization ($a = \tau$) between $D_1[\bar{\sigma}.\mathbf{0}/Z]$ and $!\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]$ (the case $D_2[\bar{\sigma}.\mathbf{0}/Z]$ and $!\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]$ is analogous) . Then the parallel can only synchronize on σ and $\bar{\sigma}$, in fact the process $!\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]$ can only perform the action σ. So we have[3]

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{\sigma}} D_1'[\bar{\sigma}.\mathbf{0}/Z]} \quad \dfrac{\dfrac{}{\sigma.D_0[\bar{\sigma}.\mathbf{0}] \xrightarrow{\sigma} D_0[\bar{\sigma}.\mathbf{0}/Z]}}{!\sigma.D_0[\bar{\sigma}.\mathbf{0}] \xrightarrow{\sigma} D_0[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}]}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}] \xrightarrow{\tau} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_0[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}]}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}] \xrightarrow{\tau} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \mid D_0[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}]}}{[\![D_1 \mid D_2]\!] \xrightarrow{\tau} [\![D_1' \mid D_2']\!]}$$

[2] Note that given $D \in \mathcal{E}^{\mathrm{def}}$, if $D[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{a} E$ then $D' = E[Z/\bar{\sigma}.\mathbf{0}] \in \mathcal{E}^{\mathrm{def}}$ and $D'[\bar{\sigma}.\mathbf{0}/Z] = E$.

[3] Note that $\bar{\sigma} \neq \tau$, hence the synchronization rule cannot be used for the replication.

Then we can derive

$$\dfrac{\dfrac{\Gamma}{D_1[\bar\sigma.\mathbf{0}/Z] \xrightarrow{\bar\sigma} D_1'[\bar\sigma.\mathbf{0}/Z]} \quad \dfrac{\dfrac{}{\sigma.D_0[\bar\sigma.\mathbf{0}] \xrightarrow{\sigma} D_0[\bar\sigma.\mathbf{0}/Z]}}{!\sigma.D_0[\bar\sigma.\mathbf{0}] \xrightarrow{\sigma} D_0[\bar\sigma.\mathbf{0}/Z] \mid !\sigma.D_0[\bar\sigma.\mathbf{0}]}}{D_1[\bar\sigma.\mathbf{0}/Z] \mid !\sigma.D_0[\bar\sigma.\mathbf{0}] \xrightarrow{\tau} D_1'[\bar\sigma.\mathbf{0}/Z] \mid D_0[\bar\sigma.\mathbf{0}/Z] \mid !\sigma.D_0[\bar\sigma.\mathbf{0}]}}{[\![D_1]\!] \xrightarrow{\tau} [\![D_1' \mid D_0]\!]}}{[\![D_1]\!] \mid [\![D_2]\!] \xrightarrow{\tau} [\![D_1' \mid D_0]\!] \mid [\![D_2]\!]}$$

with $([\![D_1' \mid D_0 \mid D_2]\!], [\![D_1' \mid D_0]\!] \mid [\![D_2]\!]) \in \mathcal{R}$.

– No synchronization. Then $a \notin S$ and it is performed by $D_1[\bar\sigma.\mathbf{0}/Z]$ or $D_2[\bar\sigma.\mathbf{0}/Z]$; without loss of generality we assume that a is performed by $D_1[\bar\sigma.\mathbf{0}/Z]$, so we have

$$\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar\sigma.\mathbf{0}/Z] \xrightarrow{a} D_1'[\bar\sigma.\mathbf{0}/Z]}}{D_1[\bar\sigma.\mathbf{0}/Z] \mid D_2[\bar\sigma.\mathbf{0}/Z] \xrightarrow{a} D_1'[\bar\sigma.\mathbf{0}/Z] \mid D_2[\bar\sigma.\mathbf{0}/Z]}}{[\![D_1 \mid D_2]\!] \xrightarrow{a} [\![D_1' \mid D_2]\!]} \ (a \notin S)$$

Then we can derive

$$\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar\sigma.\mathbf{0}/Z] \xrightarrow{a} D_1'[\bar\sigma.\mathbf{0}/Z]}}{[\![D_1]\!] \xrightarrow{a} [\![D_1']\!]}}{[\![D_1]\!] \mid [\![D_2]\!] \xrightarrow{a} [\![D_1']\!] \mid [\![D_2]\!]} \ (a \notin S)$$

with $([\![D_1' \mid D_2]\!], [\![D_1']\!] \mid [\![D_2]\!]) \in \mathcal{R}$

On the other hand suppose that $[\![D_1]\!] \mid [\![D_2]\!] \xrightarrow{a} E$, this means that

$$(D_1[\bar\sigma.\mathbf{0}/Z] \mid !\sigma.D_0[\bar\sigma.\mathbf{0}/Z]) \setminus S \mid (D_2[\bar\sigma.\mathbf{0}/Z] \mid !\sigma.D_0[\bar\sigma.\mathbf{0}/Z]) \setminus S \xrightarrow{a} E$$

We distinguish three cases

– Synchronization ($a = \tau$) between $[\![D_1]\!]$ and $[\![D_2]\!]$, hence they synchronize on $b, \bar b \notin S \cup \{\tau\}$ and

$$\dfrac{\dfrac{\dfrac{\Gamma_1}{D_1[\bar\sigma.\mathbf{0}/Z] \xrightarrow{b} D_1'[\bar\sigma.\mathbf{0}/Z]} \ (b \notin S)}{[\![D_1]\!] \xrightarrow{b} [\![D_1']\!]} \quad \dfrac{\dfrac{\Gamma_2}{D_2[\bar\sigma.\mathbf{0}/Z] \xrightarrow{\bar b} D_2'[\bar\sigma.\mathbf{0}/Z]} \ (b \notin S)}{[\![D_2]\!] \xrightarrow{\bar b} [\![D_2']\!]}}{[\![D_1]\!] \mid [\![D_2]\!] \xrightarrow{\tau} [\![D_1']\!] \mid [\![D_2']\!]}$$

Then we can derive

$$\dfrac{\dfrac{\Gamma_1}{D_1[\bar\sigma.\mathbf{0}/Z] \xrightarrow{b} D_1'[\bar\sigma.\mathbf{0}/Z]} \quad \dfrac{\Gamma_2}{D_2[\bar\sigma.\mathbf{0}/Z] \xrightarrow{\bar b} D_2'[\bar\sigma.\mathbf{0}/Z]}}{D_1[\bar\sigma.\mathbf{0}/Z] \mid D_2[\bar\sigma.\mathbf{0}/Z] \xrightarrow{\tau} D_1'[\bar\sigma.\mathbf{0}/Z] \mid D_2'[\bar\sigma.\mathbf{0}/Z]} \ (b \notin S)$$

$$[\![D_1 \mid D_2]\!] \xrightarrow{\tau} [\![D_1' \mid D_2']\!]$$

with $(\llbracket D_1' \mid D_2' \rrbracket, \llbracket D_1' \rrbracket \mid \llbracket D_2' \rrbracket) \in \mathcal{R}$.

– No synchronization, without loss of generality we assume that a is performed by $\llbracket D_1 \rrbracket$. We distinguish two subcases
  - a is performed by $D_1[\bar{\sigma}.\mathbf{0}/Z]$. Then we have

$$\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{a} D_1'[\bar{\sigma}.\mathbf{0}/Z]}}{\llbracket D_1 \rrbracket \xrightarrow{a} \llbracket D_1' \rrbracket}}{\llbracket D_1 \rrbracket \mid \llbracket D_2 \rrbracket \xrightarrow{a} \llbracket D_1' \rrbracket \mid \llbracket D_2 \rrbracket} \; (a \notin S)$$

    Hence we can deduce

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{a} D_1'[\bar{\sigma}.\mathbf{0}/Z]}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{a} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z]}}{\llbracket D_1 \mid D_2 \rrbracket \xrightarrow{a} \llbracket D_1' \mid D_2 \rrbracket}} \; (a \notin S)$$

    with $(\llbracket D_1' \mid D_2 \rrbracket, \llbracket D_1' \rrbracket \mid \llbracket D_2 \rrbracket) \in \mathcal{R}$.
  - $a = \tau$ and it is produced by the synchronization between $D_1[\bar{\sigma}.\mathbf{0}/Z]$ and $!\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]$. Then

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{b} D_1'[\bar{\sigma}.\mathbf{0}/Z]} \quad \dfrac{\dfrac{}{\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{b}} D_0[\bar{\sigma}.\mathbf{0}/Z]}}{!\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{b}} D_0[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\tau} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_0[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]}}{\llbracket D_1 \rrbracket \xrightarrow{\tau} \llbracket D_1' \mid D_0 \rrbracket}}{\llbracket D_1 \rrbracket \mid \llbracket D_2 \rrbracket \xrightarrow{\tau} \llbracket D_1' \mid D_0 \rrbracket \mid \llbracket D_2 \rrbracket}$$

    in this case we deduce

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{D_1[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{b} D_1'[\bar{\sigma}.\mathbf{0}/Z]} \quad \dfrac{\dfrac{}{\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{b}} D_0[\bar{\sigma}.\mathbf{0}/Z]}}{!\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\bar{b}} D_0[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]}}{D_1[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z] \xrightarrow{\tau} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_0[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]}}{D_1[\bar{\sigma}\mathbf{0}/Z] \mid D_2[\bar{\sigma}\mathbf{0}/Z] \mid D_0[\bar{\sigma}\mathbf{0}/Z] \xrightarrow{\tau} D_1'[\bar{\sigma}.\mathbf{0}/Z] \mid D_0[\bar{\sigma}.\mathbf{0}/Z] \mid D_2[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]}}{\llbracket D_1 \mid D_2 \rrbracket \xrightarrow{\tau} \llbracket D_1' \mid D_0 \mid D_2 \rrbracket}$$

    with $(\llbracket D_1' \mid D_0 \mid D_2 \rrbracket, \llbracket D_1' \mid D_0 \rrbracket \mid \llbracket D_2 \rrbracket) \in \mathcal{R}$

Since we have considered all the possible cases, we can conclude that $\mathcal{R}$ is a strong bisimulation, hence $\llbracket D_1 \mid D_2 \rrbracket \sim \llbracket D_1 \rrbracket \mid \llbracket D_2 \rrbracket$ for any $D_1, D_2 \in \mathcal{E}^{\mathrm{def}}$. □

Lemma 11. *Given a constant Z, if $Z \stackrel{\mathrm{def}}{=} D_0$ then $\llbracket Z \rrbracket \sim \tau.\llbracket D_0 \rrbracket$.*

Proof. By definition we have:

$$\begin{aligned}[\![Z]\!] &= (\,\bar{\sigma}.\mathbf{0} \mid !\sigma.[\bar{\sigma}.\mathbf{0}/Z]\,) \setminus S \\ \tau.[\![D_0]\!] &= \tau.(\,(\,D_0[\bar{\sigma}.\mathbf{0}] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]\,) \setminus S\,)\end{aligned}$$

It is easy to see that $[\![Z]\!]$ can perform only a τ action and reduce to the process $(\,(\,D_0[\bar{\sigma}.\mathbf{0}] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]\,) \setminus S\,)$. On the other hand, $\tau.[\![D_0]\!]$ can perform only a τ action and reduce to the same process. We conclude that $[\![Z]\!] \sim \tau.[\![D_0]\!]$. □

Lemma 12. *Given $D \in \mathcal{E}^{\mathrm{def}}$, then $[\![a.D]\!] \sim a.[\![D]\!]$.*

Proof. By definition we have:

$$\begin{aligned}[\![a.D]\!] &= (\,a.D[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]\,) \setminus S \\ a.[\![D]\!] &= a.(\,(\,D[\bar{\sigma}.\mathbf{0}/Z] \mid !\sigma.D_0[\bar{\sigma}.\mathbf{0}/Z]\,) \setminus S\,)\end{aligned}$$

It is easy to see that both the processes can only perform an action a and reduce to $[\![D]\!]$, hence we conclude that $[\![a.D]\!] \sim a.[\![D]\!]$. □

Proof (Lemma 6). It is a consequence of Lemmas 9, 10, 11, and 12. □

Lemma 13. *Given $D \in \mathcal{E}^{\mathrm{def}}$, if $D \xrightarrow{a} D'$ then there exists $E \in \mathcal{E}^{!}$ such that $[\![D]\!] \xrightarrow{a} E$ and $E \sim [\![D']\!]$.*

Proof. We prove the lemma by induction on the structure of D. The basis of induction are the cases in which $D = \mathbf{0}$ or $D = Z$ or $D = a.D_1$.

Case $D = \mathbf{0}$. It is trivial, in fact $[\![\mathbf{0}]\!] = \mathbf{0}$ does not perform any action.

Case $D = Z$. A constant can only perform a τ action, hence $a = \tau$. If $Z \stackrel{\mathrm{def}}{=} D_0$ then $Z \xrightarrow{\tau} D_0$, we conclude that $[\![Z]\!] \xrightarrow{\tau} [\![D_0]\!]$ by Lemma 6.

Case $D = a.D_1$. Then $a.D_1 \xrightarrow{a} D_1$, but also $a.[\![D_1]\!] \xrightarrow{a} [\![D_1]\!]$, but $a.[\![D_1]\!] \sim [\![a.D_1]\!]$ by Lemma 6, hence there exists $E \in \mathcal{E}^{!}$ such that $[\![a.D_1]\!] \xrightarrow{a} E$ and $E \sim [\![D_1]\!]$.

Case $D = D_1 + D_2$. Then $D_1 + D_2 \xrightarrow{a} D'$. Without loss of generality we assume that a is performed by D_1, hence $D_1 + D_2 \xrightarrow{a} D'$. By induction on D_1 there exists $E' \in \mathcal{E}^{!}$ such that $E' \sim [\![D']\!]$ and $[\![D_1]\!] \xrightarrow{a} E'$, hence $[\![D_1]\!] + [\![D_2]\!] \xrightarrow{a} E'$. But $[\![D_1]\!] + [\![D_2]\!] \sim [\![D_1 + D_2]\!]$ by lemma 6, so we conclude that there exists $E \in \mathcal{E}^{!}$ such that $[\![D_1 + D_2]\!] \xrightarrow{a} E$ and $E \sim E' \sim [\![D']\!]$.

Case $D = D_1 \mid D_2$. We distinguish two subcases

- Synchronization ($a = \tau$). Then $D_1 \mid D_2 \xrightarrow{\tau} D_1' \mid D_2'$ is deduced from $D_1 \xrightarrow{b} D_1'$ and $D_2 \xrightarrow{\bar{b}} D_2'$. By induction on D_1, D_2 there exists $E_1', E_2' \in \mathcal{E}^{!}$ such that $E_1' \sim [\![D_1']\!]$, $E_2' \sim [\![D_2']\!]$ and $[\![D_1]\!] \xrightarrow{b} E_1'$, $[\![D_2]\!] \xrightarrow{\bar{b}} E_2'$; hence $[\![D_1]\!] \mid [\![D_2]\!] \xrightarrow{\tau} E_1' \mid E_2'$. But $[\![D_1]\!] \mid [\![D_2]\!] \sim [\![D_1 \mid D_2]\!]$ by Lemma 6, and so there exists $E \in \mathcal{E}^{!}$ such that $[\![D_1 \mid D_2]\!] \xrightarrow{\tau} E$ and $E \sim E_1' \mid E_2' \sim [\![D_1']\!] \mid [\![D_2']\!] \sim [\![D_1' \mid D_2']\!]$, by Lemma 6.

– No synchronization. Without loss of generality we assume that a is performed by D_1, so we have $D_1 \mid D_2 \stackrel{a}{\rightarrow} D_1' \mid D_2$ and $D_1 \stackrel{a}{\rightarrow} D_1'$. By induction on D_1 there exists $E' \in \mathcal{E}^!$ such that $E' \sim [\![D_1']\!]$ and $[\![D_1]\!] \stackrel{a}{\rightarrow} E'$, hence $[\![D_1]\!] \mid [\![D_2]\!] \stackrel{a}{\rightarrow} E' \mid [\![D_2]\!]$. But $[\![D_1]\!] \mid [\![D_2]\!] \sim [\![D_1 \mid D_2]\!]$ by Lemma 6, hence there exists $E \in \mathcal{E}^!$ such that $[\![D_1 \mid D_2]\!] \stackrel{a}{\rightarrow} E$ and $E \sim E' \mid [\![D_2]\!] \sim [\![D_1']\!] \mid [\![D_2]\!] \sim [\![D_1' \mid D_2]\!]$, by Lemma 6.

□

Lemma 14. *Given $D \in \mathcal{E}^{\mathrm{def}}$, if $[\![D]\!] \stackrel{a}{\rightarrow} E$ then there exists $D' \in \mathcal{E}^{\mathrm{def}}$ such that $D \stackrel{a}{\rightarrow} D'$ and $[\![D']\!] \sim E$.*

Proof. We prove the lemma by induction on the structure of D. The basis of induction are the cases in which $D = \mathbf{0}$ or $D = Z$ or $D = a.D_1$.

Case $D = \mathbf{0}$. It is trivial since $[\![\mathbf{0}]\!] = \mathbf{0}$ does not perform any action.

Case $D = Z$. Lemma 6 says that $[\![Z]\!] \sim \tau.[\![D_0]\!]$, provided that $Z \stackrel{\mathrm{def}}{=} D_0$; hence $[\![Z]\!]$ can only perform a τ action and moreover $[\![Z]\!] \stackrel{\tau}{\rightarrow} E$ with $E \sim [\![D_0]\!]$. We conclude that $Z \stackrel{\tau}{\rightarrow} D_0$ by applying the Constant rule.

Case $D = a.D_1$. Lemma 6 says that $[\![a.D_1]\!] \sim a.[\![D_1]\!]$, hence $[\![a.D_1]\!]$ can only perform the action a and moreover $[\![a.D_1]\!] \stackrel{a}{\rightarrow} E$ with $E \sim [\![D_1]\!]$. We conclude that $a.D_1 \stackrel{a}{\rightarrow} D_1$ by applying the Prefix rule.

Case $D = D_1 + D_2$. Then $[\![D_1 + D_2]\!] \stackrel{a}{\rightarrow} E$. Lemma 6 says that $[\![D_1 + D_2]\!] \sim [\![D_1]\!] + [\![D_2]\!]$, hence there exists $E' \in \mathcal{E}^!$ such that $[\![D_1]\!] + [\![D_2]\!] \stackrel{a}{\rightarrow} E'$ and $E' \sim E$. Without loss of generality we assume that this is derived from $[\![D_1]\!] \stackrel{a}{\rightarrow} E'$ by applying Sum. By induction on D_1 there exists $D' \in \mathcal{E}^{\mathrm{def}}$ such that $D_1 \stackrel{a}{\rightarrow} D'$ and $[\![D]\!] \sim E'$; we conclude that $D_1 + D_2 \stackrel{a}{\rightarrow} D'$ and $[\![D]\!] \sim E' \sim E$.

Case $D = D_1 \mid D_2$. Then $[\![D_1 \mid D_2]\!] \stackrel{a}{\rightarrow} E$. Lemma 6 says that $[\![D_1 \mid D_2]\!] \sim [\![D_1]\!] \mid [\![D_2]\!]$, hence there exists $E' \in \mathcal{E}^!$ such that $[\![D_1]\!] \mid [\![D_2]\!] \stackrel{a}{\rightarrow} E'$ and $E' \sim E$. Now we distinguish two subcases.

– Synchronization ($a = \tau$). Then $[\![D_1]\!] \mid [\![D_2]\!] \stackrel{\tau}{\rightarrow} E' = E_1 \mid E_2$ is deduced from $[\![D_1]\!] \stackrel{b}{\rightarrow} E_1$ and $[\![D_2]\!] \stackrel{\bar{b}}{\rightarrow} E_2$. By induction on D_1, D_2 there exists $D_1', D_2' \in \mathcal{E}^{\mathrm{def}}$ such that $[\![D_1']\!] \sim E_1$, $[\![D_2']\!] \sim E_2$ and $D_1 \stackrel{b}{\rightarrow} D_1'$, $D_2 \stackrel{\bar{b}}{\rightarrow} D_2'$; hence $D_1 \mid D_2 \stackrel{\tau}{\rightarrow} D_1' \mid D_2'$, and $[\![D_1' \mid D_2']\!] \sim [\![D_1']\!] \mid [\![D_2']\!] \sim E_1 \mid E_2 \sim E$, by Lemma 6.
– No synchronization. Without loss of generality we assume that a is performed by $[\![D_1]\!]$, so we have $[\![D_1]\!] \mid [\![D_2]\!] \stackrel{a}{\rightarrow} E' = E'' \mid [\![D_2]\!]$ and $[\![D_1]\!] \stackrel{a}{\rightarrow} E''$. By induction on D_1 there exists $D_1' \in \mathcal{E}^{\mathrm{def}}$ such that $[\![D_1']\!] \sim E''$ and $D_1 \stackrel{a}{\rightarrow} D_1'$, hence $D_1 \mid D_2 \stackrel{a}{\rightarrow} D_1' \mid D_2$ and $[\![D_1' \mid D_2]\!] \sim [\![D_1']\!] \mid [\![D_2]\!] \sim E'' \mid [\![D_2]\!] \sim E$, by Lemma 6.

□

Finally we can prove Theorem 4 by using Lemmas 13 and 14. Since $D \in \mathcal{E}^{\mathrm{def}}$ and $[\![D]\!] \in \mathcal{E}^!$, the bisimulation we consider is a relation $\sim \subseteq \mathcal{E}^{\mathrm{def}} \times \mathcal{E}^!$. It is formally defined as follow.

Definition 9 (Strong Bisimulation on $\mathcal{E}^{\mathrm{def}} \times \mathcal{E}^!$). *A binary relation $\mathcal{R} \subseteq \mathcal{E}^{\mathrm{def}} \times \mathcal{E}^!$ is a* strong bisimulation *if $(D, E) \in \mathcal{R}$ implies, for all $a \in Act$,*

- *if* $D \stackrel{a}{\rightarrow} D'$*, then there exists* $E' \in \mathcal{E}^{!}$ *such that* $E \stackrel{a}{\rightarrow} E'$ *and* $(D', E') \in \mathcal{R}$*;*
- *if* $E \stackrel{a}{\rightarrow} E'$*, then there exists* $D' \in \mathcal{E}^{\mathrm{def}}$ *such that* $D \stackrel{a}{\rightarrow} D'$ *and* $(D', E') \in \mathcal{R}$*.*

Two processes $D \in \mathcal{E}^{\mathrm{def}}, E \in \mathcal{E}^{!}$ *are* strongly bisimilar*, denoted by* $D \sim E$*, if there exists a strong bisimulation* $\mathcal{R}$ *containing the pair* (D, E)*.*

Proof (Theorem 4). Let $\mathcal{R} \equiv \{ (D, [\![D]\!]) : D \in \mathcal{E}^{\mathrm{def}} \}$. Clearly, by Lemmas 13 and 14, $\mathcal{R}$ is a strong bisimulation up to $\sim$. We conclude that $D \sim [\![D]\!]$ for every $D \in \mathcal{E}^{\mathrm{def}}$. □