On Unique Decomposition of Processes in the Applied π-Calculus

Jannik Dreier, Cristian Ene, Pascal Lafourcade, and Yassine Lakhnech

Université Grenoble 1, CNRS, Verimag, FRANCE
firstname.lastname@imag.fr

Abstract. Unique decomposition has been a subject of interest in process algebra for a long time (for example in BPP [2] or CCS [11, 13]), as it provides a normal form with useful cancellation properties. We provide two parallel decomposition results for subsets of the Applied π-Calculus: we show that any closed normed (i.e. with a finite shortest complete trace) process P can be decomposed uniquely into prime factors P_i with respect to strong labeled bisimilarity, i.e. such that $P \sim_l P_1|\ldots|P_n$. We also prove that closed finite processes can be decomposed uniquely with respect to weak labeled bisimilarity.

Keywords: Applied Pi Calculus, Unique Decomposition, Normal Form, Weak Bisimilarity, Strong Bisimilarity, Factorization, Cancellation

1 Introduction

Process Algebras or Calculi allow one to formally model and analyze distributed systems. Famous examples include the Calculus of Communicating Systems (CCS) due to Milner [10], or Basic Parallel Processes (BPP) [2]. These calculi contain basic operations such as emission and reception of messages as well as parallel composition or interleaving. In an extension to CCS, Milner, Parrow and Walker developed the π-Calculus [12], which also features channel passing and scope extrusion. Abadi and Fournet [1] subsequently proposed the Applied π-Calculus, a variant of the π-Calculus designed for the verification of cryptographic protocols. It additionally features equational theories and active substitutions.

In all of these process algebras the question of unique process decomposition naturally arises. Can we rewrite a process P as $P =^1 P_1|P_2|\ldots|P_n$, where $|$ is the parallel composition operator, and each P_i is prime in the sense that it cannot be rewritten as the parallel composition of two non-zero processes?

Such a decomposition provides a maximally parallelized version of a given program P. Additionally, it is useful as it provides a normal form, and a cancellation result in the sense that $P|Q = P|R$ implies $Q = R$. This is convenient in proofs, for example when proving the equivalence of different security notions

[1] Here $=$ does not designate syntactical identity, but rather some behavioral equivalence or bisimilarity relation.

in electronic voting [3]: one can show that coercion of one voter and coercion of multiple voters are (under some realistic hypotheses) equivalent. This simplifies the analysis of e-voting protocols, in particular some proofs of observational equivalence. If there is an efficient procedure to transform a process into its normal form, such a decomposition can also be used to verify the equivalence of two processes [5]: once the processes are in normal form, one only has to verify if the factors on both sides are identical.

However, existing results [2, 6, 11, 13] on the unique decomposition focus on "pure" calculi such as CCS or BPP or variants thereof. The Applied π-Calculus, as an "impure" variant of the π-Calculus designed for the verification of cryptographic protocols, has a more complex structure and semantics. For example, it features an equational theory to model cryptographic primitives, and active substitutions, i.e. substitutions that apply to all processes. This creates an element that is not zero, but still exhibits no transitions.

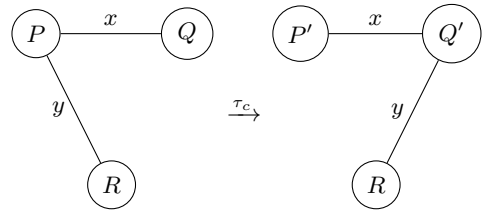
Additionally, the Applied π-Calculus inherits the expressive power of the π-Calculus including *channel* or *link passing* (sometimes also called *mobility*) and *scope extrusion*. Consider three parallel processes P, Q and R, where P and Q synchronize using an internal reduction τ_c, i.e. $P|Q|R \xrightarrow{\tau_c} P'|Q'|R$ (see Figure 1). Channel passing allows a process P to send a channel y he shares with R to process Q (Figure 1a). Scope extrusion arises for example when P sends a restricted channel y he shares with R to Q, since the scope after the transition includes Q' (Figure 1b). This is of particular importance for unique decomposition since two parallel processes sharing a restricted channel might not be decomposable and hence a simple reduction might "fuse" two prime factors.

1.1 Our Contributions

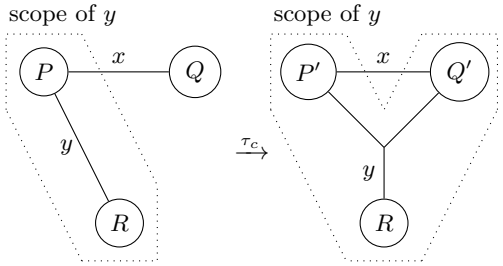
We provide two decomposition results for subsets of the Applied π-Calculus. In a first step, we prove that closed normed (i.e. with a finite shortest complete trace) processes can be uniquely decomposed with respect to strong labeled bisimilarity. In the second step we show that any closed finite (i.e. with a finite longest complete trace) process can be uniquely decomposed with respect to (weak) labeled bisimilarity, the standard bisimilarity notion in the Applied π-Calculus. Note that although we require the processes to be finite or normed, no further hypothesis is needed, i.e. they may use the full power of the calculus including channel passing and scope extrusion. As a direct consequence of the uniqueness of the decomposition, we also obtain cancellation results for both cases.

1.2 Outline of the Paper

In the next section, we recall the Applied π-Calculus, and establish different subclasses of processes. In Section 3 we provide our first unique decomposition result with respect to strong bisimilarity. In the next Section we show the second result w.r.t. weak bisimilarity. Then we discuss related work in Section 5 and conclude in Section 6. The full proofs can be found in our technical report [4].



(a) Channel/Link Passing



(b) Scope extrusion

Fig. 1: Features of the Applied π-Calculus

2 Preliminaries

In this section we recall briefly the Applied π-Calculus proposed by Abadi and Fournet [1] as an extension of the π-Calculus [12].

2.1 Applied π-Calculus

The Applied π-Calculus is a formal language for describing concurrent processes. The calculus consists of *names* (which typically correspond to data or channels), *variables*, and a *signature* Σ of *function symbols* which can be used to build *terms*. Functions typically include encryption and decryption (for example $\texttt{enc}(message, key)$, $\texttt{dec}(message, key)$), hashing, signing etc. Terms are correct combinations of names and functions, i.e. respecting arity and sorts. We distinguish the type "channel" from other *base* types. Equalities are modeled using an equational theory E which defines a relation $=_E$. A classical example, which describes the correctness of symmetric encryption, is $\texttt{dec}(\texttt{enc}(message, key), key) =_E message$.

Plain processes are constructed using the grammar depicted in Figure 2a. *Active* or *extended processes* are plain processes or active substitutions as shown in Figure 2b. Note that we do not include the "+"-operator which implements a nondeterministic choice, yet we can implement something similar using a restricted channel (see Example 4). For more details on encoding the operator with respect to different semantics, see [14, 15].

$P, Q :=$	plain processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction ("new")
$\texttt{if } M = N \texttt{ then } P \texttt{ else } Q$	conditional (M, N terms)
$\texttt{in}(u, x).P$	message input
$\texttt{out}(u, M).P$	message output

(a) Plain Processes

$A, B, P, Q :=$	active processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution

(b) Active/Extended Processes

Fig. 2: Process Grammars

The substitution $\{^M/_x\}$ replaces the variable x with a term M. Note that we do not allow two active substitutions to define the same variable, as this might lead to situations with unclear semantics. We denote by $fv(A)$, $bv(A)$, $fn(A)$, $bn(A)$ the free variables, bound variables, free names or bound names respectively.

As an additional notation we write $\nu S.A$ for $\nu s_1.\nu s_2 \ldots \nu s_n.A$ where $s_1, \ldots s_n$ are the elements of a set of variables and names S. By abuse of notation we sometimes leave out ".0" at the end of a process. We will also write A^k for $A| \ldots |A$ (k times), in particular $A^0 = 0$ as 0 is the neutral element of parallel composition.

The *frame* $\Phi(A)$ of an active process A is obtained by replacing all plain processes in A by 0. This frame can be seen as a representation of what is statically known to the environment about a process. The domain $dom(\Phi)$ of a frame Φ is the set of variables for which Φ defines a substitution. By abuse of notation, we also write $dom(A)$ to denote the domain of the frame $\Phi(A)$ of an active process A. Note that $dom(A) \subseteq fv(A)$, and that – as we cannot have two active substitutions for the same variable – $P = Q|R$ implies $dom(P) = dom(Q) \cup dom(R)$ and $dom(Q) \cap dom(R) = \emptyset$. A frame or process is *closed* if all variables are bound or defined by an active substitution. An evaluation context $C[\_]$ denotes an active process with a hole for an active process that is not under replication, a conditional, an input or an output.

The semantics of the calculus presupposes a notion of *Structural Equivalence* ($\equiv$), which is defined as the smallest equivalence relation on extended processes

that is closed under application of evaluation contexts, α-conversion on bound names and bound variables such that:

PAR-0	$A\|0 \equiv A$	
PAR-A	$A\|(B\|C) \equiv (A\|B)\|C$	
PAR-C	$A\|B \equiv B\|A$	
NEW-0	$\nu n.0 \equiv 0$	
NEW-C	$\nu u.\nu v.A \equiv \nu v.\nu u.A$	
NEW-PAR	$A\|\nu u.B \equiv \nu u.(A\|B)$	if $u \notin fn(A) \cup fv(A)$
REPL	$!P \equiv P\|!P$	
REWRITE	$\{^M/_x\} \equiv \{^N/_x\}$	if $M =_E N$
ALIAS	$\nu x.\, \{^M/_x\} \equiv 0$	
SUBST	$\{^M/_x\}\, \|A \equiv \{^M/_x\}\, \|A\, \{^M/_x\}$	

Note the contagious nature of active substitutions: by rule SUBST they apply to any parallel process.

Example 1. Consider the following running example, where x and y are variables, and c, d, k, l, m and n are names:

$$P_{ex} = \nu k.\nu l.\nu m.\nu d.\, (\{^l/_y\}\, |\mathtt{out}(c, enc(n, k))|\mathtt{out}(d, m)|\mathtt{in}(d, x).\mathtt{out}(c, x))$$

We have $dom(P_{ex}) = \{y\}$, $fv(P_{ex}) = \{y\}$, $bv(P_{ex}) = \{x\}$, $fn(P_{ex}) = \{n, c\}$, $bn(P_{ex}) = \{k, l, m, d\}$ and

$$\Phi(P_{ex}) = \nu k.\nu l.\nu m.\nu d.\, (\{^l/_y\}\, |0|0|0) \equiv \nu k.\nu l.\nu m.\nu d.\, (\{^l/_y\})$$

Internal Reduction ($\xrightarrow{\tau}$) is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts such that:

COMM	$\mathtt{out}(a, x).P \mid \mathtt{in}(a, x).Q \xrightarrow{\tau_c} P \mid Q$
THEN	$\mathtt{if}\ M = M\ \mathtt{then}\ P\ \mathtt{else}\ Q \xrightarrow{\tau_t} P$
ELSE	$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q \xrightarrow{\tau_e} Q$ for any ground terms such that $M \neq_E N$

Note that in accordance with the original notations [1], we sometimes omit the labels τ_c, τ_t and τ_e, and write $P \to P'$ for $P \xrightarrow{\gamma} P'$ with $\gamma \in \{\tau_c, \tau_t, \tau_e\}$.

Interactions of extended processes are described using labeled operational semantics ($\xrightarrow{\alpha}$), where α can be an input or an output of a channel name or variable of base type, e.g. $\mathtt{out}(a, u)$ where u is a variable or a name.

IN	$\mathtt{in}(a,x).P \xrightarrow{\mathtt{in}(a,M)} P\{^M/_x\}$
OUT-ATOM	$\mathtt{out}(a,u).P \xrightarrow{\mathtt{out}(a,u)} P$
OPEN-ATOM	$\dfrac{A \xrightarrow{\mathtt{out}(a,u)} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\mathtt{out}(a,u)} A'}$
SCOPE	$\dfrac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$
PAR	$\dfrac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$
STRUCT	$\dfrac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

Labeled *external transitions* are not closed under evaluation contexts. Note that a term M (except for channel names and variables of base type) cannot be output directly. Instead, we have to assign M to a variable, which can then be output. This is to model that the output of $\mathtt{enc}(m,k)$ (message m encrypted with key k) does not give the environment access to m.

Example 2. Consider our running example process P_{ex}. Using an internal reduction, we can execute the following transition:

$$\begin{aligned}
P_{ex} &= \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)).0 \mid \mathtt{out}(d,m).0 \mid \mathtt{in}(d,x).\mathtt{out}(c,x).0) \\
&\equiv \nu k.\nu l.\nu m.\nu d.(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \boldsymbol{\nu x.(\{^m/_x\})} \mid \mathtt{out}(d,m) \mid \\
&\quad \mathtt{in}(d,x).\mathtt{out}(c,x)) && \text{by PAR-0, ALIAS} \\
&\equiv \nu k.\nu l.\nu m.\nu d.(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \nu x.(\,\{^m/_x\} \mid \mathtt{out}(d, \boldsymbol{x}) \mid \\
&\quad \mathtt{in}(d,x).\mathtt{out}(c,x))) && \text{by SUBST, NEW-PAR} \\
&\xrightarrow{\tau_c} \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \nu x.\,(\{^m/_x\} \mid \mathtt{out}(c,x))) \\
&\equiv \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \mathtt{out}(c, \boldsymbol{m})) \\
&&& \text{by SUBST, ALIAS, NEW-PAR, PAR-0}
\end{aligned}$$

Similarly, we can also execute an external transition:

$$\begin{aligned}
P_{ex} &\equiv \nu k.\nu l.\nu m.\nu d.(\{^l/_y\} \mid \nu z.\,\{^{enc(n,k)}/_z\} \mid \mathtt{out}(c,z) \mid \mathtt{out}(d,m) \mid \\
&\quad \mathtt{in}(d,x).\mathtt{out}(c,x)) \\
&\xrightarrow{\nu z.\mathtt{out}(c,z)} \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \{^{enc(n,k)}/_z\} \mid \mathtt{out}(m,d) \mid \mathtt{in}(x,d).\mathtt{out}(x,c))
\end{aligned}$$

The following two definitions allow us to reason about the messages exchanged with the environment.

Definition 1 (Equivalence in a Frame [1]). *Two terms M and N are equal in the frame $\phi \equiv \nu\tilde{n}.\sigma$, written $(M = N)\phi$, if and only if $M\sigma =_E N\sigma$, and $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$.*

Note that any frame ϕ can be written as $\nu\tilde{n}.\sigma$ modulo structural equivalence, i.e. using rule NEW-PAR.

Definition 2 (Static Equivalence ($\approx_s$) [1]). *Two closed frames ϕ and ψ are statically equivalent, written $\phi \approx_s \psi$, when $dom(\phi) = dom(\psi)$ and when for all*

terms M and N we have $(M = N)\phi$ if and only if $(M = N)\psi$. Two extended processes A and B are statically equivalent ($A \approx_s B$) if their frames are statically equivalent.

The intuition behind this definition is that two processes are statically equivalent if the messages exchanged previously with the environment cannot be distinguished, i.e. all operations on both sides gave the same results.

2.2 Depth and Norm of Processes

We prove unique decomposition for different subsets of processes, namely finite and normed processes. This requires to formally define the length of process traces. Let $\mathbf{Int} = \{\tau_c, \tau_t, \tau_e\}$ denote the set of labels corresponding to internal reductions or *silent* transitions, and $\mathbf{Act} = \{\mathtt{in}(a, M), \mathtt{out}(a, u), \nu u.\mathtt{out}(a, u)\}$ for any channel name n, term M and variable or name u, denote the set of labels of possible external or *visible* transitions. By construction $\mathbf{Act} \cap \mathbf{Int} = \emptyset$.

The *visible depth* is defined as the length of the longest complete trace of visible actions, i.e. labeled transitions, excluding internal reductions. Note that this may be infinite for processes including replication. We write $P \not\rightarrow$ if P cannot execute any transition, and $P \xrightarrow{\mu_1\mu_2\cdots\mu_n} P'$ for $P \xrightarrow{\mu_1} P_1 \xrightarrow{\mu_2} P_2 \xrightarrow{\mu_3} \ldots \xrightarrow{\mu_n} P'$.

Definition 3 (Visible Depth). *Let $length_v : (\mathbf{Act} \cup \mathbf{Int})^* \mapsto \mathbb{N}$ be a function where $length_v(\epsilon) = 0$ and $length_v(\mu w) = \begin{cases} 1 + length_v(w) & \text{if } \mu \in \mathbf{Act} \\ length_v(w) & \text{otherwise} \end{cases}$ Then the* visible depth *$|P|_v \in (\mathbb{N} \cup \{\infty\})$ of a closed process P is defined as follows:*

$$|P|_v = \sup\left\{ length_v(w) : P \xrightarrow{w} P' \not\rightarrow, w \in (\mathbf{Act} \cup \mathbf{Int})^* \right\}$$

The *total depth* is defined as the length of the longest complete trace of actions (including internal reductions).

Definition 4 (Total Depth). *Let $length_t : (\mathbf{Act} \cup \mathbf{Int})^* \mapsto \mathbb{N}$ be a function where $length_t(\epsilon) = 0$ and $length_t(\mu w) = 1 + length_t(w)$. The* total depth *$|P|_t \in (\mathbb{N} \cup \{\infty\})$ of a closed process P is defined as follows:*

$$|P|_t = \sup\left\{ length_t(w) : P \xrightarrow{w} P' \not\rightarrow, w \in (\mathbf{Act} \cup \mathbf{Int})^* \right\}$$

The norm of a process is defined as the length of the shortest complete trace, including internal reductions, where communications are counted as two. This is necessary to ensure that the norm of $P|Q$ is the sum of the norm of P and the norm of Q.

Definition 5 (Norm of a Process). *Let $length_n : (\mathbf{Act} \cup \mathbf{Int})^* \mapsto \mathbb{N}$ be a function where $length_n(\epsilon) = 0$ and $length_n(\mu w) = \begin{cases} 1 + length_n(w) & \text{if } \mu \neq \tau_c \\ 2 + length_n(w) & \text{if } \mu = \tau_c \end{cases}$ The norm $\mathcal{N}(P) \in (\mathbb{N} \cup \{\infty\})$ of a closed process P is defined as follows:*

$$\mathcal{N}(P) = \inf\left\{ length_n(w) : P \xrightarrow{w} P' \not\rightarrow, w \in (\mathbf{Act} \cup \mathbf{Int})^* \right\}$$

Example 3. We have $|P_{ex}|_v = 2$, $|P_{ex}|_t = 3$ and $\mathcal{N}(P_{ex}) = 4$.

The above definitions admit some simple properties.

Property 1. For any closed extended processes P, Q and R we have

- $P = Q|R$ implies $|P|_v = |Q|_v + |R|_v$
- $P = Q|R$ implies $|P|_t = |Q|_t + |R|_t$
- $P = Q|R$ implies $\mathcal{N}(P) = \mathcal{N}(Q) + \mathcal{N}(R)$
- $|P|_v \leq |P|_t$

Now we can define the two important subclass of processes: finite processes, i.e. processes with a finite longest complete trace, and normed processes, i.e. processes with a finite shortest complete trace.

Definition 6 (Finite and normed processes). *A closed process P is called* finite*, if $|P|_t$ is finite (which implies $|P|_v$ is finite). A closed process P is called* normed*, if $\mathcal{N}(P)$ is finite.*

It is easy to see that any finite process is normed, but not all normed processes are finite, as the following example illustrates.

Example 4. Consider $P = \nu a.(out(a, m)|(in(a, x).(!in(b, y)))|in(a, x))$. Then we have $P \to P' \sim_l 0$, hence P is normed. However we also have $P \to P'' \sim_l !in(b, y)$, which has infinite traces. Hence P is not finite.

It is also clear that not all processes are normed. Consider the following example.

Example 5. Consider $P = !(\nu x.out(c, x))$. It is easy to see that for no sequence of transitions s we have $P \xrightarrow{s} P' \not\to$, i.e. P has no finite traces.

3 Decomposition w.r.t. Strong Labeled Bisimilarity

We begin with the simpler case of strong labeled bisimilarity, defined as follows.

Definition 7 (Strong Labeled Bisimilarity ($\sim_l$)). *Strong labeled bisimilarity is the largest symmetric relation $\mathcal{R}$ on closed active processes, such that $A\ \mathcal{R}\ B$ implies:*

1. *$A \approx_s B$,*
2. *if $A \to A'$, then $B \to B'$ and $A'\ \mathcal{R}\ B'$ for some B',*
3. *if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq dom(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $B \xrightarrow{\alpha} B'$ and $A'\ \mathcal{R}\ B'$ for some B'.*

Note that $P \sim_l Q$ implies $|P|_t = |Q|_t$ and $\mathcal{N}(P) = \mathcal{N}(Q)$ for any closed processes P and Q. To ensure that labeled bisimilarity is a congruence w.r.t to parallel composition ("|") and closed under the application of contexts, we will require that active substitutions are only defined on variables of base type [7].

We define strong parallel primeness as follows: A process is prime if it cannot be decomposed into non-trivial subprocesses (w.r.t. strong labeled bisimilarity). We require the processes to be closed, which is necessary as our bisimulation relation is only defined on closed processes.

Definition 8 (Strongly Parallel Prime). *A closed process P is* strongly parallel prime*, if*

- $P \not\sim_l 0$ *and*
- *for any two closed processes Q and R such that $P \sim_l Q|R$, we have $Q \sim_l 0$ or $R \sim_l 0$.*

Example 6. Consider our running example:

$$P_{ex} = \nu k.\nu l.\nu m.\nu d.\,(\{^{l}/_{y}\}\,|\mathtt{out}(c, enc(n,k))|\mathtt{out}(d,m)|\mathtt{in}(d,x).\mathtt{out}(c,x))$$

We can decompose P_{ex} as follows:

$$P_{ex} \sim_l (\nu l.\,\{^{l}/_{y}\})|(\nu k.\mathtt{out}(c, enc(n,k)))|(\nu d.(\nu m.\mathtt{out}(d,m)|\mathtt{in}(d,x).\mathtt{out}(c,x)))$$

The first factor $S_1 = \nu l.\,\{^{l}/_{y}\}$ is prime since we cannot have two substitutions defining the same variable. It is easy to see that the second factor $S_2 = \nu k.\mathtt{out}(c, enc(n,k))$ is prime, as it can only perform one external transition. The third factor $S_3 = \nu d.(\nu m.\mathtt{out}(d,m)|\mathtt{in}(d,x).\mathtt{out}(c,x))$ is prime because its two parts can synchronize using a shared restricted channel and then perform a visible external transition. Since $dom(S_3) = \emptyset$ and $\mathcal{N}(S_3) = 2$, the only possible decomposition would be into two factors of norm 1 each, i.e. such that $S_3 \sim_l S_3'|S_3''$. This would however mean that both transitions of $S_3'|S_3''$ can be executed in any order, whereas in S_3 we have to start with the internal reduction. Hence no such decomposition exists.

With respect to applications in protocol analysis, this illustrates that shared restricted names, for example private channels or shared keys, can prohibit decomposition. This is unavoidable, since a decomposition should not change the behavior of the processes (up to $\sim_l$), yet it might appear to hinder the usefulness of the decomposition on first view. However, a decomposition that does not preserve the behavioral equivalence is probably not useful either, and note that – since our definition is solely based on the semantics and the bisimilarity notion – it allows to decompose as far as possible without changing the observed behavior, and thus any further decomposition will change the behavior. As a side-effect, the decomposition will show where shared restricted names (modeling for example keys) are actually used in a noticeable (w.r.t. to $\sim_l$) way, and where they can be ignored and processes can be further decomposed.

Note also that within a prime factor we can recursively apply the decomposition as our bisimilarity notion is closed under the application of contexts. For example if we have a prime factor $P = \nu a.P'$, we can bring P' into normal form, i.e. $P' \sim_l P_1'|...|P_n'$, and rewrite $P = \nu a.P'$ as $P \sim_l \nu a.(P_1'|...|P_n')$.

It is clear that not all processes can be written as a unique decomposition of parallel primes according to our definition.

Example 7. Consider $!P$ for a process $P \not\sim_l 0$. By definition we have $!P = P|!P$, hence $!P$ is not prime. At the same time any such decomposition contains again $!P$, a non-prime factor, which needs to be decomposed again. Thus there is no decomposition into prime factors.

However we can show that any closed normed process has a unique decomposition with respect to strong labeled bisimilarity. In a first step, we prove the existence of a decomposition.

Theorem 1 (Existence of Factorization). *Any closed normed process P can be expressed as the parallel product of strong parallel primes, i.e. $P \sim_l P_1 | \ldots | P_n$ where for all $1 \leq i \leq n$ P_i is strongly parallel prime.*

Proof. Sketch: The proof proceeds by induction on the norm of P, and inside each case by induction on the size of the domain. The second induction is necessary to deal with active substitutions, which cannot perform transitions. The main idea is simple: If a process is not prime, by definition it can be decomposed into two "smaller" processes, where we can apply the induction hypothesis. □

To show the uniqueness of the decomposition, we need some preliminary lemmas about transitions and the domain of processes. The first lemma captures the fact that intuitively any process which cannot perform any transition and has an empty domain, is bisimilar to 0 (the empty process).

Lemma 1. *For any closed process A with $dom(A) = \emptyset$ and $\mathcal{N}(A) = 0$, we have $A \sim_l 0$.*

We also need to show that if a normed process can execute a transition, it can also execute a norm-reducing transition.

Lemma 2. *Let A be a closed normed process with $A \xrightarrow{\mu} A'$ where μ is an internal reduction or visible transition. Then $A \xrightarrow{\mu'} A''$ with $\mathcal{N}(A'') < \mathcal{N}(A)$.*

These lemmas allow us to show the uniqueness of the decomposition.

Theorem 2 (Uniqueness of Factorization). *The strong parallel factorization of a closed normed process P is unique (up to $\sim_l$).*

Proof. Sketch: In the proof we have to deal with numerous cases due to the complex semantics of the calculus. Here we focus on the main differences compared to existing proofs for simpler calculi (e.g. [13]).

The proof proceeds by induction on the norm of P, and inside each case by induction on the size of the domain. By Lemma 1, each prime factor can either perform a transition, or has a non-empty domain. A transition may not always be norm-reducing since processes can be infinite, but in this case Lemma 2 gives us that if a normed process can execute a transition, it can also execute a norm-reducing one - which we can then consider. We suppose the existence of two different factorizations, and show that this leads to a contradiction. Consider the following four cases:

- If we have a process that cannot do any transition and has an empty domain, by Lemma 1 we have the unique factorization 0.
- If the process cannot perform a transition but has a non-empty domain, we can apply a restriction on part of the domain to hide all factors but one

(since we cannot have two substitutions defining the same variable). We can then use the fact that labeled bisimilarity is closed under the application of contexts to exploit the induction hypothesis, which eventually leads to a contradiction to the primality of the factors.
- In the case of a process with empty domain, but that can perform a transition, we can execute a transition and then apply the induction hypothesis. However, we have to be careful since in case of an internal reduction factors could fuse using scope extrusion (see Figure 1b). Hence, whenever possible, we choose a visible transition. If no such transition exists, processes cannot fuse using an internal reduction either, since this would mean they synchronized on a public channel, which implies the existence of visible transitions. Thus we can safely execute the invisible transition.
- In the last case (non-empty domain and visible transitions) we have to combine the above two techniques.

□

As a direct consequence, we have the following cancellation result.

Lemma 3 (Cancellation Lemma). *For any closed normed processes A, B and C, we have*

$$A|C \sim_l B|C \Rightarrow A \sim_l B$$

Proof. Sketch: All processes have a unique factorization and can be rewritten accordingly. As both sides are bisimilar, they have the same unique factorization, hence A and B must be composed of the same factors, thus they are bisimilar.

□

4 Decomposition w.r.t. Weak Labeled Bisimilarity

In this part, we discuss unique decomposition with respect to (weak) labeled bisimilarity. This is the standard bisimilarity notion in the Applied π-Calculus as defined by Abadi and Fournet in their original paper [1].

Definition 9 ((Weak) Labeled Bisimilarity ($\approx_l$) [1]). *(Weak) Labeled Bisimilarity is the largest symmetric relation $\mathcal{R}$ on closed active processes, such that $A\ \mathcal{R}\ B$ implies:*

1. $A \approx_s B$,
2. *if* $A \to A'$, *then* $B \to^* B'$ *and* $A'\ \mathcal{R}\ B'$ *for some* B',
3. *if* $A \xrightarrow{\alpha} A'$ *and* $fv(\alpha) \subseteq dom(A)$ *and* $bn(\alpha) \cap fn(B) = \emptyset$, *then* $B \to^* \xrightarrow{\alpha} \to^* B'$ *and* $A'\ \mathcal{R}\ B'$ *for some* B'.

The resulting bisimilarity notion is weak in the sense that only visible external transitions have to be matched exactly, and there may be a number of silent internal reductions in the background which are not taken into account. Note that $P \approx_l Q$ implies $|P|_v = |Q|_v$ for any closed processes P and Q.

Again we will assume that active substitutions can only be defined on variables of base type to ensure that labeled bisimilarity is a congruence w.r.t. to

parallel composition ("|") and closed under the application of contexts. Under this condition, it also coincides with observational equivalence [7]. This was claimed in the original paper [1] without requiring the additional condition, but turned out to be untrue when a counterexample was found (see [7] for more details).

To obtain our unique decomposition result for weak labeled bisimilarity, we need to define parallel prime with respect to weak labeled bisimilarity.

Definition 10 (Weakly Parallel Prime). *A closed extended process P is* weakly parallel prime*, if*

- $P \not\approx_l 0$ *and*
- *for any two closed processes Q and R such that $P \approx_l Q|R$, we have $Q \approx_l 0$ or $R \approx_l 0$.*

This definition is analogous to strongly parallel prime. However, as the following example shows, in contrast to strong bisimilarity, not all normed processes have a unique decomposition w.r.t. to weak bisimilarity.

Example 8. Consider $P = \nu a.(out(a, m)|(in(a, x).(!in(b, y)))|in(a, x))$. Then we have $P \approx_l P|P$, hence we have no unique decomposition. Note that this example does not contradict our previous result, as we have $P \not\sim_l P|P$, as $P \to P' \sim_l 0$, but $P|P \to P'' \sim_l P$ and $P|P \not\to P'''$ for any $P''' \sim_l 0$. Hence, w.r.t. strong labeled bisimilarity, P is prime.

If however we consider normed processes that contain neither restriction ("ν") nor conditionals, we have that any normed process is finite (and hence has a unique decomposition, as we show below).

Lemma 4. *For any process P that does not contain restriction ("ν") or conditionals ("if then else"), we have that P is finite if and only if P is normed.*

Similarly any process that does not contain replication is finite.

In the following we show that all finite processes have a unique decomposition w.r.t. to (weak) labeled bisimilarity. Again, in a first step, we show that a decomposition into prime factors exists.

Theorem 3 (Existence of Factorization). *Any closed finite active process P can be expressed as the parallel product of parallel primes, i.e. $P \approx_l P_1|\ldots|P_n$ where for all $1 \leq i \leq n$ P_i is weakly parallel prime.*

The proof is analogous to the proof of Theorem 1, but we have to proceed by induction on the visible depth instead of the norm, as two weakly bisimilar processes may have a different norm.

To prove uniqueness, we again need some preliminary lemmas about transitions and the domain of processes. This first lemma captures the fact that intuitively any process that cannot perform any visible transition and has an empty domain, is weakly bisimilar to 0 (the empty process).

Type of Process	Strong Bisimilarity ($\sim_l$)	Weak Bisimilarity ($\approx_l$)
finite	Theorem 1	Theorem 3
normed	Theorem 1	Counterexample 4
general	Counterexample 7	Counterexample 7

Table 1: Summary of unique factorization results for the Applied π-Calculus

Lemma 5. *If for a closed process A with $dom(A) = \emptyset$ there does not exist a sequence of transitions $A \rightarrow^* \xrightarrow{\alpha} A'$, then we have $A \approx_l 0$.*

Now we can show the uniqueness of the decomposition.

Theorem 4 (Uniqueness of Factorization). *The parallel factorization of a closed finite process P is unique (up to $\approx_l$).*
Proof. Sketch: In the proof we show the following statement: Any closed finite processes P and Q with $P \approx_l Q$ have the same factorization (up to $\approx_l$). The proof proceeds by induction on the sum of the total depth of both factorizations, and in each case on the size of the domain. We show that if we suppose the existence of two different factorizations, this leads to a contradiction.

The proof follows the same structure as the one for strong bisimilarity. In the case of processes with non-empty domain and no visible transition, we use the same idea and apply restrictions to use the induction hypothesis. In the other cases, when executing a transition to apply the induction hypothesis, we have to be more careful since each transition can be simulated using additionally several internal reductions. This can affect several factors, and prime factors could fuse using an internal reduction and scope extrusion (see Figure 1b). We can circumvent this problem by choosing transitions that decrease the visible depth by exactly one (such a transition must always exist). A synchronization of two factors in the other factorization would use at least two visible actions and the resulting processes cannot be bisimilar any more, since bisimilar processes have the same depth. Using Lemma 5 we know that each prime factor has either a non-empty domain or can execute a visible transition, which allows us to conclude. □

Again we have a cancellation result using the same proof as above.

Lemma 6 (Cancellation Lemma). *For any closed finite processes A, B and C, we have*

$$A|C \approx_l B|C \Rightarrow A \approx_l B$$

5 Related Work

Unique decomposition (or factorization) has been a field of interest in process algebra for a long time. The first results for a subset of CCS were published by

Moller and Milner [11, 13]. They showed that finite processes with interleaving can be uniquely decomposed with respect to strong bisimilarity. The same is true for finite processes with parallel composition, where – in contrast to interleaving – the parallel processes can synchronize. They also proved that finite processes with parallel composition can be uniquely decomposed w.r.t. weak bisimilarity. Compared to the Applied π-Calculus, BPP and CCS do not feature channel passing, scope extrusion and active substitutions.

Later on Christensen [2] proved a unique decomposition result for normed processes (i.e. processes with a finite shortest complete trace) in BPP with interleaving or parallel composition w.r.t. strong bisimilarity.

Luttik and van Oostrom [9] provided a generalization of the unique decomposition results for ordered monoids. They show that if the calculus satisfies certain properties, the unique decomposition result follows directly. Recently Luttik also extended this technique for weak bisimilarity [8]. Unfortunately this result cannot be employed in the Applied π-Calculus as active substitutions are minimal elements (with respect to the transition relation) different from 0.

6 Conclusion and Future Work

We presented two unique decomposition results for subsets of the Applied π-Calculus. We showed that any closed finite process can be decomposed uniquely with respect to weak labeled bisimilarity, and that any normed process can be decomposed uniquely with respect to strong labeled bisimilarity. Table 1 sums up our results.

As the concept of parallel prime decomposition has its inherent limitations with respect to replication ("!", see Example 7), a natural question is to find an extension to provide a normal form even in cases with infinite behavior. A first result in this direction has been obtained by Hirschkoff and Pous [6] for a subset of CCS with top-level replication. They define the *seed* of a process P as the process Q, Q bisimilar to P, of least size (in terms of prefixes) whose number of replicated components is maximal (among the processes of least size), and show that this representation is unique. They also provide a result for the Restriction-Free-π-Calculus (i.e. no "ν"). It remains however open if a similar result can be obtained for the full calculus.

Another interesting question is to find an efficient algorithm that converts a process into its unique decomposition. It is unclear if such an algorithm exists and can be efficient, as simply deciding if a process is finite can be non-trivial.

References

1. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '01, pages 104–115, New York, 2001. ACM.

2. Søren Christensen. *Decidability and Decompostion in Process Algebras.* PhD thesis, School of Computer Science, University of Edinburgh, 1993.
3. Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Defining privacy for weighted votes, single and multi-voter coercion. In *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS), Pisa, Italy, September 10-12, 2012*, volume 7459 of *LNCS*, pages 451–468. Springer, 2012.
4. Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. On parallel factorization of processes in the applied pi calculus. Technical Report TR-2012-3, Verimag Research Report, March 2012. Available at http://www-verimag.imag.fr/TR/TR-2012-3.pdf.
5. Jan Friso Groote and Faron Moller. Verification of parallel systems via decomposition. In *CONCUR '92: Proceedings of the Third International Conference on Concurrency Theory*, pages 62–76, London, UK, UK, 1992. Springer-Verlag.
6. Daniel Hirschkoff and Damien Pous. On bisimilarity and substitution in presence of replication. In *37th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6199 of *LNCS*, pages 454–465. Springer, 2010.
7. Jia Liu. A proof of coincidence of labeled bisimilarity and observational equivalence in applied pi calculus. Technical Report ISCAS-SKLCS-11-05, 2011. Available at http://lcs.ios.ac.cn/ jliu/.
8. Bas Luttik. Unique parallel decomposition in branching and weak bisimulation semantics. Technical report, 2012. Available at http://arxiv.org/abs/1205.2117v1.
9. Bas Luttik and Vincent van Oostrom. Decomposition orders – another generalisation of the fundamental theorem of arithmetic. *Theoretical Computer Science*, 335(2-3):147–186, 2005.
10. Robin Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 1989.
11. Robin Milner and Faron Moller. Unique decomposition of processes. *Theoretical Computer Science*, 107(2):357–363, 1993.
12. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes. *Information and Computation*, 100(1):1–40, 1992.
13. Faron Moller. *Axioms for Concurrency.* PhD thesis, School of Computer Science, University of Edinburgh, 1989.
14. Uwe Nestmann and Benjamin C. Pierce. Decoding choice encodings. *Information and Computation*, 163(1):1–59, 2000.
15. Catuscia Palamidessi and Oltea Mihaela Herescu. A randomized encoding of the pi-calculus with mixed choice. *Theoretical Computer Science*, 335(23):373 – 404, 2005.