REIHE INFORMATIK
15/96

A generalized inverse expansion for pure interleaving

Huu Trung Do
Universität Mannheim
Fakultät für Mathematik und Informatik
Seminargebäude A5
D-68131 Mannheim

A generalized inverse expansion for pure interleaving

Huu Trung Do

Universität Mannheim, Fakultät für Mathematik und Informatik, Germany
Lehrstuhl für Praktische Informatik I
do@pi1.informatik.uni-mannheim.de

16. October 1996

Abstract

Inverse expansion for pure interleaving which is introduced in [PHQ+92] is a method for transforming a sequential finite-state process given in Basic LOTOS into two subprocesses running independently. Thereby, the sets of the gates occurring in these subprocesses are disjoint and must be given as the input parameter by a user. The property fulfilled by this transformation is the strong equivalence according to [Mil89]. In this paper this method is generalized, i.e. the given process is transformed into more than two processes. Moreover, it is also applicable to the class of the recursive processes which is not treated in [PHQ+92].

Contents

1 Introduction

In the design of complex systems it is desirable to have a set of tools which support the system designer and implementors along the trajectory from an inital, abstract specification, down to concrete design and implementation. For this purpose many so-called correctness preserving transformations have been investigated in the last years [BvdLV95].

These transformations can help a designer to transform a given LOTOS specification $S1$ into a new LOTOS specification $S2$ that fulfills some new design properties and, at the same time, preserves the correctness by guaranteeing that S_1 and S_2 are semantically equivalent.

One of these correctness preserving transformations is *Inverse expansion* which is based on the inversion of the *Expansion Theorem* and introduced in [PHQ+92]. There are two types of *Inverse expansion* to be distinguished:

1. The *Pure Interleaving*

   This method decomposes a sequential finite-state process into two subprocesses running independently.

2. The *Visible Communication Decomposition*

   This method decomposes a sequential finite-state process into the parallel composition of two subprocesses running asynchronously.

In this paper we only concern with the inverse expanssion [1] in the case of pure interleaving. The other case will not be discussed further.

The formal description of *Pure Interleaving* is as follows.: Let P be a sequential process and A be the set of gates in P with $A = A_1 \cup A_2$ and $A_1 \cap A_2 = \emptyset$. Then P is transformed into $Q = Q_1 \,|||\, Q_2$ with $P \sim Q$. $|||$ means that Q_1 and Q_2 are running independently and $\sim$ stands for the strong equivalence according to [Mil89]. The set of gates performed by Q_i for $i = 1, 2$ is A_i.

Unfortunately, the case where P is recursive is not treated in [PHQ+92]. It is therefore desirable to have a generalized method which allows P to be recursive and transforms P not only into two, but more than two processes, e.g. $Q = Q_1 \,|||\, Q_2 \,|||\cdots|||\, Q_n$ where $n \in I\!N$ and $A = A_1 \cup A_2 \cdots A_n$ with $A_i \cap A_j = \emptyset$ for $i, j \in \{1, \ldots, n\}$ and $i \neq j$.

In this paper such a generalized method is presented. The idea of this method is different from [PHQ+92] and has some analogies with the idea of the method presented in [Jan85] for the COSY Formalism. However, the notion of equivalence defined by Janicki is not a strong bisimulation equivalence and has absolutely an another intention which we do not follow here.

The rest of the paper is organized as follows. Section 2 gives the syntax and semantic of the subset of Basic LOTOS. In section 3 the transformation problem and the transformation method are explained. For application section 4 recalls a demonstration example that was already discussed in [BvdLV95] for inverse expansion, but to which the inverse expansion's method presented in [PHQ+92] cannot be applied since it is not applicable to the class of recursive processes. The solution found there is just computed by hand. Here we will show that this solution can be obtained by our method. In section 5 the correctness of the method discussed in section 3 is proven. Section 6 concludes the paper.

2 Syntax and semantic

This section recalls the syntax and the operational semantic of the subset of Basic LOTOS. Moreover, the strong bisimulation equivalence according to [Mil89] and some basic notations that will be used in section 3 are also introduced. For the details to LOTOS the reader is referred to [ISO89].

Definition 2.1 *Let $\mathcal{G}$ be a set of action names, $\mathcal{PN}$ a set of process names, $g \in \mathcal{G}$, $G \subseteq \mathcal{G}$ and $P \in \mathcal{PN}$. Then $\mathcal{L}$ is defined by the following grammar:*

$$B \quad ::= \quad \textbf{stop} \quad | \quad (g; B) \quad | \quad (B \,[]\, B) \quad | \quad (B \,|[G]|\, B) \quad | \quad P$$

$\square$

[1] In the framework of this paper *Inverse Expansion* is understood as *Inverse Expansion* for *Pure Interleaving*.

stop represents an inactive process that cannot offer anything to the environment. $g; B$ is a process that first executes g and behaves after that like B. $[]$ is a nondeterministic operator, e.g. $B_1 \, [] \, B_2$ behaves either like B_1 if the first action resolved in interaction with the environment stems from B_1 or like B_2 if otherwise. The parallel composition of two processes is represented by $B_1 \, |[G]| \, B_2$ where $g \in G$ is a synchronisation action, i.e. an action that can only be performed if g is performed by B_1 and B_2 in co-operation, in other words at the same time. P denotes a process instantiation. With P it is possible to define a process to be recursive.

For the rest of the paper $|[\emptyset]|$ is also denoted by $|||$. Each $B \in \mathcal{L}$ is called a process. Note the parentheses enclosing the process terms are omitted if they are not important. Let A be a finite set then $\sum_{a \in A} a; B(a)$ stands for $a_1; B(a_1) \, [] \, a_2; B(a_2) \ldots [] \, a_n; B(a_n)$ if $A = \{a_1, a_2, \ldots, a_n\}$. Analogical to this, as a shorthand for $B_1 \, ||| \, B_2 \ldots ||| \, B_n$ we write $|||_{i=1}^{n} B_i$ or $|||_{i \in I} B_i$ where $I = \{1, \ldots, n\}$.

Remark 2.1 For technical reasons we assume that the following holds:

1. The letter 'S' occurs in $\mathcal{PN}$.
2. If $P \in \mathcal{P}$ then $P\_n \in \mathcal{PN}$, where $n \in I\!N_0$. $\square$

Definition 2.2 *A finite set $\mathcal{P} \subseteq (\mathcal{PN} \times \mathcal{L})$ is called a process environment if*

$$\forall (P, B), (P', B') \in \mathcal{P} : (P, B) \neq (P', B') \Longrightarrow P \neq P'.$$

The set of all process environments is denoted with $Env_{\mathcal{L}}$, i.e.

$$Env_{\mathcal{L}} := \{\mathcal{P} \mid \mathcal{P} \text{ is a process environment}\}.$$

$\square$

We define a function that assigns the set of all action names occurring in B to each process B.

Definition 2.3 *Let $B \in \mathcal{L}$ and $\mathcal{P} \in Env_{\mathcal{L}}$. Then*

- *$Act(B)$ is inductively defined as follows:*
  1. $B = \textbf{stop} \Longrightarrow Act(B) := \emptyset$.
  2. $B = (g; B') \Longrightarrow Act(B) := \{g\} \cup Act(B')$.
  3. $B = (B_1 \, [] \, B_2) \Longrightarrow Act(B) := Act(B_1) \cup Act(B_2)$.
  4. $B = (B_1 \, |[g_1, \ldots, g_n]| \, B_2) \Longrightarrow Act(B) := Act(B_1) \cup Act(B_2)$.
  5. $B = P \Longrightarrow Act(B) := \emptyset$.
- $Act(\mathcal{P}) := \bigcup \{Act(B) \mid (P, B) \in \mathcal{P}\}$.
- $Act(B, \mathcal{P}) := Act(B) \cup Act(\mathcal{P})$. $\square$

The operational semantic of $\mathcal{L}$ is a function that assigns a transition system to each $B \in \mathcal{L}$ and is defined with the transition rules. We first give the definition of transition system as follows.

Definition 2.4 *Let L be a set. Then $T = (Q, \Leftrightarrow\!\!\!\rightarrow, q_0)$ with*

- *Q is a set (of states).*

- $\Leftrightarrow\!\!\rightarrow \subseteq Q \times L \times Q$ (*transition relation*)
- $q_0 \in Q$ (*initial state*)

is called a transition system. $\mathcal{TS}$ *denotes the class of all transition systems. We say* T *is finite if* Q *and* L *are finite.* $\square$

Note, for a shorthand $p \overset{e}{\Leftrightarrow\!\!\rightarrow} q$ stands for $(p, e, q) \in \Leftrightarrow\!\!\rightarrow$.

Definition 2.5 *Let* $B \in \mathcal{L}$. *Then the operational semantic of* $\mathcal{L}$ *is a function* $\mathcal{OS} : (\mathcal{L} \times Env_{\mathcal{L}}) \rightarrow \mathcal{TS}$ *defined as*

$$\mathcal{OS}(B, \mathcal{P}) := (\mathcal{L}, \Leftrightarrow\!\!\rightarrow_{\mathcal{P}}, B),$$

where $\Leftrightarrow\!\!\rightarrow_{\mathcal{P}} \subseteq \mathcal{L} \times \mathcal{G} \times \mathcal{L}$ *is defined as follows:* $\Leftrightarrow\!\!\rightarrow_{\mathcal{P}}$ *is the least set which fulfills the following transition rules:*

1. $(g; B) \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B$
2. $\dfrac{B_1 \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_1'}{(B_1 \,[]\, B_2) \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_1'}$
3. $\dfrac{B_2 \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_2'}{(B_1 \,[]\, B_2) \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_2'}$
4. $\dfrac{B_1 \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_1' \wedge a \notin G}{(B_1 \,|[G]|\, B_2) \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} (B_1' \,|[G]|\, B_2)}$
5. $\dfrac{B_2 \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_2' \wedge a \notin G}{(B_1 \,|[G]|\, B_2) \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} (B_1 \,|[G]|\, B_2')}$
6. $\dfrac{B_1 \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_1' \wedge B_2 \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B_2' \wedge a \in G}{(B_1 \,|[G]|\, B_2) \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} (B_1' \,|[G]|\, B_2')}$
7. $\dfrac{(P, B) \in \mathcal{P} \wedge B \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'}{P \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'}$ $\square$

Based on the transition systems the strong bisimulation equivalence according to [Mil89] is defined as follows:

Definition 2.6 *Let* $T_i = (Q_i, \rightarrow_i, q_i)$ *with* $i = 1, 2$ *be a transition system.* T_1 *and* T_2 *are (strongly bisimilar) equivalent* $(T_1 \sim T_2)$ *if there exists a relation* $R \subseteq Q_1 \times Q_2$ *with* $(q_1, q_2) \in R$ *and for all* $(p, q) \in R$ *the following holds:*

1. *If* $p \overset{a}{\Leftrightarrow\!\!\rightarrow}_1 p'$ *then* $\exists q' \in Q_2 : q \overset{a}{\Leftrightarrow\!\!\rightarrow}_2 q'$ *and* $(p', q') \in R$.
2. *If* $q \overset{a}{\Leftrightarrow\!\!\rightarrow}_2 q'$ *then* $\exists p' \in Q_1 : p \overset{a}{\Leftrightarrow\!\!\rightarrow}_1 p'$ *and* $(p', q') \in R$.

Such relation R *is called a bisimulation between* T_1 *and* T_2. $\square$

Definition 2.7 *Let* $B, B' \in \mathcal{L}$ *and* $\mathcal{P}, \mathcal{P}' \in Env_{\mathcal{L}}$. B *in* $\mathcal{P}$ *and* B' *in* $\mathcal{P}'$ *are (strongly bisimilar) equivalent* $(B_1 \sim_{\mathcal{P},\mathcal{P}'} B_2)$ *if* $\mathcal{OS}(B, \mathcal{P}) \sim \mathcal{OS}(B, \mathcal{P}')$. $\square$

The following lemma shows that $\mathcal{OS}(B,\mathcal{P})$ is equivalent with the transition system whose set of states consists of all states which can be reached from an initial state via $\Leftrightarrow\!\!\!\rightarrow_{\mathcal{P}}$. This property will be often used in section 3 and 5.

Definition 2.8 *Let $B \in \mathcal{L}$ and $\mathcal{P} \in Env_{\mathcal{L}}$. $Re(B,\mathcal{P})$ is the least set fulfilling the following:*

- $B \in Re(B,\mathcal{P})$.
- $\forall C' : (\exists C \in Re(B,\mathcal{P}) : \exists g : C \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} C') \Longrightarrow C' \in Re(B,\mathcal{P})$. □

Definition 2.9 *Let $B \in \mathcal{L}$ and $\mathcal{P} \in Env_{\mathcal{L}}$. Then*

$$TS(B,\mathcal{P}) := (Re(B,\mathcal{P}), \Leftrightarrow\!\!\!\rightarrow, B),$$

where $\Leftrightarrow\!\!\!\rightarrow \; = \; \Leftrightarrow\!\!\!\rightarrow_{\mathcal{P}} \cap (Re(B,\mathcal{P}) \times Re(B,\mathcal{P}))$. □

Lemma 2.1 *Let $B \in \mathcal{L}$ and $\mathcal{P} \in Env_{\mathcal{L}}$. Then $OS(B,\mathcal{P}) \sim TS(B,\mathcal{P})$.*

Proof: Easy and omitted. □

Now we show that $Re(B,\mathcal{P})$ can be identified with the least fixpoint of a continuous function on the so-called c.p.o (complete partial order). The notions like c.p.o., fixpoint of a continuous function, poset ... stem from the well-known domain theory and are summarized briefly in appendix A. For details see e.g. [Win93].

To prove this statement we first construct a c.p.o. on which we then define a function and show that this function is continuous. Applying the Kleene's theorem (see appendix A) the proof of this statement is straightforward.

Definition 2.10 *Let $B \in \mathcal{L}$. Then*

1. $\mathcal{M}(B) := \{m \mid \; m \subseteq \mathcal{L} \wedge \{B\} \subseteq m\}$.
2. $Pos(B) := (\mathcal{M}(B), \subseteq, \{B\})$. □

Lemma 2.2 *$Pos(B)$ is a c.p.o.*

Proof: Easy and omitted. □

Definition 2.11 *$\mathcal{F}1_{B,\mathcal{P}} : \mathcal{M}(B) \rightarrow \mathcal{M}(B)$ is defined as*

$$\mathcal{F}1_{B,\mathcal{P}}(m) := m \cup \{C' \mid \; \exists C \in m : \exists g : C \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} C'\}.$$ □

Lemma 2.3 *$\mathcal{F}1_{B,\mathcal{P}}$ is continuous.*

Proof: Let $M \subseteq \mathcal{M}(B)$ be a chain and $L = \bigcup M$. We obtain:

$$\begin{aligned}
\mathcal{F}1_{B,\mathcal{P}}(L) &= L \cup \{C' \mid \; \exists C \in L : \exists g : C \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} C'\} \quad \text{Def. 2.11} \\
&= L \cup \{C' \mid \exists m \in M : \exists C \in m : \exists g : C \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} C'\} \\
&= \{C' \mid (\exists m \in M : C' \in m) \vee (\exists m \in M : \exists C \in m : \exists g : C \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} C')\} \\
&= \{C' \mid \exists m \in M : C' \in m \vee \exists C \in m : \exists g : C \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} C'\} \\
&= \{D \mid \exists m \in M : D \in m \cup \{C' \mid \exists C \in m : \exists g : C \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} C'\}\} \\
&= \{D \mid \exists m \in M : D \in \mathcal{F}1_{B,\mathcal{P}}(m)\} \\
&= \{D \mid \exists X : (\exists m \in M : X = \mathcal{F}1_{B,\mathcal{P}}(m)) \wedge D \in X\} \\
&= \{D \mid \exists X \in M' : D \in X\}, \\
&\quad\;\, \text{where } M' = \{X \mid \exists m \in M : X = \mathcal{F}1_{B,\mathcal{P}}(m)\}. \\
&= \bigcup \{X \mid \exists m \in M : X = \mathcal{F}1_{B,\mathcal{P}}(m)\}
\end{aligned}$$

□

Corollar 2.1 $Re(B,\mathcal{P}) = \bigcup_{i\in I\!N_0} \mathcal{F}1^i_{B,\mathcal{P}}(\{B\})$. $\square$

As the aim of this paper is to give an approach to transform a sequential finite-state process into n processes running independently we restrict ourselves to the subset of $\mathcal{L}$ (denoted by $\mathcal{L}_{seq}$) in which the opportunity of describing the parallel processes is not given, i.e. only the sequential processes are considered.

Definition 2.12 *Let $\mathcal{G}$ and $\mathcal{PN}$ be the sets in the definition 2.1. Let $g \in \mathcal{G}$ and $P \in \mathcal{PN}$. Then $\mathcal{L}_{seq}$ is defined by the following grammar:*

$$B \quad ::= \quad \textbf{stop} \quad | \quad (g;P) \quad | \quad (B\,[]\,B)$$

$\square$

It is obvious that $\mathcal{L}_{seq}$ is a subset of $\mathcal{L}$. Therefore, the semantic defined for $\mathcal{L}$ is also valid for $\mathcal{L}_{seq}$.

Definition 2.13 *A process environment $\mathcal{P} \in Env_{\mathcal{L}}$ is called sequential if $\forall (P,B) \in \mathcal{P} : B \in \mathcal{L}_{seq}$ holds. The set of all sequential process environments is denoted by Env_{seq}, i.e.*

$$Env_{seq} := \{\mathcal{P} \,|\, \mathcal{P} \textit{ is sequential}\}.$$

$\square$

Note that the transition system $\mathcal{OS}(B,\mathcal{P})$ with $B \in \mathcal{L}$ and $\mathcal{P} \in Env_{\mathcal{L}}$ is dependent on $\mathcal{P}$. There can be a case where a state P (process name) in $Re(B,\mathcal{P})$ does not have a transition, i.e. $\neg(\exists B' \in \mathcal{L} : \exists g : P \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} B')$, because of $\neg(\exists C : (P,C) \in \mathcal{P})$. Such $\mathcal{P}$ as a process environment is not complete and so not praticable. That's why we now introduce for $\mathcal{L}$ a new notion of the so-called closed processes. For a closed process $B \in \mathcal{L}_{seq}$ in an environment $P \in Env_{seq}$ we will show that the following statement

$$\forall P \in \mathcal{PN} : P \in Re(B,\mathcal{P}) \Longrightarrow \exists C : (P,C) \in \mathcal{P} \qquad (*)$$

holds. Moreover, if a process $B \in \mathcal{L}_{seq}$ is closed in a process environment $\mathcal{P} \in Env_{seq}$ then B is finite-state in $\mathcal{P}$. Thereby, a process $B \in \mathcal{L}$ in a process environment $\mathcal{P} \in Env_{\mathcal{L}}$ is called finite-state if $Re(B,\mathcal{P})$ is finite.

Definition 2.14 *Let $B \in \mathcal{L}$ and $P \in \mathcal{P}$. $Pv(B)$ is inductively defined as follows:*

1. $B = \textbf{stop} \Longrightarrow Pv(B) := \emptyset$.
2. $B = (g;B') \Longrightarrow Pv(B) := Pv(B')$.
3. $B = (B_1\,[]\,B_2) \Longrightarrow Pv(B) := Pv(B_1) \cup Pv(B_2)$.
4. $B = (B_1\,|[g_1,\ldots,g_n]|\,B_2) \Longrightarrow Pv(B) := Pv(B_1) \cup Pv(B_2)$.
5. $B = P \Longrightarrow Pv(B) := \{P\}$. $\square$

$Pv(B)$ is a set of all process names occuring in B.

Definition 2.15 *Let $\mathcal{P} \in Env_{\mathcal{L}}$. Then $PN(\mathcal{P}) := \{P \,|\, (P,B) \in \mathcal{P}\}$.* $\square$

Definition 2.16 *Let $B \in \mathcal{L}$ and $\mathcal{P} \in Env_{\mathcal{L}}$. Then $Rpv(B,\mathcal{P})$ is the least set which fulfills the following:*

1. $Pv(B) \subseteq Rpv(B, \mathcal{P})$.
2. $\forall P \in Rpv(B, \mathcal{P}) : \exists B' : (P, B') \in \mathcal{P} \Longrightarrow Pv(B') \subseteq Rpv(B, \mathcal{P})$. □

We define the notion of a closed process B in an process environment $\mathcal{P}$ as follows.

Definition 2.17 *A process $B \in \mathcal{L}$ is closed in $\mathcal{P} \in Env_{\mathcal{L}}$ if $Rpv(B, \mathcal{P}) = PN(\mathcal{P})$.* □

To prove the proposition $(*)$ we first show that by analogy with $Re(B, \mathcal{P})$ the set $Rpv(B, \mathcal{P})$ can be identified with the least fixpoint of the continuous function on the c.p.o. which is defined as follows.

Definition 2.18 *Let $B \in \mathcal{L}_{seq}$ and $\mathcal{P} \in Env_{seq}$. Then*

1. $\mathcal{M}(B, \mathcal{P}) := \{m \mid \ m \subseteq (Pv(B) \cup (\bigcup_{(P,B') \in \mathcal{P}} Pv(B'))) \wedge Pv(B) \subseteq m\}$.
2. $Pos(B, \mathcal{P}) := (\mathcal{M}(B, \mathcal{P}), \subseteq, Pv(B))$.

□

Lemma 2.4 *$Pos(B, \mathcal{P})$ is a c.p.o.* □

Proof: Easy and omitted. □

Definition 2.19 *$\mathcal{F}2_{B,\mathcal{P}} : \mathcal{M}(B, \mathcal{P}) \Leftrightarrow\!\!\!\rightarrow \mathcal{M}(B, \mathcal{P})$ is defined as*

$$\mathcal{F}2_{B,\mathcal{P}}(m) := m \cup (\bigcup \{X \mid \exists P \in m : \exists B' : (P, B') \in \mathcal{P} \wedge X = Pv(B')\}).$$

□

Lemma 2.5 *$\mathcal{F}2_{B,\mathcal{P}}$ is continuous on $Pos(B, \mathcal{P})$.*

Proof: Let $M \subseteq \mathcal{M}(B, \mathcal{P})$ be a chain and $L = \bigcup M$. We obtain:

$$\begin{aligned}
\mathcal{F}2_{B,\mathcal{P}}(L) &= L \cup (\bigcup \{X \mid \exists P \in L : \exists B' : (P, B') \in \mathcal{P} \wedge X = Pv(B')\}) \qquad \text{Def. 2.19} \\
&= L \cup (\bigcup \{X \mid \exists m \in M : \exists P \in m : \exists B' : (P, B') \in \mathcal{P} \wedge X = Pv(B')\}) \\
&= \{C \mid (\exists m \in M : C \in m) \vee (\exists X : (\exists m \in M : \exists P \in m : \exists B' : (P, B') \in \mathcal{P} \\
&\qquad\qquad \wedge\ X = Pv(B')\}) \wedge C \in X)\} \\
&= \{C \mid \exists m \in M : C \in m \vee (\exists X : (\exists P \in m : \exists B' : (P, B') \in \mathcal{P} \\
&\qquad\qquad \wedge\ X = Pv(B')\}) \wedge C \in X)\} \\
&= \{C \mid \exists m \in M : C \in m \cup \\
&\qquad (\bigcup \{X \mid \exists P \in m : \exists B' : (P, B') \in \mathcal{P} \wedge\ X = Pv(B')\})\} \\
&= \{C \mid \exists m \in M : C \in \mathcal{F}2_{B,\mathcal{P}}(m)\} \\
&= \{C \mid \exists X \in M' : C \in X\}, \text{where } M' = \{X \mid \exists m \in M : X = \mathcal{F}2_{B,\mathcal{P}}(m)\} \\
&= \bigcup \{X \mid \exists m \in M : X = \mathcal{F}2_{B,\mathcal{P}}(m)\}
\end{aligned}$$

□

Corollar 2.2 $Rpv(B, \mathcal{P}) = \bigcup_{i \in \mathbb{N}_0} \mathcal{F}2^i_{B,\mathcal{P}}(Pv(B))$. □

This corollar is now used for proving the equation $Rpv(B, \mathcal{P}) = Re(B, \mathcal{P}) \setminus \{B\}$ which then implies obviously the proposition $(*)$. We first need some preliminaries.

Lemma 2.6 *Let $B \in \mathcal{L}_{seq}$ and $\mathcal{P} \in Env_{seq}$. Then*

$$\forall B' : (\exists g : B \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B') \Leftrightarrow B' \in Pv(B).$$

Proof: Structural induction on B. Easy and omitted. □

Lemma 2.7 *Let $P \in PN(\mathcal{P})$ and $\mathcal{P} \in Env_{\mathcal{L}}$. Then*

$$\forall B : \exists g : P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B \Leftrightarrow \exists C : (P, C) \in \mathcal{P} \wedge C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B.$$

Proof: This is a consequence of definition 2.5. □

Lemma 2.8 *Let $B \in \mathcal{L}_{seq}$, $\mathcal{P} \in Env_{seq}$ and $i \in \mathbb{N}$. Then*

$$\mathcal{F}1^{i}_{B,\mathcal{P}}(\{B\}) \setminus \{B\} = \mathcal{F}2^{i-1}_{B,\mathcal{P}}(Pv(B)).$$

Proof: We show with mathematical induction on $\mathbb{N}$. For $i = 1$ it is obvious. We assume that the induction hypothesis holds for $i \Leftrightarrow 1$ where $i > 2$. The induction step can now be shown as follows: Let $L = \mathcal{F}1^{i-1}_{B,\mathcal{P}}(\{B\})$ and $L' = \mathcal{F}2^{i-2}_{B,\mathcal{P}}(\{B\})$. We obtain:

$$\begin{aligned}
\mathcal{F}1^{i}_{B,\mathcal{P}}(\{B\}) &= L \cup \ \{P' \,|\, \exists P \in L : \exists g : P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'\} \\
&= L \cup \{P' \,|\, \exists g : B \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'\} \cup \{P' \,|\, \exists P \in L \setminus \{B\} : \exists g : P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'\} \\
&= L \cup \{P' \,|\, \exists P \in L \setminus \{B\} : \exists g : P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'\}, \text{ da } Pv(B) \subseteq L. \\
&= L \cup \{P' \,|\, \exists P \in L \setminus \{B\} : \exists g : \exists B' : (P, B') \in \mathcal{P} \wedge B' \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'\}, \\
&\quad \text{Lemma 2.7. Note that } P \in L \setminus \{B\} (= L') \text{ is a process name.} \\
&= L \cup \{P' \,|\, \exists P \in L \setminus \{B\} : \exists B' : (P, B') \in \mathcal{P} \wedge \exists g : B' \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'\} \\
&= L \cup \{P' \,|\, \exists P \in L \setminus \{B\} : \exists B' : (P, B') \in \mathcal{P} \wedge P' \in Pv(B')\}, \\
&\quad \text{Lemma 2.6} \\
&= L \cup \{P' \,|\, \exists X : (\exists P \in L \setminus \{B\} : \\
&\qquad\qquad \exists B' : (P, B') \in \mathcal{P} \wedge X = Pv(B')) \wedge P' \in X\} \\
&= L' \cup \{B\} \cup (\bigcup \{X \,|\, \exists P \in L' : \exists B' : (P, B') \in \mathcal{P} \wedge X = Pv(B')\}) \\
&= \mathcal{F}2^{i-1}_{B,\mathcal{P}}(Pv(B)) \cup \{B\}
\end{aligned}$$

□

Proposition 2.1 *Let $B \in \mathcal{L}_{seq}$ and $\mathcal{P} \in Env_{seq}$. Then $Rpv(B, \mathcal{P}) = Re(B, \mathcal{P}) \setminus \{B\}$.*

Proof:

$$\begin{aligned}
Rpv(B, \mathcal{P}) &= \bigcup_{i \in \mathbb{N}_0} \mathcal{F}2^{i}_{B,\mathcal{P}}(Pv(B)) \\
&= \bigcup_{i \in \mathbb{N}_0} (\mathcal{F}1^{i+1}_{B,\mathcal{P}}(\{B\}) \setminus \{B\}) \\
&= (\bigcup_{i \in \mathbb{N}_0} (\mathcal{F}1^{i+1}_{B,\mathcal{P}}(\{B\})) \setminus \{B\} \\
&= (\bigcup_{i \in \mathbb{N}_0} (\mathcal{F}1^{i}_{B,\mathcal{P}}(\{B\})) \setminus \{B\} \\
&= Re(B, \mathcal{P}) \setminus \{B\}
\end{aligned}$$

□

From this proposition we conclude that a process $B \in \mathcal{L}_{seq}$ in a process environment $\mathcal{P} \in Env_{seq}$ is finite-state if B is closed in $\mathcal{P}$. For a finite transition system T we give in the following a function to construct a process in $\mathcal{L}_{seq}$ whose transition system is equivalent with T.

Definition 2.20 *Let $T = (Q, \Leftrightarrow\!\!\!\rightarrow, q_0)$ be finite, $q \in Q$ and $f : Q \to I\!N_0$ an injective function. Then*

1. $Out(q, T) := \{(g, q') \mid \ q \stackrel{g}{\Leftrightarrow\!\!\!\rightarrow} q'\}$.

2. $Proc(T, S\_f(q)) := \begin{cases} \sum_{(g,q')\in Out(q,T)} g; S\_f(q') & \text{if } \ Out(q,T) \neq \emptyset \\ \textbf{stop} & \text{if } \ \text{else otherwise} \end{cases}$

3. $PE(T, f) := \{(S\_f(q), Proc(T, S\_f(q))) \mid \ q \in Q\}$ □

Note the letter 'S' is a process name in $\mathcal{PN}$ (see remark 2.1). The parentheses in $Proc(T, S\_f(q))$ are omitted because we have in fact $(B_1 \,[]\, B_2) \,[]\, B_3 \equiv B_1 \,[](B_2 \,[]\, B_3)$, where $B_i = g_i; P_i$ with $i = 1, 2, 3$, $g_i \in \mathcal{G}$ and $P_i \in \mathcal{PN}$, and $\equiv$ is defined as follows:

$$B \equiv B' :\leftrightarrow B \stackrel{g}{\Leftrightarrow\!\!\!\rightarrow} B'' \Leftrightarrow B' \stackrel{g}{\Leftrightarrow\!\!\!\rightarrow} B''$$

That means that [] is associative relating to $\equiv$. The proof of this proposition is not difficult and therefore omitted.

Lemma 2.9 *Let $T = (Q, \Leftrightarrow\!\!\!\rightarrow, q_0)$ be finite. Then*

$$T \sim \mathcal{OS}(Proc(T, S\_f(q_0)), PE(T, f)).$$

Proof: Since $\mathcal{OS}(Proc(T, S\_f(q_0)), PE(T, f)) \sim \mathcal{OS}(S\_f(q_0), PE(T, f))$ holds (it is easy to construct a bisimulation between these transition systems) we show

$$T \sim \mathcal{OS}(S\_f(q_0), PE(T, f)).$$

Let $R = \{(q, S\_f(q)) \,|\, q \in Q\}$. Clearly that $R \subseteq (Q \times \mathcal{P})$. We show that R is a bisimulation between T and $\mathcal{OS}(S\_f(q_0), PE(T, f))$. Let $(q, S\_f(q)) \in R$.

1. From $q \stackrel{g}{\Leftrightarrow\!\!\!\rightarrow} q'$ we have $(a, q') \in Out(q)$ which implies

   $$S\_f(q) = (\sum_{(b,q'')\in M} b; S\_f(q'')) \ \ [] \ \ a; S\_f(q')$$

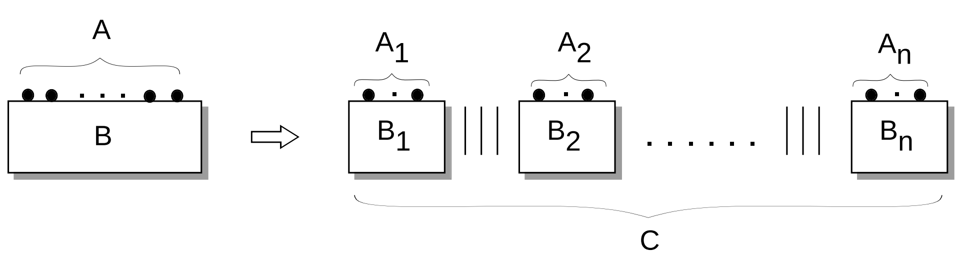
   where $M = Out(q, T) \setminus \{(a, q')\}$. Hence we can follow that $S\_f(q) \stackrel{g}{\Leftrightarrow\!\!\!\rightarrow} S\_f(q')$. As $q' \in Q$ holds so $(q', S\_f(q')) \in R$ holds.

2. By analogy with 1.

Since $(q_0, S\_f(q_0)) \in R$ holds we obtain $T \sim OS(S\_f(q_0), PE(T, f))$. □

3 Transformation

The aim of the transformation in this section is to obtain from a sequential process B an equivalent process C with a higher degree of parallelism. That means that B is decomposed into at least two processes running independently (see the figure below).



where A is the set of gates performed by B, A_i the set of gates performed by B_i for $i = 1, \ldots, n$, $A = \bigcup_{i=1}^{n} A_i$ and $A_i \cap A_j = \emptyset$ for $i, j \in \{1, \ldots, n\}$ with $i \neq j$.

For applying this transformation we assume like [PHQ$^+$92] that the following must be given as input: 1) The closed process $B \in \mathcal{L}_{seq}$ which is closed in $\mathcal{P} \in Env_{seq}$ and 2) the sets A_i mentioned above.

The method presented in this paper differs from [PHQ$^+$92] in the following items:

1. It is also applicable to the class of recursive processes which is not allowed in [PHQ$^+$92].
2. The approach chosen in this paper is not the same like this in [PHQ$^+$92]. The idea of the method in this paper has some analogy with the idea of the method presented in [Jan85] for the COSY Formalism. However, the notion of equivalence defined by Janicki is not a strong bisimulation equivalence and has absolutely an another intention.
3. A given process is transformed into more than two processes. These processes can be computed independently of each other.

In the remainder of this section we first formalise the transformation problem. After that we present a method to solve it.

3.1 Formal description of the transformation problem

The formal description of the transformation problem is given in the following definition.

Definition 3.1 *Let $B \in \mathcal{L}_{seq}$ and $\mathcal{P} \in Env_{seq}$ where B is closed in $\mathcal{P}$. Let $A_i \subseteq Act(B, \mathcal{P})$ with $A_i \neq \emptyset$ and $i = 1, \ldots, n$ where the following holds:*

- $Act(B, \mathcal{P}) = \bigcup_{i=1}^{n} A_i$.
- $A_i \cap A_j = \emptyset$ *for* $i, j \in \{1, \ldots, n\}$ *and* $i \neq j$.

B in $\mathcal{P}$ is splitted under $\mathcal{A}$, where $\mathcal{A} = \{A_1, \ldots, A_n\}$, if there is a process $C \in \mathcal{L}$ with $C = |||_{i=1}^{n} C_i$ such that the following holds:

1. $C_i \in \mathcal{L}_{seq}$ *for* $i = 1, \ldots, n$.
2. *For each i there exists $\mathcal{P}_i \in Env_{seq}$ such that C_i is closed in $\mathcal{P}_i$.*
3. $Act(C_i, \mathcal{P}_i) = A_i$
4. $PN(\mathcal{P}_i) \cap PN(\mathcal{P}_j) = \emptyset$ *for* $i, j \in \{1, \ldots, n\}$ *with* $i \neq j$.
5. $B \sim_{\mathcal{P}, \mathcal{P}'} C$, *where* $\mathcal{P}' = \bigcup_{i=1}^{n} \mathcal{P}_i$.

We say, C in $\mathcal{P}_1, \ldots, \mathcal{P}_n$ is a solution for B in $\mathcal{P}$ under $\mathcal{A}$. □

Note that $|||$ is associative relating to $\sim$ (see section 5). Thus the parentheses in C can be omitted. In addition $\mathcal{P}'$ is in fact a process environment.

3.2 Transformation method

To define a method solving the transformation problem we first need some preliminaries.

Definition 3.2 *Let $B \in \mathcal{L}_{seq}$, $A \subseteq \mathcal{G}$ and $i \in I\!N$. Then $Pj1(B, A, i)$ is inductively defined as follows:*

- *If $B = \mathbf{stop}$ then $Pj1(B, A, i) := \mathbf{stop}$.*
- *If $B = (g; B')$ then*

$$Pj1(B, A, i) := \begin{cases} (g; B'\_i) & \text{if} \quad g \in A \\ \mathbf{stop} & \text{else otherwise} \end{cases}$$

- *Let $B = (B_1 \,[]\, B_2)$.*
  - *If $Pj1(B_1, A, i) = \mathbf{stop} = Pj1(B_2, A, i)$ then $Pj1(B, A, i) := \mathbf{stop}$.*
  - *If $Pj1(B_1, A, i) \neq \mathbf{stop} = Pj1(B_2, A, i)$ then $Pj1(B, A, i) := Pj1(B_1, A, i)$.*
  - *If $Pj1(B_1, A, i) = \mathbf{stop} \neq Pj1(B_2, A, i)$ then $Pj1(B, A, i) := Pj1(B_2, A, i)$.*
  - *If $Pj1(B_1, A, i) \neq \mathbf{stop} \neq Pj1(B_2, A, i)$ then $Pj1(B, A, i) := (Pj1(B_1, A, i) \,[]\, Pj1(B_2, A, i))$.* □

In $Pj1(B, A, i)$ only such subprocesses of B which are prefixed with an action in A are numerated with i. The others are omitted. Note that $Pj1(B, A, i) \in \mathcal{L}_{seq}$. This can easy be proven with the structural induction on B.

Definition 3.3 *Let $\mathcal{P} \in Env_{seq}$, $A \subseteq \mathcal{G}$ and $i \in I\!N$. Then*

$$Pj2(\mathcal{P}, A, i) := \{(P\_i, Pj1(B, A, i)) \mid (P, B) \in \mathcal{P}\}.$$

□

Definition 3.4 *Let B and $\mathcal{A}$ be defined as in the definition 3.1. Then $Inv(B, \mathcal{A}) := |||_{i=1}^{n} C_i$ where $C_i = Pj1(B, A_i, i)$ for $i = 1, \ldots, n$.* □

Definition 3.5 *Let $P \in Env_{\mathcal{L}}$ and $X \subseteq PN(\mathcal{P})$. Then*

$$Del(\mathcal{P}, X) := \{(P, B') \mid P \in X \wedge \exists B' : (P, B') \in \mathcal{P}\}$$

□

The transformation method is based on the following important theorem:

Theorem 3.1 *Let B, $\mathcal{P}$ and $\mathcal{A}$ be defined as in the definition 3.1. Let $Inv(B, \mathcal{A}) = |||_{i=1}^{n} C_i$, where $C_i = Pj1(B, A_i, i)$ for $i = 1, \ldots, n$, and $\mathcal{P}' = \bigcup_{i=1}^{n} \mathcal{P}_i$, where*

$$\mathcal{P}_i = Del(Pj2(\mathcal{P}, A_i, i), Rpv(C_i, Pj2(\mathcal{P}, A_i, i))).$$

Then

a) If $B \sim_{\mathcal{P},\mathcal{P}'} Inv(B, \mathcal{A})$ then $Inv(B, \mathcal{A})$ in $\mathcal{P}_1, \ldots, \mathcal{P}_n$ is a solution for B in $\mathcal{P}$ under $\mathcal{A}$.

b) B in $\mathcal{P}$ is splitted under $\mathcal{A}$ iff $B \sim_{\mathcal{P},\mathcal{P}'} Inv(B, \mathcal{A})$.

Proof: The proof is postponed to the section 5. $\square$

We use this theorem to define the transformation method as follows: Let B, $\mathcal{P}$ and $\mathcal{A}$ be defined as in the definition 3.1.

1. Compute $C_i = Pj1(B, A_i, i)$ and $\mathcal{P}_i = Del(Pj2(\mathcal{P}, A_i, i), Rpv(C_i, Pj2(\mathcal{P}, A_i, i)))$ for $i = 1, \ldots, n$.
2. Let $C = |||_{i=1}^{n} C_i$ and $\mathcal{P}' = \bigcup_{i=1}^{n} \mathcal{P}_i$. Use e.g. the CWB-Tool (Concurrency WorkBench) [CPS93] to examine whether $B \sim_{\mathcal{P},\mathcal{P}'} C$ holds. If this is true then C is a solution. Otherwise, no solution does exist.

Note that to compute the set $\mathcal{Q} = Rpv(C_i, Pj2(\mathcal{P}, A_i, i))$ we have to compute the least fixpoint of the function $\mathcal{F}2_{C_i,\mathcal{Q}}$ because of the corollar 2.2. As $\mathcal{P}$ is finite $\mathcal{Q}$ is also finite. Thus $\mathcal{Q}$ is always computable.

Example 3.1 Let $B = a; P1 \,[]\, c; P2$ and $\mathcal{P}$ the process environment consisting of the following process instantiations:

- $P0 = B$
- $P1 = c; P3 \,[]\, b; P0$
- $P2 = a; P3 \,[]\, d; P0$
- $P3 = d; P1 \,[]\, b; P2$

It is easy to see that $Re(B, \mathcal{P}) = \{P0, P1, P2, P3\}$ holds, i.e. B is closed in $\mathcal{P}$. Let $\mathcal{A} = \{A_1, A_2\}$ with $A_1 = \{a, b\}$ and $A_2 = \{c, d\}$. We have

1. $C_1 = Pj1(B, A_1, 1) = a; P1\_1$ and $\mathcal{P}_1 = Del(Pj2(\mathcal{P}, A_1, 1), Rpv(C_1, Pj2(\mathcal{P}, A_1, 1)))$ consists of
   - $P0\_1 = a; P1\_1$
   - $P1\_1 = b; P0\_1$
2. $C_2 = Pj1(B, A_2, 2) = c; P2\_2$ and $\mathcal{P}_2 = Del(Pj2(\mathcal{P}, A_2, 2), Rpv(C_2, Pj2(\mathcal{P}, A_2, 2)))$ consists of
   - $P0\_2 = c; P2\_2$
   - $P2\_2 = d; P0\_2$

Let $C = C_1 \,|||\, C_2$ and $\mathcal{P}' = \mathcal{P}_1 \cup \mathcal{P}_2$. Using the CWB-Tool to analyse the strong bisimulation equivalence results $B \sim_{\mathcal{P},\mathcal{P}'} C$. Thus C is the solution of the transformation problem. $\square$

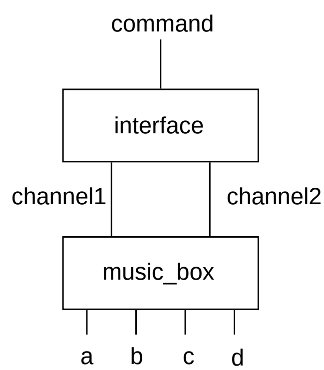
4 Application

In this section we demonstrate the practical applicability of the method presented in section 3.2. For it we recall an example that has already been discussed in [BvdLV95] for inverse expansion's method presented in [PHQ+92]. The solution found in [BvdLV95] for this example is computed just by 'hand' because the method in [PHQ+92] is not applicable to the class of the recursive processes. For this reason we like now to show how to obtain this solution systematically with the method presented in this paper.

In this example we deal with the Programmable Sound Sequencer whose function is described as follows (originally from [BvdLV95]):

The Programmable Sound Sequencer is a system which can accept requests for producing predefined sequences of sounds. More precisely, a user, identified by a password (`Psw1` or `Psw2`) can require the execution of a programm called `Prog1`, consisting of the sequence of sounds `(a, d)`, or of `Prog2`, consisting of sequence `(b, c, d)`. In fact, an elementary constraint is imposed, which increases the selectivity associated with the passwords: `Psw1` (resp. `Psw2`) entitles the user to select only `Prog1` (resp. `Prog2`).

The system can store up to two different requests, but does not necessary satisfy them in order in which they are accepted. The system's structure consists of two modules '`interface`' and '`music_box`'. The interface is responsible for a communication with the user environment and the music box for the output of sounds. Every time, whenever the parameter `Prog1` (resp. `Prog1`) with the consistent password is received at the gate `command` then it will be forwarded by the interface via a channel `channel1` (resp. `channel2`) (see the figure below). The music box will start to play the sound `a` and `b` (resp. `b`, `c` and `d`) if the program `Prog1` (resp. `Prog2`) is received.



The specification of the Programmable Sound Sequencer written in Full LOTOS is given as follows (Note in Full LOTOS there are many new constructs which cannot be explained within the framework of this paper. Reader are therefore referred to [ISO89, BB87].):

```
specification programmable_sound_sequencer [command, a, b, c, d] : noexit

  type Password is
      sorts Password
      opns Psw1, Psw2   :  --> Password
  endtype (*Password*)

  library Set, Boolean, NatrualNummber endlib

  type Program is Boolean
      sorts Program
      opns Prog1, Prog2   :  --> Programm
           _ eq _, _ ne _ : Program, Program --> Bool
      eqns
        ofsort Bool forall p,q: Program
            Prog1 eq Prog1 = true; Prog1 eq Prog2 = false;
            Prog2 eq Prog1 = false; Prog2 eq Prog2 = true;
            p ne q = not(p eq q);
  endtype (*Program*)

  type ProgramSet is Set actualizedby Program using
```

```
     sortnames  Bool for FBool
                Program for Element
                Program_Set for Set
  endtype (*ProgramSet*)

  type Consistency is Password, Program, Boolean
     opns consistent : password, program --> Bool
     eqns
        ofsort Bool
        consistent(Psw1, Prog1) = true;
        consistent(Psw2, Prog2) = true;
        consistent(Psw1, Prog2) = false;
        consistent(Psw2, Prog1) = false;
  endtype (*Consistency*)

behaviour
  hide channel1, channel2 in
     ( interface[command, channel1, channel2]({})
       |[channel1, channel2]| music_box[channel1, channel2, a, b, c, d]
     )
where
process interface[command, channel1, channel2](prog_set: Progam_Set) : noexit :=
     [Card(prog_set) eq 0] -->
        command ?psw: Password ?prog: Program [consistent(psw, prog)];
        interface[command, channel1, channel2](Insert(prog, prog_set))

[]   [Card(prog_set) eq Succ(0)] -->
        (   command ?psw: Password ?prog: Program
              [consistent(psw, prog) and (prog NotIn prog_set)];
            interface[command, channel1, channel2](Insert(prog, prog_set))
        []  (choice prog: Program  [] [prog IsIn prog_set] -->
              (  channel1 !prog [prog = Prog1];
                 interface[command, channel1, channel2](Remove(prog, prog_set))
              [] channel2 !prog [prog = Prog2];
                 interface[command, channel1, channel2](Remove(prog, prog_set))
              )
            )
        )
[]   [Card(prog_set) eq Succ(Succ(0))] -->
         choice prog: Program  [] [prog IsIn prog_set] -->
              (  channel1 !prog [prog = Prog1];
                 interface[command, channel1, channel2](Remove(prog, prog_set))
              [] channel2 !prog [prog = Prog2];
                 interface[command, channel1, channel2](Remove(prog, prog_set))
              )
endproc (*interface*)

process music_box[channel1, channel2, a, b, c, d] : noexit :=
      channel1 ?p: Program [p = Prog1];
      a; d; music_box[channel1, channel2, a, b, c, d]
  []  channel2 ?p: Program [p = Prog2];
      b; c; d; music_box[channel1, channel2, a, b, c, d]
endproc (*music_box*)
```

Let us look at the process `interface` initialized with the empty program set, i.e.

`interface[command, channel1, channel2]({}).`

So we' d like to know whether it's possible to split the interface process into two subprocesses `interface1` and `interface2` (that handle separately the requests of `Prog1` by the user with `Psw1`, and of `Prog2` by the user with `Psw2`) such that `interface[command, channel1, channel2]({})` and `interface1` ||| `interface2` are equivalent. To answer this question we have first to compute the semantic (i.e. a transition system) of the interface process, denoted by T. Afterwards, we apply our method on $Proc(T, S\_f(q))$ (see definition 2.20) to obtain $interface1$ and $interface2$ if such a solution does exist.

The semantic of a full LOTOS process is a transition system that is derived with the rules given in [ISO89]. Deriving the interface's transition system with these rules we obtain figure 1. Thereby, the states in T are numerated with $1, 2, 3$ and 4. Since T is finite we obtain with the function $Proc(\ldots)$:

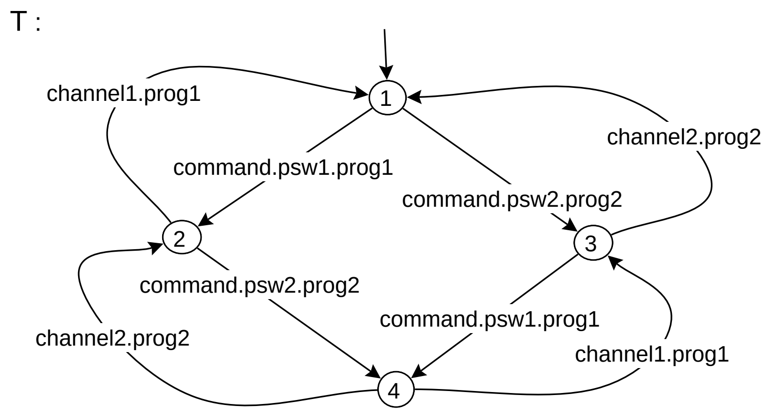


Figure 1: Transition system of the interface

$$
\begin{aligned}
Proc(T, S\_f(1)) &= command.psw1.prog1; S\_f(2) \,[]\, command.psw2.prog2; S\_f(3) \\
&= command.psw1.prog1; S\_2 \,[]\, command.psw2.prog2; S\_3
\end{aligned}
$$

and $PE(T, f)$ consists of the following process instantiations:

$$
\begin{aligned}
S\_f(1) = S\_1 &= command.psw1.prog1; S\_2 \,[]\, command.psw2.prog2; S\_3 \\
S\_f(2) = S\_2 &= channel1.prog1; S\_1 \,[]\, command.psw2.prog2; S\_4 \\
S\_f(3) = S\_3 &= channel2.prog2; S\_1 \,[]\, command.psw1.prog1; S\_4 \\
S\_f(4) = S\_4 &= channel1.prog1; S\_3 \,[]\, channel2.prog2; S\_2
\end{aligned}
$$

where $f(i) = i$ for $i = 1, \ldots, 4$. Let be $B = Proc(T, S\_f(1))$ and $\mathcal{P} = PE(T, f)$. We try now to split B into two processes (i.e. `interface1` and `interface12`) according to $\mathcal{A} = \{A_1, A_2\}$ with

$$A_1 = \{command.psw1.prog1, channel1.prog1\}$$

and $A_2 = \{command.psw2.prog2, channel2.prog2\}$. As we know from example 3.1 that such solution exists we obtain the following similar result:

1. $C_1 = Pj1(B, A_1, 1) = command.psw1.prog1; (S\_2)\_1$
   and $\mathcal{P}_1 = Del(Pj2(\mathcal{P}, A_1, 1), Rpv(C_1, Pj2(\mathcal{P}, A_1, 1)))$ consists of

- $(S\_1)\_1 = command.psw1.prog1; (S\_2)\_1$
- $(S\_2)\_1 = channel1.prog1; (S\_1)\_1$

2. $C_2 = Pj1(B, A_2, 2) = command.psw2.prog2; (S\_3)\_2$
   and $\mathcal{P}_2 = Del(Pj2(\mathcal{P}, A_2, 2), Rpv(C_2, Pj2(\mathcal{P}, A_2, 2)))$ consists of
   - $(S\_1)\_2 = command.psw2.prog2; (S\_3)\_2$
   - $(S\_3)\_2 = channel2.prog2; (S\_1)\_2$

Because C_1 (resp. C_2) and $(S\_1)\_1$ (resp. $(S\_1)\_2$) are equivalent, `interface1` (resp. `interface12`) can be identified with $(S\_1)\_1$ (resp. $(S\_1)\_2$). Thus we can now give an equivalent specification of the interface process as follows:

```
process interface[command, channel1, channel2] : noexit :=
   interface1[command, channel1] ||| interface2[command, channel2]
where
  process interface1[command, channel1] : noexit :=
     command.psw1.prog1; channel1.prog1; interface1[command, channel1]
  endproc

  process interface2[command, channel2] : noexit :=
     command.psw2.prog2; channel2.prog2; interface2[command, channel2]
  endproc
endproc
```

In [BvdLV95] this specification is computed just by hand, i.e. without using the methodological approach, because B is recursive.

5 Correctness proof

This section gives the proof of the theorem 3.1. We first need some lemmata.

Lemma 5.1 *Let $B_i, C_k \in \mathcal{L}$ with $i = 1, 2, 3$ and $k = 1, 2$, and $\mathcal{P}, \mathcal{Q} \in Env_{\mathcal{L}}$. Then*

1. $B_1 \;|||\; (B_2 \,|||\, B_3) \sim_{\mathcal{P},\mathcal{P}} (B_1 \,|||\, B_2) \;|||\; B_3$.
2. $B_1 \,|||\, B_2 \sim_{\mathcal{P},\mathcal{P}} B_2 \,|||\, B_1$
3. $B_1 \,|||\, B_2 \sim_{\mathcal{P},\mathcal{Q}} C_1 \,|||\, C_2$, *if $B_k \sim_{\mathcal{P},\mathcal{Q}} C_k$ holds.*

Proof: For 1.: Let $C = B_1 \;|||\; (B_2 \,|||\, B_3)$, $D = (B_1 \,|||\, B_2) \;|||\; B_3$ and $R \subseteq (\mathcal{L} \times \mathcal{L})$ with

$$R = \{(q_1 \,|||(q_2 \,|||\, q_3), (q_1 \,|||\, q_2) \,|||\, q_3) \mid q_i \in Re(B_i, \mathcal{P})\}.$$

It is easy to show that R is a bisimulation (relating to $\mathcal{P}$). Since $(C, D) \in R$ holds we have $C \sim_{\mathcal{P},\mathcal{P}} D$. For 2.: Trivial. For 3.: Let R_k be a bisimulation between $\mathcal{OS}(B_k, \mathcal{P})$ and $\mathcal{OS}(C_k, \mathcal{P})$. Define $R \subseteq (\mathcal{L} \times \mathcal{L})$ as follows:

$$R = \{(q_1 \,|||\, q_2, q_3 \,|||\, q_4) \mid (q_1, q_3) \in R_1 \wedge (q_2, q_4) \in R_2\}$$

It is easy to show that R is a bisimulation between $\mathcal{OS}(B_1 \,|||\, B_2, \mathcal{P})$ and $\mathcal{OS}(C_1 \,|||\, C_2, \mathcal{Q})$. Thus we have $B_1 \,|||\, B_2 \sim_{\mathcal{P},\mathcal{Q}} C_1 \,|||\, C_2$. $\square$

Lemma 5.2 *Let $B \in \mathcal{L}_{seq}$, $\mathcal{P} \in Env_{seg}$ and $(P, C) \in \mathcal{P}$. Then*

1. *$Pv(B) \subseteq PN(\mathcal{P})$, if B is closed in P.*
2. *$Pv(C) \subseteq PN(\mathcal{P})$, if B is closed in P.*
3. *$Pv(Pj1(B, A, i)) \subseteq \{P\_i \mid P \in Pv(B)\}$.*

Proof: For 1.: From $Pv(B) \subseteq Rpv(B, \mathcal{P})$ and definition 2.17 it follows $Pv(B) \subseteq PN(\mathcal{P})$. For 2.: Since $P \in Rpv(B, \mathcal{P})$ holds we have $Pv(C) \subseteq Rpv(B, \mathcal{P})$ because of definition 2.16. Thus $Pv(C) \subseteq PN(\mathcal{P})$. For 3.: Structural induction on B. Easy and omitted. $\square$

Lemma 5.3 *Let $B \in \mathcal{L}_{seq}$ be closed in $\mathcal{P} \in Env_{seq}$, $A \subseteq Act(B, \mathcal{P})$, $C = Pj1(B, A, i)$ and $\mathcal{Q} = Pj2(\mathcal{P}, A, i)$. Then*

$$\mathcal{F}2^k_{C,\mathcal{Q}}(Pv(C)) \subseteq PN(\mathcal{Q}).$$

Proof: Mathematical induction on k

- $k = 0$

  By lemma 5.2(3) $Pv(C) \subseteq \{P\_i \mid P \in Pv(B)\}$ holds and with Lemma 5.2(1) we have $Pv(C) \subseteq \{P\_i \mid P \in PN(\mathcal{P})\}$. From that it follows $Pv(C) \subseteq PN(\mathcal{Q})$.

- Induction hypothesis for $k \Leftrightarrow 1$

- Induction step:

$$\begin{array}{rcl}
\mathcal{F}2^k_{C,\mathcal{Q}}(Pv(C)) & = & \mathcal{F}2_{C,\mathcal{Q}}(\mathcal{F}^{k-1}_{C,\mathcal{Q}}(Pv(C))) \\
& = & \mathcal{F}2^{k-1}_{C,\mathcal{Q}}(Pv(C)) \cup \\
& & (\bigcup\{X \mid \exists P \in \mathcal{F}2^{k-1}_{C,\mathcal{Q}}(Pv(C)) : \exists B' : (P, B') \in \mathcal{Q} \wedge X = Pv(B')\}) \\
& \subseteq & \mathcal{F}2^{k-1}_{C,\mathcal{Q}}(Pv(C)) \cup (\bigcup\{X \mid \exists (P, B') \in \mathcal{Q} \wedge X = Pv(B')\}) \\
& \subseteq & \mathcal{F}2^{k-1}_{C,\mathcal{Q}}(Pv(C)) \cup \\
& & (\bigcup\{X \mid \exists (P, B') \in \mathcal{P} : X = \{P'_i \mid P' \in Pv(B')\}\}), \\
& & \text{by Lemma 5.2(3)} \\
& \subseteq & PN(\mathcal{Q}), \text{ by induction hypothesis and since for } (P, B') \in \mathcal{P} \\
& & \text{we have } Pv(B') \subseteq PN(\mathcal{P}) \text{ because of lemma 5.2(2).} \\
& & \text{Thus } \{P'_i \mid P' \in Pv(B')\} \subseteq PN(\mathcal{Q}).
\end{array}$$

$\square$

Lemma 5.4 *Let $B \in \mathcal{L}_{seq}$ and $\mathcal{P} \in Env_{seq}$. Then*

1. *$Act(B) = \{g \mid \exists B' : B \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'\}$.*
2. *$Act(B, \mathcal{P}) = \{g \mid \exists B', B'' \in Re(B, \mathcal{P}) : B' \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B''\}$, if B is closed in $\mathcal{P}$.*

Proof: For 1.: Structural induction on B. Easy and omitted. For 2.: Let

$$M = \{g \mid \exists B', B'' \in Re(B, \mathcal{P}) : B' \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B''\}.$$

- '$\subseteq$':

  For $g \in Act(\mathcal{P})$ we have $\exists (P, B') \in \mathcal{P} : g \in Act(B')$. By 1. $g \in \{a \,|\, \exists B'' : B' \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B''\}$ holds and consequently $g \in \{a \,|\, \exists B'' : P \overset{a}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B''\}$. Since $P \in Re(B, \mathcal{P})$ holds ($Re(B, \mathcal{P}) = PN(\mathcal{P}) \cup \{B\}$ because B is closed in $\mathcal{P}$) it follows $g \in M$. By analogical reasoning the same is true for $g \in Act(B)$.

- '$\supseteq$':

  For $g \in M$ we have $\exists B', B' \in Re(B, \mathcal{P}) : B' \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B''$. As $Re(B, \mathcal{P}) = PN(\mathcal{P}) \cup \{B\}$ holds it follows $B' \in PN(\mathcal{P})$ or $B' = B$. If $B' \in PN(\mathcal{P})$ then $\exists C : (B', C) \in \mathcal{P} \wedge C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B''$. By 1. $g \in Act(C)$ holds and therefore $g \in Act(\mathcal{P})$. If $B' = B$ then by analogical reasoning we have $g \in Act(B)$. $\square$

Lemma 5.5 *Let $B \in \mathcal{L}_{seq}$, $\mathcal{P}, \mathcal{Q} \in Env_{seq}$ and $P \in PN(\mathcal{P})$. Then*

1. $\forall g : \forall B' : B \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B' \Leftrightarrow B \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} B'$.
2. $\forall g : \forall B' : P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B' \Leftrightarrow P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} B'$, *if* $\mathcal{P} \subseteq \mathcal{Q}$.
3. $Re(B, \mathcal{P}) = Re(B, \mathcal{Q})$, *if B is closed in $\mathcal{P}$ and* $\mathcal{P} \subseteq \mathcal{Q}$.
4. $B \sim_{\mathcal{P},\mathcal{Q}} B$, *if B is closed in $\mathcal{P}$ and* $\mathcal{P} \subseteq \mathcal{Q}$.

Proof: For 1.: Structural induction on B. Easy and omitted. For 2.:

a) '$\Rightarrow$': From $P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'$ it follows $\exists C : (P, C) \in \mathcal{P} \wedge C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'$. By 1. $C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} B'$ holds and hence $P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} B'$.

b) '$\Leftarrow$': By analogy with a) we have $\exists C : (P, C) \in \mathcal{Q} \wedge C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} B'$. We show that $(P, C) \in \mathcal{P}$. Assume $(P, C) \notin \mathcal{P}$. Then we have $(P, C) \neq (P, C')$ with $(P, C') \in \mathcal{P}$ because of $P \in PN(\mathcal{P})$. Since $\mathcal{Q}$ is a process environment we have $P \neq P$ and consequently a contradiction. Thus $(P, C) \in \mathcal{P}$. By 1. this implies $P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'$.

For 3.: Evidently, it suffices to show that $\mathcal{F}1^k_{B,\mathcal{P}}(\{B\}) = \mathcal{F}1^k_{B,\mathcal{Q}}(\{B\})$. We show this with mathematical induction on k.

- $k = 0$. Trivial.
- Induction hypothesis for $k \Leftrightarrow 1$.
- Induction step: Let $L = \mathcal{F}1^{k-1}_{B,\mathcal{P}}(\{B\})$ and $L' = \mathcal{F}1^{k-1}_{B,\mathcal{Q}}(\{B\})$. So we have

$$\begin{aligned}
\mathcal{F}1^k_{B,\mathcal{P}}(\{B\}) &= L \cup \{C' \,|\, \exists C \in L : \exists g : C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} C'\} \\
&= L' \cup \{C' \,|\, \exists C \in L' : \exists g : C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} C'\}, \text{ by induction hypothesis} \\
&= L' \cup \{C' \,|\, \exists C \in L' : \exists g : C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} C'\}, \text{ since } B \text{ is closed } \textit{in } \mathcal{P} \\
& \text{ and thus either } C \in PN(\mathcal{P}) \text{ or } C = B \text{ holds.} \\
& \text{ By 1. and 2. we have } C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} C'. \\
&= \mathcal{F}1^k_{B,\mathcal{Q}}(\{B\})
\end{aligned}$$

Zu 4.: With 1, 2 und 3 it is easy to show that $R = \{(q, q) \,|\; q \in Re(B, \mathcal{P})\}$ is a bisimulation between $TS(B, \mathcal{P})$ and $TS(B, \mathcal{Q})$. Since $(B, B) \in R$ holds we have $B \sim_{\mathcal{P},\mathcal{Q}} B$. $\square$

Lemma 5.6 *Let $B \in \mathcal{L}$ and $\mathcal{P} \in Env_{\mathcal{L}}$. Let $TS(B, \mathcal{P}) \sim T$, where $T = (Q, \Leftrightarrow\!\!\rightarrow, q_0)$, and $R \subseteq (Re(B, \mathcal{P}) \times Q)$ be a bisimulation with $(B, q_0) \in R$ belonging to it. Then*

$$\forall C \in Re(B, \mathcal{P}) : \exists q \in Q : (C, q) \in R.$$

Proof: Let $C \in Re(B, \mathcal{P})$. Then we have by Korollar 2.1 $\exists k : C \in \mathcal{F}1^k_{B,\mathcal{P}}(\{B\})$. With mathematical induction on k it is easy to show that there are $C_i \in Re(B, \mathcal{P})$ and $g_i \in \mathcal{G}$ with $i = 0, \ldots, k \Leftrightarrow 1$ such that the following holds:

$$B = C_0 \stackrel{g_0}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} C_1 \stackrel{g_1}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} C_2 \cdots C_{k-1} \stackrel{g_{k-1}}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} C_k = C$$

Since R is a bisimulation there exists $q_h \in Q$ with $h = 0, \ldots, k \Leftrightarrow 1$ such that

$$q_0 \stackrel{g_0}{\Leftrightarrow\!\!\rightarrow} q_1 \stackrel{g_1}{\Leftrightarrow\!\!\rightarrow} q_2 \cdots q_{k-1} \stackrel{g_{k-1}}{\Leftrightarrow\!\!\rightarrow} q_k$$

and $(C_i, q_i) \in R$ holds. Hence it follows $\exists q \in Q : (C, q) \in R$. $\square$

Lemma 5.7 *Let $C_1, C_2 \in \mathcal{L}$, $\mathcal{P} \in Env_{\mathcal{L}}$, $D = C_1 \,|||\, C_2$. Then $Re(D, P) \subseteq M$, where*

$$M = \{p \,|||\, q \mid p \in Re(C_1, \mathcal{P}), q \in Re(C_2, \mathcal{P})\}.$$

Proof: We show with mathematical induction on k that $\mathcal{F}1^k_{D,\mathcal{P}}(\{D\}) \subseteq M$.

- $k = 0$. Trivial.
- Induction hypothesis for $k \Leftrightarrow 1$.
- Induction step: Let $L = \mathcal{F}1^{k-1}_{D,\mathcal{P}}(\{D\})$. We obtain

$$\begin{aligned} \mathcal{F}1^k_{D,\mathcal{P}}(\{D\}) &= L \cup \{C' \mid \exists C \in L : \exists g : C \stackrel{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} C'\} \\ &\subseteq M \cup \{C' \mid \exists C \in L : \exists g : C \stackrel{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} C'\}, \text{ by induction hypothesis} \\ &\subseteq M, \text{ since } C' \in M \text{ holds.} \end{aligned}$$

From corollar 2.1 it follows $Re(D, P) \subseteq M$. $\square$

Lemma 5.8 *Let $C_i \in \mathcal{L}_{seq}$ be closed in $\mathcal{P}_i \in Env_{seq}$ with $i = 1, \ldots, n$. Let $\mathcal{P} \in Env_{seq}$ with $\mathcal{P}_i \subseteq \mathcal{P}$, $D = |||_{i=1}^n C_i$ and $q \in Re(D, \mathcal{P})$. Then*

$$\forall q' : \forall g : q \stackrel{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} q' \Longrightarrow g \in \bigcup_{i=1}^{n} Act(C_i, \mathcal{P}_i).$$

Proof: We show with mathematical induction on n.

- $n = 1$. It follows from lemma 5.4(2).
- Induction hypothesis for $n \Leftrightarrow 1$.
- Induction step: Since $|||$ is associative and commutative relating to $\sim$ we can write $D \sim_{\mathcal{P},\mathcal{P}} C_n \,|||\, (|||_{i=1}^{n-1} C_i)$. Let $C = C_n \,|||\, C'$ and $C' = |||_{i=1}^{n-1} C_i$. From lemma 5.6 it follows $\exists p, p' \in Re(C, \mathcal{P}) : p \stackrel{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} p'$. By Lemma 5.7 we have $p = p_1 \,|||\, p_2$ with $p_1 \in Re(C_n, \mathcal{P})$ and $p_2 \in Re(C', \mathcal{P})$. Thus

  1) either $\exists p_1' : p_1 \stackrel{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} p_1'$ or

2) $\exists p_2' : p_2 \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} p_2'$ holds.

For 1): Since by lemma 5.5(3) $Re(C_n, \mathcal{P}) = Re(C_n, \mathcal{P}_n)$ holds and consequently by lemma 5.5(1,2) $p_1 \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}_n} p_1'$, we have by lemma 5.4(2) $g \in Act(C_n, \mathcal{P}_n)$.

For 2): $g \in \bigcup_{i=1}^{n-1} Act(C_i, \mathcal{P}_i)$ holds by induction hypothesis.

This concludes that $g \in \bigcup_{i=1}^{n} Act(C_i, \mathcal{P}_i)$. □

Definition 5.1 *Let $B \in \mathcal{L}$, $\mathcal{P} \in Env_{\mathcal{L}}$ and $A \subseteq Act(B, \mathcal{P})$. Then $Res(B, A, \mathcal{P}) := (\mathcal{L}, \Leftrightarrow\!\!\!\rightarrow, B)$ is a transition system where $\Leftrightarrow\!\!\!\rightarrow \; = \; \Leftrightarrow\!\!\!\rightarrow_{\mathcal{P}} \cap (\mathcal{L} \times A \times \mathcal{L})$.* □

Lemma 5.9 *Let $B, C_i \in \mathcal{L}_{seq}$, $\mathcal{P}, \mathcal{Q}_i \in Env_{seq}$ and $i = 1, \ldots, n$. Let*

- $C = |||_{i=1}^{n} C_i$,
- *C_i be closed in $\mathcal{Q}_i$,*
- *$PN(\mathcal{Q}_i) \cap PN(\mathcal{Q}_j) = \emptyset$ for $i \neq j$, $\mathcal{Q} = \bigcup_{i=1}^{n} \mathcal{Q}_i$,*
- *$A_i = Act(C_i, \mathcal{Q}_i)$, $A_i \cap A_j = \emptyset$ for $i \neq j$ and*
- $B \sim_{\mathcal{P},\mathcal{Q}} C$.

Then $TS(C_k, \mathcal{Q}) \sim Res(B, A_k, \mathcal{P})$ for $k = 1, \ldots, n$.

Proof: Since $|||$ is associative and commutative relating to $\sim$ we have $B \sim_{\mathcal{P},\mathcal{Q}} D$, where $D = C_k \,|||\, D'$ and $D' = (|||_{i=1, i\neq k}^{n} C_i)$. Let R be a bisimulation between $TS(D, \mathcal{Q})$ and $TS(B, \mathcal{P})$, and $Res(B, A_k, \mathcal{P}) = (\mathcal{L}, \Leftrightarrow\!\!\!\rightarrow, B)$. Define $S \subseteq (Re(C_k, \mathcal{Q}) \times \mathcal{L})$ as follows:

$$S = \{(p, q) \,|\, p \in Re(C_k, \mathcal{Q}) \wedge \exists p' : p \,|||\, p' \in Re(D, \mathcal{Q}) \wedge (p \,|||\, p', q) \in R\}$$

We show that S is a bisimulation between $TS(C_k, \mathcal{Q})$ and $Res(B, A_k, \mathcal{P})$. Let $(p, q) \in S$.

1. Firstly, we have $\exists p'' : (p \,|||\, p'', q) \in R$. If $p \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{Q}} p'$ then $p \,|||\, p'' \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{Q}} p' \,|||\, p''$. From that it follows $\exists q' : q \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} q' \wedge (p' \,|||\, p'', q') \in R$. Since because of lemma 5.5(1,2,3) $p \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{Q}_k} p'$ holds we have by lemma 5.4(2) $g \in A_k$. Thus $q \overset{g}{\Leftrightarrow\!\!\!\rightarrow} q'$. It is obvious that $(p', q') \in S$.

2. If $q \overset{g}{\Leftrightarrow\!\!\!\rightarrow} q'$ then $q \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} q'$. Thus $\exists z : p \,|||\, p'' \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{Q}} z \;\wedge\; (z, q') \in R$. Since $g \in A_k$ and $A_k \cap A_j = \emptyset$ for $j \neq k$ holds we can follow because of lemma 5.8 that $p \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{Q}} p'$, and consequently $z = p' \,|||\, p''$. From that it follows $(p', q') \in S$.

This concludes that S is a bisimulation. Since $(D, B) \in R$ holds we have $(C_k, B) \in S$. Thus $TS(C_k, \mathcal{Q}) \sim Res(B, A_k, \mathcal{P})$. □

Lemma 5.10 *Let $B \in \mathcal{L}_{seq}$, $\mathcal{P}, \mathcal{Q} \in Env_{seq}$, $A \subseteq \mathcal{G}$, $g \in A$ and $i \in I\!N$. Then*

$$\forall B' : B \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{P}} B' \Leftrightarrow Pj1(B, A, i) \overset{g}{\Leftrightarrow\!\!\!\rightarrow}_{\mathcal{Q}} B'\_i.$$

Proof: Structural induction on B. □

Lemma 5.11 *Let*

- *$B \in \mathcal{L}_{seq}$ be closed in $\mathcal{P} \in Env_{seq}$.*
- $A \subseteq Act(B, \mathcal{P})$.

- $C = Pj1(B, A, i)$ *for* $i \in I\!N$.
- $\mathcal{Q} = Del(\mathcal{P}', Rpv(C, \mathcal{P}'))$, *where* $\mathcal{P}' = Pj2(\mathcal{P}, A, i)$.

Then $Res(B, A, \mathcal{P}) \sim \mathcal{OS}(C, \mathcal{Q})$.

Proof: Let $Res(B, A, \mathcal{P}) = (\mathcal{L}, \Leftrightarrow\!\!\rightarrow, B)$ and $R \subseteq (\mathcal{L} \times \mathcal{L})$ with

$$R = \{(P, P\_i) \mid P \in PN(\mathcal{P})\} \cup \{(B, C)\}.$$

We show that R is a bisimulation between $Res(B, A, \mathcal{P})$ and $\mathcal{OS}(C, \mathcal{Q})$. Let $(p, q) \in R$. There are two cases:

1. $p = B$ and $q = C$
   (a) If $B \overset{g}{\Leftrightarrow\!\!\rightarrow} B'$ then $B \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'$. By lemma 5.10 we have $C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} B'\_i$. Since B is closed in $\mathcal{P}$ and $B' \in Re(B, \mathcal{P})$ holds we have by proposition 2.1 $B' \in PN(\mathcal{P}) \cup \{B\}$. Since by lemma 2.6 B' is a process name, i.e. $B' \in PN(\mathcal{P})$, we can follow $(B', B'\_i) \in R$.
   (b) If $C \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} C'$ then by lemma 5.10 $B \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} B'$ (i.e. $B \overset{g}{\Leftrightarrow\!\!\rightarrow} B'$) and $C' = B'\_i$. By analogy with (a) we have $B' \in PN(\mathcal{P})$ und hence $(B', C') \in R$.
2. $p = P$ and $q = P\_i$, where $P \in PN(\mathcal{P})$
   (a) If $P \overset{g}{\Leftrightarrow\!\!\rightarrow} P'$ then $P \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'$. Thus $\exists B' : (P, B') \in \mathcal{P} \wedge B' \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{P}} P'$. Let $D = Pj1(B', A, i)$. By Lemma 5.10 $D \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} P'\_i$ holds und thus $P\_i \overset{g}{\Leftrightarrow\!\!\rightarrow}_{\mathcal{Q}} P'\_i$. By analogy with 1(a) we obtain $(P', P'\_i) \in R$.
   (b) By analogy with 2(a). □

Now we are prepared to prove the theorem 3.1.

Proof of theorem 3.1:

- For a):
  1. $C_i \in \mathcal{L}_{seq}$ for $i = 1, \ldots, n$. Trivial.
  2. C_i is closed in $\mathcal{P}_i$ for $i = 1, \ldots, n$, i.e. $Rpv(C_i, \mathcal{P}_i) = PN(\mathcal{P}_i)$.
     We first show that the following holds:

     $$Rpv(C_i, Pj2(\mathcal{P}, A_i, i)) = PN(\mathcal{P}_i)$$

     '$\supseteq$': Trivial.
     '$\subseteq$': Let $P \in Rpv(C_i, \mathcal{Q})$, where $\mathcal{Q} = Pj2(\mathcal{P}, A_i, i)$. Because of corollar 2.2 we have $\exists k \in I\!N_0 : P \in \mathcal{F}2^k_{C_i,\mathcal{Q}}(Pv(C_i))$ and by lemma 5.3 $P \in PN(\mathcal{Q})$. That means $\exists B' : (P, B') \in \mathcal{Q}$. Thus $P \in PN(\mathcal{P}_i)$.
     We show now that $Rpv(C_i, \mathcal{P}_i) = Rpv(C_i, \mathcal{Q})$. Obviously, it is sufficient to show that

     $$\mathcal{F}2^k_{C_i,\mathcal{P}_i}(Pv(C_i)) = \mathcal{F}2^k_{C_i,\mathcal{Q}}(Pv(C_i)).$$

     We show this with mathematical induction on k.
     $k = 0$. Trivial.
     Induction hypothesis for $k \Leftrightarrow 1$.

Induction step: Let $L = \mathcal{F}2^{k-1}_{C_i,\mathcal{P}_i}(Pv(C_i))$ and $L' = \mathcal{F}2^{k-1}_{C_i,\mathcal{Q}}(Pv(C_i))$. We obtain:

$$\begin{aligned}
\mathcal{F}2^{k}_{C_i,\mathcal{P}_i}(Pv(C_i)) &= L \cup (\bigcup\{X \mid \exists P \in L : \exists B' : (P, B') \in \mathcal{P}_i \wedge X = Pv(B')\}) \\
&= L' \cup (\bigcup\{X \mid \exists P \in L' : \exists B' : (P, B') \in \mathcal{P}_i \wedge X = Pv(B')\}), \\
&\quad \text{by induction hypothesis} \\
&= L' \cup (\bigcup\{X \mid \exists P \in L' : \exists B' : (P, B') \in \mathcal{Q} \wedge X = Pv(B')\}), \\
&\quad \text{since from } P \in L' \text{ it follows } P \in Rpv(C_i, \mathcal{Q}) \\
&\quad \text{and from } (P, B') \in \mathcal{Q} \text{ it follows } (P, B') \in \mathcal{P}_i. \\
&= \mathcal{F}2^{k}_{C_i,\mathcal{Q}}(Pv(C_i))
\end{aligned}$$

3. $Act(C_i, \mathcal{P}_i) = A_i$
   '$\subseteq$': Trivial.
   '$\supseteq$': Let $g \in A_i$. Because B is closed in $\mathcal{P}$ we have by lemma 5.4(2)

   $$\exists B', B'' \in Re(B, \mathcal{P}) : B' \overset{g}{\Leftrightarrow}_{\mathcal{P}} B''.$$

   As $B \sim_{\mathcal{P},\mathcal{P}'} Inv(B, \mathcal{A})$ holds and $|||$ is associative we can follow $B \sim_{\mathcal{P},\mathcal{P}'} D$, where $D = C_i \,|||\, (|||_{k \in M}\, C_k)$ and $M = \{1, \ldots, n\} \setminus \{i\}$. By lemma 5.6 we have $\exists q, q' \in Re(D, \mathcal{P}') : q \overset{g}{\Leftrightarrow}_{\mathcal{P}'} q'$. By lemma 5.7 $q = q_1 \,|||\, q_3$ und $q' = q_2 \,|||\, q_4$ holds, where

   $$q_1, q_2 \in Re(C_i, \mathcal{P}') \text{ und } q_3, q_4 \in Re(\mathop{|||}_{k \in M} C_k, \mathcal{P}').$$

   From lemma 5.8 it follows $q_1 \overset{g}{\Leftrightarrow}_{\mathcal{P}'} q_2$ and because of lemma 5.5(1,2) $q_1 \overset{g}{\Leftrightarrow}_{\mathcal{P}_i} q_2$. By lemma 5.4(2) we have $g \in Act(C_i, \mathcal{P}_i)$.
4. $PN(\mathcal{P}_i) \cap PN(\mathcal{P}_i) = \emptyset$ for $i \neq j$. Trivial.
5. $B \sim_{\mathcal{P},\mathcal{P}'} C$ holds because of the assumption.

- For b):

  '$\Leftarrow$': It follows from a).

  '$\Rightarrow$': Let $D = |||_{i=1}^{n} D_i$ in $Q_1, \ldots, Q_n$ be a solution for B in $\mathcal{P}$ under A. By Lemma 5.9 and 5.11 we have $TS(D_i, \mathcal{Q}) \sim Res(B, A_i, \mathcal{P})$, where $\mathcal{Q} = \bigcup_{i=1}^{n} \mathcal{Q}_i$, and $Res(B, A_i, \mathcal{P}) \sim \mathcal{OS}(C_i, \mathcal{P}_i)$. Since by lemma 5.5(4) $C_i \sim_{\mathcal{P}_i,\mathcal{P}'} C_i$ holds we obtain $D_i \sim_{\mathcal{Q},\mathcal{P}'} C_i$. From lemma 5.1(3) it follows $D \sim_{\mathcal{Q},\mathcal{P}'} Inv(B, A)$, i.e. $B \sim_{\mathcal{P},\mathcal{P}'} Inv(B, A)$. □

6 Conclusion

The method presented in this paper is a generalization of the so-called inverse expansion introduced in [PHQ+92] in the case of pure interleaving. It transforms a finite-state process written in Basic LOTOS into more than two subprocesses running independently. The correctness property fulfilled by this transformation is the strong bisimulation equivalence according to [Mil89]. The method is seen as 'generalized' because it is also applicable to the class of recursive processes which is not treated in [PHQ+92].

Since both in [PHQ+92] and in this paper the set of gates of the subprocesses into which the given process has to be transformed must be known for the transformation, this work can be extended in the following direction: How can one obtain the subprocesses without using the set of gates where the number of the subprocesses should be maximal? In other words, is it possible to decompose the initial process into an equivalent process with a maximal degree of parallelism? For the class of deterministic processes the positive answer for this question can be found in [Do96a]. For the class of nonrecursive nondeterministic processes this question has been answered in [Do96b]. The treatment of recursive nondeterministic processes is, to our best knowledge, still an open problem.

ACKNOWLEDGMENT: I am grateful to Markus Roggenbach for a critical reading of this paper.

A The Kleene's theorem

In this appendix we recall the Kleene's theorem from the domain theory that is needed in section 2. For an introduction to this topic reader are referred to [Win93].

Definition A.1 *Let D be a set and $\sqsubseteq \subseteq (D \times D)$ a relation which is:*

1. *refexive:* $\forall d \in D : d \sqsubseteq d$
2. *transitive:* $\forall d, d', d'' \in D : d \sqsubseteq d' \wedge d' \sqsubseteq d'' \Longrightarrow d \sqsubseteq d''$
3. *antisymmetric:* $\forall d, d' \in D : d \sqsubseteq d' \wedge d' \sqsubseteq d \Longrightarrow d = d'$

Then $\sqsubseteq$ is called a partial order and $(D, \sqsubseteq)$ a partial order set (poset). $\square$

Definition 1.2 *Let $(D, \sqsubseteq)$ be a poset and $D' \subseteq D$.*

1. *$d \in D$ is an upper bound of D' if $\forall d' \in D' : d' \sqsubseteq d$.*
2. *$d \in D$ is a least upper bound of D' (l.u.b.), denoted by $\sqcup D'$, if d is an upper bound and $\forall d'' \in D : d''$ is an upper bound of $D' \Longrightarrow d \sqsubseteq d''$.*
3. *D' is a chain, if $D' \neq \emptyset$ and $\forall d, d' \in D' : d \sqsubseteq d' \vee d' \sqsubseteq d$.* $\square$

Note when $D = \{d_i \mid i \in I\}$ for an indexing set then we also write $\sqcup D$ as $\sqcup_{i \in I} d_i$.

Definition 1.3 *$(D, \sqsubseteq, \perp)$ is a complete partial order (c.p.o.) if $(D, \sqsubseteq)$ is a poset, each chain in D has a l.u.b. and $\perp$ is a least element in D (i.e. $\forall d \in D : \perp \sqsubseteq d$).* $\square$

Definition 1.4 *Let $(D, \sqsubseteq, \perp)$ and $(D', \sqsubseteq', \perp')$ be c.p.o. and $F : D \to D'$. F is continous if for each chain E in D the following holds:*

$$F(\bigsqcup E) = \bigsqcup F(E).$$

Thereby, $F(E)$ stands for $\{F(e) \mid e \in E\}$. $\square$

Theorem 1.1 (Kleene's Theorem) *Let $(D, \sqsubseteq, \perp)$ be a c.p.o. and $F : D \to D$ continous. Then*

$$\bigsqcup_{n \in I\!N_0} F^n(\perp)$$

is the least fixed point of F. Thereby, $d \in D$ is a fixed point of F if $F(d) = d$. $\square$

References

[BB87] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. In *Computer Networks and ISDN Systems 14*, pages 25–59. Elsevier Science Publishers B.V. North-Holand, 1987.

[BvdLV95] T. Bolognesi, J. van de Lagemaat, and C. Visser. *LOTOSphere: Software Development with LOTOS*. Kluwer Academic Publisher, 1995.

[CPS93] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A Semantics-Based Tool for the Verification of concurrent Systems. *ACM Transactions on Programming Languages and Systems*, 14(1):36–72, January 1993.

[Do96a] H. T. Do. Maximal process decomposing with recursion. Internal note, October 1996.

[Do96b] H. T. Do. Maximale Zerlegung von parallelen Prozessen ohne Rekursion. Manuskripte der Reihe Informatik Nr. 12/96, Fakultät für Mathematik und Informatik der Universität Mannheim, Juni 1996.

[ISO89] ISO/BS. *ISO 8807 - Information processing systems - Open Systems Interconnection - LOTOS- A formal description technique based on the temporal ordering of observational behaviour*. BSI, 1989.

[Jan85] R. Janicki. Transforming sequential systems into concurrent systems. In *Theoretical Computer Science 36*, pages 27–58. North-Holand, 1985.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.

[PHQ+92] S. Pav*ó*n, M. Hultström, J. Quemada, D. Frutos, and Y. Ortega. Inverse Expansion. In *Formal Description Technique IV*, pages 297–312. Elsevier Science Publishers B.V. North-Holand, 1992.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages, An Introdution*. The MIT Press, 1993.