

Modular Construction of Finite and Complete Prefixes of Petri net Unfoldings

Agnes Madalinski^{*†}

Faculty of Engineering Science, University Austral de Chile

General Lagos 2086, Valdivia, Chile

amadalin@uach.cl

Eric Fabre[†]

IRISA/INRIA Rennes, Campus de Beaulieu

35042 Rennes CEDEX, France

fabre@irisa.fr

Abstract. This paper considers distributed systems, defined as a collection of components interacting through interfaces. Components, interfaces and distributed systems are modeled as Petri nets. It is well known that the unfolding of such a distributed system factorises, in the sense that it can be expressed as the composition of unfoldings of its components. This factorised form of the unfolding generally provides a more compact representation of the system runs, because each component does not need to represent the possible choices (conflicts) appearing in the other components. Moreover, the unfolding factorisation makes it possible to analyse the system by parts.

The paper focuses on the derivation of a finite and complete prefix (FCP) in the unfolding of a distributed system. Specifically, one would like to directly obtain such a FCP in factorised form, without computing first a FCP of the global distributed system and then factorising it. The construction of such a “modular FCP” is based on deriving summaries of component behaviours w.r.t. their interfaces, that are then communicated to the neighbouring components. The latter combine

Address for correspondence: Facultad de Ciencias de la Ingeniería, Universidad Austral de Chile, Campus Miraflores, General Lagos 2086, Valdivia, Chile

^{*}This work was performed while Agnes Madalinski was a post-doc in the DistribCom team at INRIA, Rennes, France.

[†]The authors would like to thank the ACSD’08 program committee for selecting this work and proposing to submit a longer version to a special issue of *Fundamenta Informaticae*.

these summaries with their local behaviours, and prepare interface summaries for the next components. This globally takes the form of a message passing algorithm, where the global system is never considered.

Keywords: Petri net, unfolding, finite complete prefix, distributed system, modular computation, pullback, category theory

1. Introduction

Petri nets (PNs) are a widely used model for analysing concurrent and distributed systems. Among the dedicated tools, some recent approaches have been based on the so-called Petri nets unfoldings. The unfolding technique [3, 17] provides an efficient representation of all runs of a PN in the true concurrency (or partial order) semantics: Executions are considered as partially ordered sets of events rather than sequences, which results in important memory savings for algorithms analysing behaviours of this net. Most of these algorithms do not rely on the full unfolding of a PN but rather on a finite and complete prefix (FCP) of it [6, 14, 16]. The definition of “completeness” depends on the property of interest, but it is generally based on the fact that all reachable markings of the net are represented in this prefix, which is sufficient for several standard purposes in model-checking [4, 13, 15].

This paper focuses on particular PN called “distributed systems” defined by assembling components (sub-nets) by interfaces (shared sub-nets). A central property of these distributed systems is that their unfolding can be further “compressed” *via* a factorisation property [1, 7, 8, 19]. Specifically, the unfolding of a distributed system can be expressed as the composition of the unfoldings of its components. Interestingly, the collection of the local unfoldings may be more compact than the unfolding of the global system. We briefly illustrate this property on an example (Figure 1) for readers familiar with the notion of unfolding (definitions are recalled in Section 2). The system in Figure 1(a) consists of two components, \mathcal{A} and \mathcal{B} , interacting through an interface consisting of places $\{c_0, c_1\}$ and transitions $\{t_3, t_4\}$ (restricted to the places above). There exist $m = 2$ possibilities to produce t_4 in $Unf_{\mathcal{A}}$ (Figure 1(b)) and $n = 3$ possibilities to produce t_4 in $Unf_{\mathcal{B}}$ (Figure 1(c)), which results in $m \cdot n = 6$ different combinations in the unfolding of the global system.

The factorisation property of unfoldings was first mentioned by Winskel in [19], but only found recently its first applications in diagnosis problems [12]. A closely related problem consists in computing the minimal factorised form of an unfolding, and it was considered at the same period [1, 7, 8]. Specifically, there generally exist several factorised forms for the unfolding of a distributed system. The “largest” one has the full component unfoldings as factors. But of course not all runs of a given component will remain possible in the global system. Symmetrically, the “smallest” one is obtained by selecting runs of each component that do participate to a run of the global system. Equivalently, these minimal factors can be obtained by restricting (actually projecting) the global unfolding (on) to each component. Modular algorithms have been proposed in the above contributions to compute these minimal factors, *without computing first the unfolding of the global system*. This ability to analyse global properties of a system by local computations on small objects is of course an extremely desirable feature to address large modular systems.

The objective of this paper is to further explore this idea, and show that the same principles can be applied to compute factorised forms of *finite and complete prefixes* of unfoldings, with long term objective to derive tools for modular model checking. The problem amounts to building FCPs of components

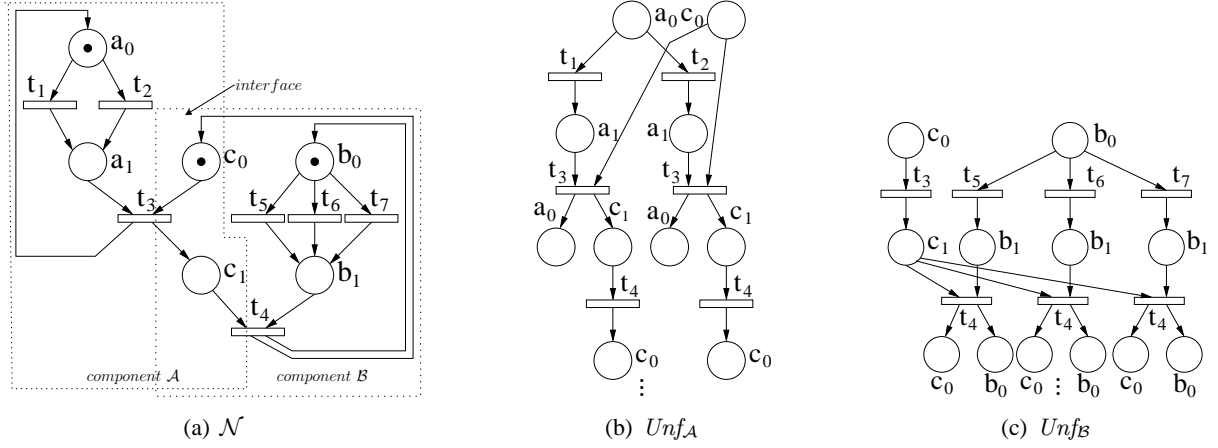


Figure 1. Distributed system and the unfoldings of its components.

in such a way that their composition is complete for the global system. With the constraint that these local prefixes must be computed in a modular manner, without reference to the global system. As a straightforward application, this would enable one to replace a component in a system and check that the global system still behaves well, without having to check the whole system again.

The literature about unfoldings has addressed problems that may look related to this question. For example, a technique to construct a finite complete prefix (FCP) of a synchronous product of labelled transition systems was presented in [5]. This technique uses the product structure of the system to simplify the construction of the FCP, but this prefix is not computed in factorised form. By contrast, [2] proposed a modular construction of complete prefixes, where compression gains are obtained by exploiting the symmetries of components. This contrasts with the present paper, where no symmetries are required. Several other technical points are also re-examined: the use of a more adequate notion of composition for unfoldings (which requires a different notion of morphism), and the absence of the non reentrance assumption, *i.e.* the fact that configurations of different components synchronise in a unique manner, when they do. This restriction prevents further compression gains when one works with unfoldings of components. Another important difference is that [2] uses global information for the derivation of local prefixes, which we want to avoid here. The objective is to use only local information, *i.e.* the component model plus information communicated by its neighbours, in order to determine the prefix to retain for each component. The information transmitted by a component to its neighbour will take the form of a “summary net” involving only behaviours of the interface that relates them.

The subject addressed here combines many technical aspects, some of which are not directly related to the problem and can be considered as “independent difficulties.” Therefore, to preserve the clarity of this first attempt at building modular FCP, the setting is voluntarily simplified at its maximum. The attention is restricted to unfoldings of safe PN, and the distributed system is reduced to two components related by an interface (Section 5 describes how this paradigm can then be extended). The interface is assumed to be a simple automaton rather than a true PN, in order to avoid some technical difficulties in the projection of branching processes¹, the same difficulties that led to the definition of augmented

¹Namely, the fact that some conflicts and causalities must be preserved by projections in the general case where the interface

branching processes in [8], and that motivated the introduction of interleaving structures in [1]. Finally, for simplicity again, the prefix definition is based on marking completeness, and on simple adequate orders.

The paper is organised as follows. Section 2 recalls the basic theoretical background concerning PNs, their unfoldings and the derivation of a canonical finite complete prefix. Section 3 defines distributed systems and recalls the associated factorisation property on their unfolding. Section 4 contains the main contribution: it presents the modular derivation of a finite and complete prefix in factorised form, focusing on the simple case of two components. Section 5 then explains how the paradigm of two components can be generalised to more complex networks of components.

2. Nets and unfoldings

This section recalls basic definitions concerning Petri nets and their unfoldings, essentially for stating the notations used in the paper. This material is adapted from [6, 14]. Not all notions are re-detailed, since the material is rather standard, and the reader is referred to the above references if he/she were bothered by some shortcuts.

2.1. Petri Nets

Nets. A net is a quadruple $\mathcal{N} = (P, T, \rightarrow, P^0)$ such that P and T are disjoint sets of *places* and *transitions*, respectively, $\rightarrow \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*. As usual, for a node $x \in P \cup T$, its *pre-set* $\bullet x$ is defined by $\bullet x = \{y \mid (y, x) \in \rightarrow\}$ and its *post-set* x^\bullet is defined by $x^\bullet = \{y \mid (x, y) \in \rightarrow\}$. $P^0 \subseteq P$ represents the initial marking of net \mathcal{N} , *i.e.* the set of places initially holding a token.

The usual sequential semantics of Petri nets are not recalled here, and we call *run* or *trajectory* a sequence of firable transitions in \mathcal{N} .

As mentioned in the introduction, this work limits its scope to *safe* nets, *i.e.* those for which every reachable marking $M : P \rightarrow \mathbb{N}$ satisfies $\forall p \in P, M(p) \in \{0, 1\}$. This allows one to identify a marking with the subset of places $M^{-1}(1)$. For technical reasons, and without loss of generality, we also limit ourselves to “producing” nets, *i.e.* such that $\forall t \in T, |t^\bullet| \geq 1$ (observe that $|t^\bullet| \geq 1$ is already necessary to safeness).

A labelled net $\mathcal{N} = (P, T, \rightarrow, P^0, \lambda, \Lambda)$ is a net extended with a finite label set Λ and a labelling function $\lambda : T \rightarrow \Lambda$ on transitions.

Morphisms. We rely on the definition of net morphism introduced by Winskel in [18], and later generalised in [20]. A morphism $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ between nets $\mathcal{N}_i = (P_i, T_i, \rightarrow_i, P_i^0)$, $i \in \{1, 2\}$, is a pair (ϕ^P, ϕ^T) with ϕ^P a relation on places and ϕ^T a partial function on transitions², such that:

- The initial marking is preserved by ϕ as follows: $P_2^0 = \phi(P_1^0)$ and $\forall p_2 \in P_2^0, \exists! p_1 \in P_1^0 : p_1 \phi p_2$.
- If $p_1 \phi p_2$ on places $p_i \in P_i$, then the restrictions $\phi : \bullet p_1 \rightarrow \bullet p_2$ and $\phi : p_1^\bullet \rightarrow p_2^\bullet$ are both total functions.

net can exhibit concurrency.

²For simplicity, ϕ^P and ϕ^T are replaced by ϕ when the context is unambiguous.

- ϕ preserves the environment of each transition: $t_2 = \phi(t_1)$ implies that restrictions $\phi^{op} : \bullet t_2 \rightarrow \bullet t_1$ and $\phi^{op} : t_2^\bullet \rightarrow t_1^\bullet$ are both total functions, where ϕ^{op} denotes the reversed relation on places.

Notice that net morphisms are designed to preserve runs : a sequence σ_1 of transitions that is fireable in \mathcal{N}_1 is mapped by ϕ into a sequence σ_2 of transitions that is also fireable in \mathcal{N}_2 . The fact that morphisms are relations on places, and not simply partial functions, allows for the duplication of places. This ability is necessary to make the composition of nets introduced below categorical products and pullbacks. In the sequel, ϕ will be said to be a partial/total function when both ϕ^T and ϕ^P are partial/total functions.

For labelled nets $\mathcal{N}_i = (P_i, T_i, \rightarrow_i, P_i^0, \lambda_i, \Lambda_i)$, the definition of a morphism $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ is reinforced by extra requirements:

- $\Lambda_2 \subseteq \Lambda_1$: the label set is reduced by ϕ ,
- $Dom(\phi_T) = \lambda_1^{-1}(\Lambda_2) \subseteq T_1$: ϕ is defined exactly on transitions having a shared label, and
- if $\phi_T(t_1) = t_2$ then $\lambda_1(t_1) = \lambda_2(t_2)$: label preservation.

This definition makes labelled nets and their morphisms a well defined category.

2.2. Branching processes

Occurrence nets. An *occurrence net* is a net $\mathcal{O} = (C, E, \rightarrow, C^0)$, where C is a set of *conditions* (places), E is the set of *events* (transitions) and $C^0 = \{c \in C : \bullet c = \emptyset\}$ is the set of initial conditions satisfying the following:

- defining the *causality relation* \prec as the transitive and irreflexive closure of the flow relation \rightarrow , one has $\forall x \in C \cup E, |\{y : y \prec x\}| < \infty$,
- every condition has at most one predecessor: $\forall c \in C, |\bullet c| \leq 1$
- no node is in self-conflict, where the conflict relation between nodes x and x' , denoted $x \# x'$, is defined by $\exists e, e' \in E, e \neq e', \bullet e \cap \bullet e' \neq \emptyset$ and $e \preceq x, e' \preceq x'$, denoting by \preceq the reflexive closure of \prec .

For technical reasons related to the construction of finite complete prefixes presented below, it is convenient to assume the existence of a distinguished event \perp in every occurrence net \mathcal{O} , which has an empty preset and has C^0 as post-set.

Configurations and cuts. A *configuration* of an occurrence net \mathcal{O} is a finite set of events $\kappa \subseteq E$ that is both conflict-free, $\forall e \neq e' \in \kappa, \neg(e \# e')$, and causally-closed, $\forall e \in \kappa, \forall e' \in E, e' \prec e \Rightarrow e' \in \kappa$. In addition it is required that the dummy initial event \perp be included in κ . A configuration corresponds to a possible run of \mathcal{O} in the partial order semantics, in the sense that any linear extension of κ forms a fireable sequence in \mathcal{O} (once the initial virtual event \perp is removed).

For every event $e \in E$, the configuration $[e] \stackrel{\text{df}}{=} \{e' \in E | e' \preceq e\}$ is called the *basic configuration*³ of e , and $\langle e \rangle \stackrel{\text{df}}{=} [e] \setminus \{e\}$ denotes the set of *causal predecessors* of event e . Given a configuration κ of \mathcal{O} , a

³The expression *local configuration* is used by several authors. However, here the term “local” is ambiguous, since it refers to a component, so it is replaced by “basic.”

set of events ϵ is called a *suffix* of κ iff $\kappa \cap \epsilon = \emptyset$ and $\kappa' = \kappa \cup \epsilon$ forms another configuration of \mathcal{O} , called an *extension* of κ . This is denoted by $\kappa' = \kappa \oplus \epsilon$.

Two nodes of \mathcal{O} are *concurrent*, denoted $x \parallel x'$, if neither $x \# x'$ nor $x \preceq x'$ nor $x' \preceq x$. A *co-set* $X \subseteq C$ is a set pairwise concurrent conditions, and a *cut* is a maximal co-set, for inclusion. One easily checks that cuts are in one to one correspondence with configurations: Given a cut X , the set $\kappa = \bullet X$ of all events preceding X forms a configuration. And conversely given a configuration κ , the set of maximal conditions for \prec in κ^\bullet , or equivalently⁴ $\kappa^\bullet \setminus \bullet \kappa$, defines a cut denoted by $Cut(\kappa)$. This relations reveals that cuts correspond to the reachable markings of \mathcal{O} .

Branching processes. A *branching process* (BP) of a net \mathcal{N} is a pair (\mathcal{O}, f) , where \mathcal{O} is an occurrence net and morphism $f : \mathcal{O} \rightarrow \mathcal{N}$ is a total function (i.e. both f^P and f^T are total functions). Such a morphism is called a *folding* of \mathcal{O} into \mathcal{N} . This folding can be seen as a labelling function on events and conditions of \mathcal{O} , by transitions and places of \mathcal{N} (it is assumed that f is not defined on \perp). It is further required that (\mathcal{O}, f) satisfy a parsimony condition: for all $e, e' \in E$, if $\bullet e = \bullet e'$ and $f(e) = f(e')$ then $e = e'$. In the sequel, we omit mentioning f when the folding is clear from the context, and simply talk about the branching process \mathcal{O} of \mathcal{N} .

This definition makes (\mathcal{O}, f) a representation of a set of runs of \mathcal{N} in a *true concurrency semantics*. Possible runs of \mathcal{O} correspond to its configurations, that are mapped by f into runs of \mathcal{N} . And conversely, a run of \mathcal{N} is the image through f of at most one configuration in \mathcal{O} , thanks to the parsimony condition. Furthermore, given a configuration κ of \mathcal{O} , the set of places $f(Cut(\kappa))$ is a reachable marking of \mathcal{N} , which is denoted by $Mark(\kappa)$, and conversely, a marking M of \mathcal{N} is *represented* in (\mathcal{O}, f) if there is a configuration κ of (\mathcal{O}, f) such that $M = Mark(\kappa)$. To obtain that any run of \mathcal{N} is represented exactly once, and consequently that any reachable marking of \mathcal{N} is the image of a marking in \mathcal{O} , one can consider the maximal branching process of \mathcal{N} , also called its unfolding $Unf_{\mathcal{N}}$, that is defined below.

A branching process (\mathcal{O}', f') of \mathcal{N} is a *prefix* of a another branching process (\mathcal{O}, f) , denoted by $(\mathcal{O}', f') \sqsubseteq (\mathcal{O}, f)$, if \mathcal{O}' is a causally closed sub-net of \mathcal{O} containing all its initial conditions, containing all postsets of events: $\forall e \in E, e \in \mathcal{O}' \Rightarrow e^\bullet \subseteq \mathcal{O}'$, and such that f' is the restriction of f to $C' \cup E'$. For any net \mathcal{N} there exists a unique (up to isomorphism) maximal (w.r.t \sqsubseteq) branching process denoted by $(Unf(\mathcal{N}), f_{\mathcal{N}})$, or for short $(Unf_{\mathcal{N}}, f_{\mathcal{N}})$, called the *unfolding* of \mathcal{N} [3, 15].

A few more notations are necessary for the sequel: The set of all finite (resp. basic) configurations of a branching process \mathcal{O} of \mathcal{N} is denoted by $\kappa_{fin}^{\mathcal{O}}$ (resp. $\kappa_{bas}^{\mathcal{O}}$), and the superscript \mathcal{O} is dropped when $\mathcal{O} = Unf_{\mathcal{N}}$.

2.3. Finite and complete prefixes

An abstract parametric model has been introduced in [13, 14] to define the notion of canonical finite and complete prefix (FCP) of an unfolding. Beside the introduction of an algorithm independent notion of FCP, this so-called *cutting context* also allows one to define different truncation criteria. It uses parameters which determine the information that the prefix should preserve (in the standard case, this is the set of all reachable markings) and specify the circumstances under which an event can be designated as a cut-off event.

⁴Taking into account that a configuration κ must contain the dummy event \perp , one has $\kappa^\bullet \setminus \bullet \kappa = (C^0 \cup \kappa^\bullet) \setminus \bullet \kappa$. The fact that transitions have a non-empty post-set also ensures that $\kappa^\bullet \setminus \bullet \kappa$ forms a maximal co-set.

Definition 2.1. A *cutting context* for $Unf_{\mathcal{N}}$ is a triple $\Theta = (\approx, \triangleleft, \{\kappa_e\}_{e \in E})$, where:

1. \approx is an equivalence relation on κ_{fin} .
2. \triangleleft , called an *adequate order*, is a strict well-founded partial order on κ_{fin} refining \subset , i.e. $\kappa \subset \kappa'$ implies $\kappa \triangleleft \kappa'$.
3. \approx and \triangleleft are *preserved by finite extensions*, i.e. for every pair of configurations $\kappa \approx \kappa'$, and for every suffix ϵ of κ , there exists a finite suffix ϵ' of κ' such that
 - (a) $\kappa' \oplus \epsilon' \approx \kappa \oplus \epsilon$, and
 - (b) if $\kappa' \triangleleft \kappa$ then $\kappa' \oplus \epsilon' \triangleleft \kappa \oplus \epsilon$.
4. $\{\kappa_e\}_{e \in E}$ is a family of subsets of κ_{fin} , i.e. $\forall e \in E, \kappa_e \subseteq \kappa_{fin}$.

The equivalence relation defines classes of configurations (typically those reaching the same marking of \mathcal{N}) of which the FCP should keep at least one representative. Intuitively, one should keep the minimal configurations for the adequate order \triangleleft , inside each class. This is actually done *via* the notion of *cut-off event* (see the definition below), that establishes truncation points in the unfolding, based on the last parameter κ_e . For example, the cutting context $\Theta_{ERV} = (\approx_{mar}, \triangleleft_{tot}, \{\kappa_e = \kappa_{bas}\}_{e \in E})$ corresponds to the framework in [6], where \approx_{mar} is the equivalence relation on reachable markings of \mathcal{N} , i.e. $\kappa' \approx_{mar} \kappa''$ iff $Mark(\kappa') = Mark(\kappa'')$, and \triangleleft_{tot} is a total adequate order (see [6]).

Definition 2.2. A branching process (\mathcal{O}, f) of a net \mathcal{N} is *complete w.r.t. a set* E_{cut} of events of $Unf_{\mathcal{N}}$ if the following holds:

1. If $\kappa \in \kappa_{fin}$, then there is $\kappa' \in \kappa_{fin}^{\mathcal{O}}$ such that $\kappa' \cap E_{cut} = \emptyset$ and $\kappa \approx \kappa'$.
2. If $\kappa \in \kappa_{fin}^{\mathcal{O}}$ is such that $\kappa \cap E_{cut} = \emptyset$, and e is an event such that $\kappa \oplus \{e\} \in \kappa_{fin}$ then $\kappa \oplus \{e\} \in \kappa_{fin}^{\mathcal{O}}$.

A branching process (\mathcal{O}, f) is *complete* if it is complete w.r.t. some set E_{cut} .

A canonical prefix is defined independently of an unfolding algorithm by means of *static cut-off events* with conjunction of the notion of *feasible events*. The latter are events whose causal predecessors are not static cut-off events, and thus, are included in the prefix determined by those cut-off events, which then appear as truncation points.

Definition 2.3. Given a cutting context Θ , the set of *feasible events*, denoted by $fsble^{\Theta}$, and the set of *static cut-off events*, denoted by cut^{Θ} , are two sets of events of $Unf_{\mathcal{N}}$ defined as follows.

1. An event e is a feasible event if $\langle e \rangle \cap cut^{\Theta} = \emptyset$.
2. An event e is a static cut-off event if it is feasible, and if there is a (so called *corresponding*) configuration $\kappa \in \kappa_e$ such that $\kappa \subseteq fsble^{\Theta} \setminus cut^{\Theta}$, $\kappa \approx [e]$, and $\kappa \triangleleft [e]$. When $\kappa = [e']$ for some (feasible and non cut-off) event e' , the latter is called a *corresponding event* of e .

The branching process $Pref_{\mathcal{N}}^{\Theta}$ induced by the set of events $fsble^{\Theta}$ is called the *canonical prefix* of $Unf_{\mathcal{N}}$. In the sequel, we will write $fsble_{\mathcal{N}}^{\Theta}$ and $cut_{\mathcal{N}}^{\Theta}$ when it is necessary to identify the net \mathcal{N} .

Remarks.

1. For a static cut-off event e , when $\kappa_e = \kappa_{bas}$, all corresponding configurations are actually given by corresponding *events* e' . If moreover the ordering \triangleleft is a total order, then e admits a unique corresponding event. The definition of extended canonical prefixes in Section 4 will assume these properties.
2. Notice that $Pref_{\mathcal{N}}^{\Theta}$ is uniquely determined by the cutting context Θ . Several fundamental properties of $Pref_{\mathcal{N}}^{\Theta}$ have been proved in [14]. In particular, $Pref_{\mathcal{N}}^{\Theta}$ is always complete w.r.t. cut^{Θ} . Moreover, it is finite if \approx has finitely many equivalence classes and if for every event e one takes $\kappa_{bas} \subseteq \kappa_e$.

3. Distributed systems

Distributed systems are modeled by a set of interconnected components interacting through shared subsystems, called interfaces. This modular structure is also called a “factorised form” of the system, due to the use of products (more precisely of pullbacks) to connect the components. This factorisation property is inherited by the unfolding of the distributed system: the unfolding of a product/pullback of components is the product/pullback of the unfoldings of components, which was the basis of modular processing in [12]. This section recalls this property, in order to prepare the derivation of finite complete prefixes in factorised form.

3.1. Compound system

Pullbacks. A distributed system is formally expressed as the combination of components by a pullback operation. In this paper, we limit ourselves to the case where all interactions between components are performed via their interface net, and where morphisms to this interface net are partial functions on both transitions and places (instead of relations on places). This results in a simpler definition of the pullback compared to [10].

Definition 3.1. Let \mathcal{A}, \mathcal{B} and \mathcal{C} be labelled nets, where \mathcal{A} and \mathcal{B} are related to their interface \mathcal{C} by morphisms $\phi_{\mathcal{A}} : \mathcal{A} \rightarrow \mathcal{C}$ and $\phi_{\mathcal{B}} : \mathcal{B} \rightarrow \mathcal{C}$ that are partial functions. Moreover, assume $\Lambda_{\mathcal{C}} = \Lambda_{\mathcal{A}} \cap \Lambda_{\mathcal{B}}$, *i.e.* that \mathcal{C} captures all the interactions between \mathcal{A} and \mathcal{B} . The *pullback* of this triple, or better the pullback of diagram $\mathcal{A} \xrightarrow{\phi_{\mathcal{A}}} \mathcal{C} \xleftarrow{\phi_{\mathcal{B}}} \mathcal{B}$, is denoted by $\mathcal{N} = \mathcal{A} \times_{\mathcal{N}}^{\mathcal{C}} \mathcal{B}$ and defined on places by

$$\begin{aligned}
P &= \{(p_{\mathcal{A}}, \star) : p_{\mathcal{A}} \in P_{\mathcal{A}}, p_{\mathcal{A}} \notin \text{Dom}(\phi_{\mathcal{A}})\} \\
&\cup \{(\star, p_{\mathcal{B}}) : p_{\mathcal{B}} \in P_{\mathcal{B}}, p_{\mathcal{B}} \notin \text{Dom}(\phi_{\mathcal{B}})\} \\
&\cup \{(p_{\mathcal{A}}, p_{\mathcal{B}}) \in P_{\mathcal{A}} \times P_{\mathcal{B}} : \phi_{\mathcal{A}}(p_{\mathcal{A}}) = \phi_{\mathcal{B}}(p_{\mathcal{B}})\}
\end{aligned} \tag{1}$$

and on transitions by

$$\begin{aligned}
T &= \{(t_{\mathcal{A}}, \star) : t_{\mathcal{A}} \in T_{\mathcal{A}}, t_{\mathcal{A}} \notin \text{Dom}(\phi_{\mathcal{A}}), \lambda_{\mathcal{A}}(t_{\mathcal{A}}) \in \Lambda_{\mathcal{A}} \setminus \Lambda_{\mathcal{B}}\} \\
&\cup \{(\star, t_{\mathcal{B}}) : t_{\mathcal{B}} \in T_{\mathcal{B}}, t_{\mathcal{B}} \notin \text{Dom}(\phi_{\mathcal{B}}), \lambda_{\mathcal{B}}(t_{\mathcal{B}}) \in \Lambda_{\mathcal{B}} \setminus \Lambda_{\mathcal{A}}\} \\
&\cup \{(t_{\mathcal{A}}, t_{\mathcal{B}}) \in T_{\mathcal{A}} \times T_{\mathcal{B}} : \phi_{\mathcal{A}}(t_{\mathcal{A}}) = \phi_{\mathcal{B}}(t_{\mathcal{B}})\}
\end{aligned} \tag{2}$$

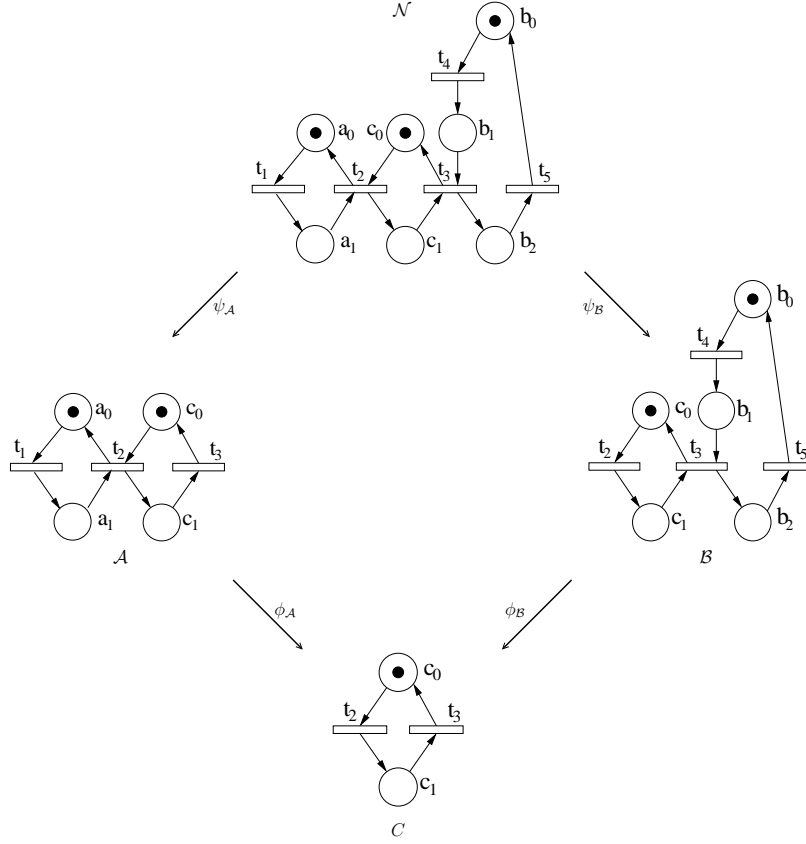


Figure 2. The pullback $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$.

The flow relation follows accordingly as well as the definition of initial places, and the label set is $\Lambda_{\mathcal{A}} \cup \Lambda_{\mathcal{B}}$.

Associated with this pullback construction, there exist canonical morphisms $\psi_{\mathcal{A}} : \mathcal{N} \rightarrow \mathcal{A}$ and $\psi_{\mathcal{B}} : \mathcal{N} \rightarrow \mathcal{B}$ that map elements of \mathcal{N} to the corresponding elements in \mathcal{A} and \mathcal{B} , respectively, and that satisfy $\phi_{\mathcal{A}} \circ \psi_{\mathcal{A}} = \phi_{\mathcal{B}} \circ \psi_{\mathcal{B}}$. Figure 2 depicts an example of $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ with associated morphisms. Place and transition names are preserved to facilitate the reading of morphisms.

Notice that the pullback is associative in the following sense, that will be used in Section 5. Consider nets $\mathcal{A}, \mathcal{B}, \mathcal{C}$ such that $\Lambda_{\mathcal{A}} \cap \Lambda_{\mathcal{C}} \subseteq \Lambda_{\mathcal{B}}$. Let net X be an interface between \mathcal{A} and \mathcal{B} , *i.e.* $\Lambda_{\mathcal{A}} \cap \Lambda_{\mathcal{B}} = \Lambda_X$, with morphisms $\phi_{\mathcal{A}, X} : \mathcal{A} \rightarrow X$ and $\phi_{\mathcal{B}, X} : \mathcal{B} \rightarrow X$. Similarly let net Y be an interface between \mathcal{B} and \mathcal{C} , *i.e.* $\Lambda_{\mathcal{B}} \cap \Lambda_{\mathcal{C}} = \Lambda_Y$, with morphisms $\phi_{\mathcal{B}, Y} : \mathcal{B} \rightarrow Y$ and $\phi_{\mathcal{C}, Y} : \mathcal{C} \rightarrow Y$. Then nets $\mathcal{N} = (\mathcal{A} \times_N^X \mathcal{B}) \times_N^Y \mathcal{C}$ and $\mathcal{N}' = \mathcal{A} \times_N^X (\mathcal{B} \times_N^Y \mathcal{C})$ are well defined and are isomorphic. They correspond to the categorical limit of diagram $\mathcal{A} \xrightarrow{\phi_{\mathcal{A}, X}} X \xleftarrow{\phi_{\mathcal{B}, X}} \mathcal{B} \xrightarrow{\phi_{\mathcal{B}, Y}} Y \xleftarrow{\phi_{\mathcal{C}, Y}} \mathcal{C}$, and it is well known that limits can be computed by parts⁵.

⁵Although we restrict the present paper to special cases of pullbacks, it can be shown that the category of labelled nets defined in this paper is complete, *i.e.* admits all finite limits [10].

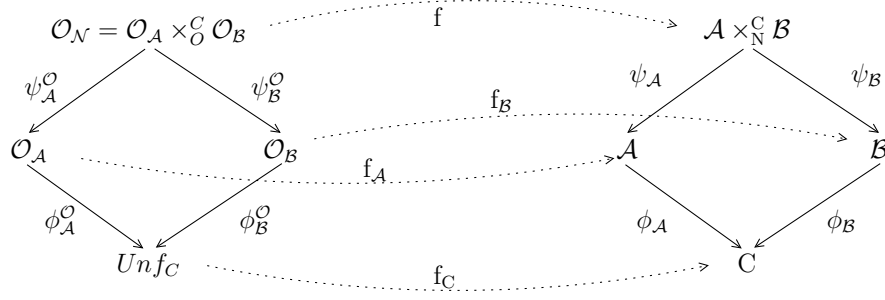


Figure 3. Commutative diagram of the pullbacks of branching processes and nets, and their relation by associated foldings.

Distributed systems. A net \mathcal{N} is said to be a *distributed system* if it can be obtained by recursive pullbacks of several components where the intermediary nets are interfaces (see paragraph above). The global *interaction structure* of a distributed system can be represented as a graph, where an edge is drawn between two components if they have a common interface, or better as a bipartite graph, alternating components and their related interfaces. For simplicity here, the notion of distributed system \mathcal{N} is first limited to two components \mathcal{A} and \mathcal{B} and an interface C . It is then generalised to tree-shaped systems in Section 5.

3.2. Factorisation of unfoldings

Pullbacks. As for ordinary nets, one can define the composition by pullback of two branching processes that have a common part. This operation actually derives from the pullback of nets. Let us first recall the so-called “universal property” of unfoldings: for every occurrence net \mathcal{O} and every morphism $\phi : \mathcal{O} \rightarrow \mathcal{N}$, there exists a unique morphism $\psi : \mathcal{O} \rightarrow Unf_{\mathcal{N}}$ such that $\phi = f_{\mathcal{N}} \circ \psi$. This makes the unfolding the smallest occurrence net representing all runs of \mathcal{N} .

The pullback of branching processes is then derived as follows (refer to Figure 3). Let (\mathcal{O}_A, f_A) and (\mathcal{O}_B, f_B) be branching processes of \mathcal{A} and \mathcal{B} , respectively. The morphism $\phi_A \circ f_A$ relates the occurrence net \mathcal{O}_A to net C , so by the universal property of Unf_C , there exists a unique morphism $\phi_A^{\mathcal{O}} : \mathcal{O}_A \rightarrow Unf_C$ that ensures $\phi_A \circ f_A = f_C \circ \phi_A^{\mathcal{O}}$, i.e. that makes the diagram commutative. By symmetry, one also has a morphism $\phi_B^{\mathcal{O}} : \mathcal{O}_B \rightarrow Unf_C$. Given these morphisms to Unf_C , the *pullback* of \mathcal{O}_A and \mathcal{O}_B in the category of occurrence nets is derived from the ordinary pullback of nets by

$$\mathcal{O}_A \times_O^C \mathcal{O}_B = Unf \left(\mathcal{O}_A \times_N^{Unf_C} \mathcal{O}_B \right). \quad (3)$$

Notice the subscript that distinguishes the two pullback operations, according to the category in which they are defined. It is easily checked that there exists a folding f from $\mathcal{O}_A \times_O^C \mathcal{O}_B$ that makes it a branching process of $\mathcal{A} \times_N^C \mathcal{B}$. Notice that (3) entails the existence of a recursive procedure to compute the pullback of branching processes, based on the recursive construction of unfoldings and on formulae (1,2).

It was shown in [19] that the factorised form of a distributed system yields a factorised form of its unfolding: Given $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ one has

$$Unf_{\mathcal{N}} = Unf_{\mathcal{A}} \times_O^C Unf_{\mathcal{B}}, \quad (4)$$

Projections. Consider the branching process (\mathcal{O}_A, f_A) of \mathcal{A} in Figure 3. Some nodes of \mathcal{O}_A are labelled by places and transitions of C , through morphism $f_C \circ \phi_A^\mathcal{O} = \phi_A \circ f_A : \mathcal{O}_A \rightarrow C$. The restriction of \mathcal{O}_A to these nodes is denoted by $(\mathcal{O}_A)|_C$. Equivalently, it corresponds to the restriction of \mathcal{O}_A to the domain of $\phi_A^\mathcal{O} : \mathcal{O}_A \rightarrow \text{Unf}_C$. By contrast, the projection $\Pi_C(\mathcal{O}_A)$ of \mathcal{O}_A on behaviours of C is defined as the image of \mathcal{O}_A in Unf_C by morphism ϕ_A .

$$\Pi_C(\mathcal{O}_A) = \phi_A^\mathcal{O}(\mathcal{O}_A). \quad (5)$$

Alternatively, this projection can be obtained by taking first $(\mathcal{O}_A)|_C$ and then performing a *trimming*, i.e. by merging isomorphic configurations in order to get a valid branching process of C .

Given (4), the notions of restriction and projection extend to any branching process of $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$. Consider Figure 3 where Unf_A and Unf_B replace \mathcal{O}_A and \mathcal{O}_B respectively. Then there exists a morphism $\phi_A^\mathcal{O} \circ \psi_A^\mathcal{O} = \phi_B^\mathcal{O} \circ \psi_B^\mathcal{O} : \text{Unf}_\mathcal{N} \rightarrow \text{Unf}_C$, whence the following definition of projection

$$\Pi_C(\text{Unf}_\mathcal{N}) = \phi_A^\mathcal{O} \circ \psi_A^\mathcal{O}(\text{Unf}_\mathcal{N}) = \phi_B^\mathcal{O} \circ \psi_B^\mathcal{O}(\text{Unf}_\mathcal{N}), \quad (6)$$

which extends to any branching process of \mathcal{N} (i.e. to any prefix of $\text{Unf}_\mathcal{N}$).

Notice that the restriction of Unf_A to a subset of nodes may erase causality or conflict relations between these nodes, which may then appear as concurrent in $(\text{Unf}_A)|_C$ whereas they were not in Unf_A . The same phenomenon occurs as well with projections. Therefore, a projection $\Pi_C(\text{Unf}_A)$ is said to be *non-misleading* iff every configuration κ' in $\Pi_C(\text{Unf}_A)$ is the image of a configuration of κ in Unf_A , and causality relations on events of κ' are not lost. Observe that projections on an interface C that is an automaton, i.e. a Petri net without concurrency, are by construction non-misleading.

The fact that the simple projections defined above may be misleading is the essential reason why this paper limits its scope to such simple interfaces as automata. For similar reasons, [1] introduced *interleaving structures*, while [8] proposed to work with *augmented branching processes*, in order to perform modular computations. Introducing here these technical refinements would certainly be possible, but would considerably load developments with technical details that are not really at the centre of the study. It is therefore chosen to focus on the specific difficulties related to obtaining finite complete prefixes with a modular approach, at the expense of a limited setting, where interface nets are not general nets but simple automata⁶.

Minimal pullback covering. Projections allow us to build the *minimal pullback covering* of $\text{Unf}_\mathcal{N}$, which restricts the factors Unf_A and Unf_B to the behaviours of components \mathcal{A}, \mathcal{B} that remain feasible in the entire system \mathcal{N} . From $\text{Unf}_\mathcal{N} = \text{Unf}_A \times_O^C \text{Unf}_B$ one can write

$$\text{Unf}_\mathcal{N} = \Pi_A(\text{Unf}_\mathcal{N}) \times_O^C \Pi_B(\text{Unf}_\mathcal{N}), \quad (7)$$

where $\Pi_A(\text{Unf}_\mathcal{N}) = \psi_A^\mathcal{O}(\text{Unf}_\mathcal{N})$ is a prefix of Unf_A , and similarly for \mathcal{B} . The “factors” $\Pi_A(\text{Unf}_\mathcal{N})$ and $\Pi_B(\text{Unf}_\mathcal{N})$ are called *minimal factors* in the sense that taking a strict prefix in any or both of them would prevent recovering the full unfolding $\text{Unf}_\mathcal{N}$ by pullback. Whence the expression “minimal pullback covering.”

⁶Notice that one can always come back to this situation when an interface C contains concurrency. The idea is to replace C by its sequentialised version, i.e. its marking graph \bar{C} , to replace as well \mathcal{A} by the product net $\bar{\mathcal{A}} = \mathcal{A} \times \bar{C}$, \mathcal{B} by the product $\bar{\mathcal{B}} = \mathcal{B} \times \bar{C}$, and then to consider the pullback of components $\bar{\mathcal{A}}$ and $\bar{\mathcal{B}}$ with interface \bar{C} .

Minimal factors can be computed in a modular manner without computing $Unf_{\mathcal{N}}$ itself. As soon as projections on C are non-misleading, one has

$$\Pi_{\mathcal{A}}(Unf_{\mathcal{N}}) = Unf_{\mathcal{A}} \times_O^C \Pi_C(Unf_{\mathcal{B}}) \quad (8)$$

(and symmetrically for $\Pi_{\mathcal{B}}(Unf_{\mathcal{N}})$) [7, 8]. This relation expresses that knowing the behaviours of \mathcal{B} on the interface net C is sufficient to \mathcal{A} to determine which of its runs remain possible in the global system \mathcal{N} . Equation (8) only holds in the case of non-misleading projections. If $\Pi_C(Unf_{\mathcal{B}})$ is misleading, it may contain “fake” concurrency, *i.e.* a configuration that is not the projection of a configuration of $Unf_{\mathcal{B}}$. This could be validated by pullback another configuration in $Unf_{\mathcal{A}}$ that would not correspond to any run of \mathcal{N} . It is precisely to avoid these fake concurrent events that *augmented branching processes* were developed in [8], in order to preserve all necessary causalities and conflicts, while [1] later proposed the alternate technique of *interleaving structures*. Assuming that all interfaces are automata, *i.e.* Petri nets without concurrency, all these difficulties are left aside from the main topic of the paper.

For more complex distributed systems, taking the shape of a network of components and interfaces, minimal factors can be obtained by extending the above principle of modular computations. It then takes the form of a message passing algorithm, which runs on the interaction structure of \mathcal{N} and progressively updates information on interfaces. The algorithm can be proved exact for systems living on a tree, and is only approximate in other cases. We refer the reader to chapter 2 of [11] for details.

4. Modular complete prefixes

The objective of this paper is to compute finite and complete prefixes (FCP) of a distributed system \mathcal{N} in factorised form. To focus on the essential difficulties of this problem, the discussion is first limited to an elementary distributed system $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$. The next section will generalise the discussion to more complex (tree-shaped) systems. As mentioned above, it is furthermore assumed that the interface C is an automaton and not a general safe net, in order to avoid the extra technical difficulty of dealing with misleading projections.

A trivial solution to obtain a FCP of \mathcal{N} in factorised form would be to start from say a global FCP, $Pref_{\mathcal{N}}$, and to project it on the components \mathcal{A}, \mathcal{B} of \mathcal{N} to obtain its minimal pullback covering:

$$Pref_{\mathcal{N}} \sqsubseteq \Pi_{\mathcal{A}}(Pref_{\mathcal{N}}) \times_O^C \Pi_{\mathcal{B}}(Pref_{\mathcal{N}}). \quad (9)$$

Notice that in (9) one has equality if $Pref_{\mathcal{N}}$ is already given in pullback form, as in (7). But in general the minimal pullback covering of $Pref_{\mathcal{N}}$ is larger. This is due to the fact that the configurations in the projections can be recombined in more numerous manners⁷ than were present in $Pref_{\mathcal{N}}$. This approach would of course provide us with the desired result, but it imposes to work first on the global system, which could be extremely expensive if the system is large. In particular, this prevents taking advantage of the compression gain provided by factorised forms, and this prevents as well processing the system by parts. The objective of this section is thus to build *directly* the factors of a FCP of $Unf_{\mathcal{N}}$.

To obtain them, it is *not* sufficient to simply build FCP of the components; in general, their combination by pullback would not be complete for the global system. Thus, the idea is to build FCP of

⁷Intuitively, this reproduces the fact that an object in a Euclidean space is included in the product of its orthogonal projections, and equal to this product if the considered object was already given in product form. For instance, observing that the product of the orthogonal projections of a circle gives a square including the circle.

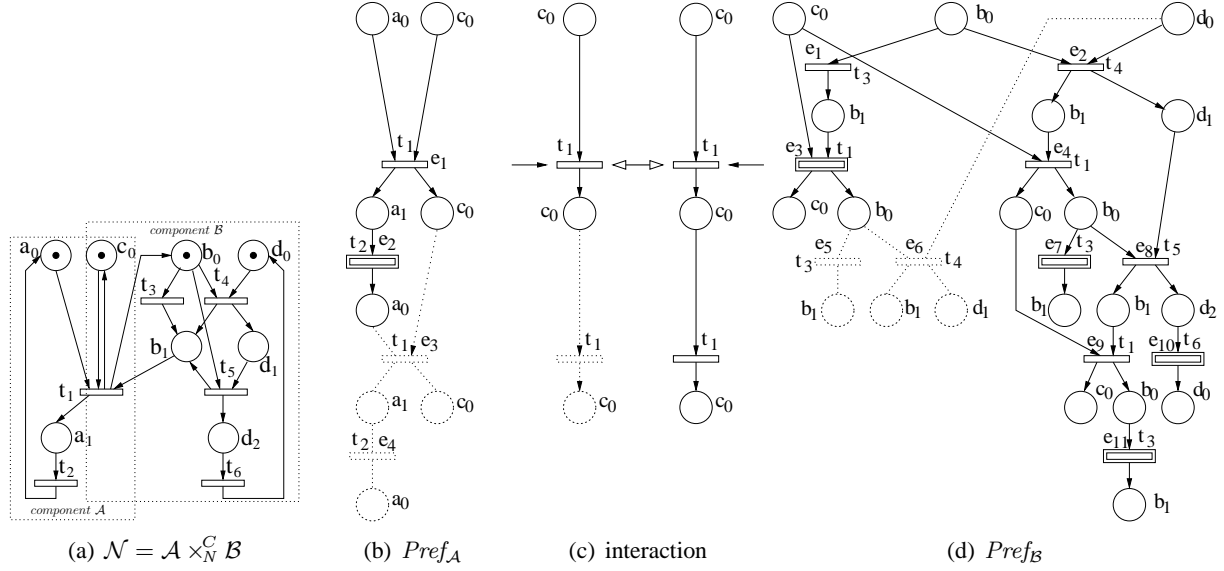


Figure 4. Illustrating the necessity of an agreement on the required behaviours of the interface, in order to obtain global completeness when the two FCP are assembled by pullback. Cut-off events in the local FCP are represented as double boxes.

components that will also provide sufficiently rich behaviours on the interface C in order to ensure global completeness. This demands of course an agreement of the two components about what behaviours the interface should provide, and so an exchange of information between components is necessary.

A first approach to implement this idea is to start with locally complete prefixes on each component, then transmit the behaviours required on the interface to the other component, and receive as well its own requirements, before recomputing local FCP with these extra requirements, and so on until convergence. However, this iterative approach might result in many iterations between the components, and will not easily extend to complex architectures of distributed systems.

Consider the example in Figure 4, which depicts $Pref_{\mathcal{A}}$ and $Pref_{\mathcal{B}}$, two local FCP, with a simple interface only consisting of the transition t_1 and the place c_0 . Events and conditions are directly labelled by transitions and place names of the corresponding Petri net and cut-off events are represented by double boxes. $Pref_{\mathcal{A}}$ needs one occurrence of the interface transition t_1 . By contrast, $Pref_{\mathcal{B}}$ needs two occurrences of t_1 to be complete. Thus, $Pref_{\mathcal{A}}$ has to be extended by one occurrence of t_1 ; this new behaviour on the interface has to be propagated to $Pref_{\mathcal{B}}$. In addition, one needs to check for global completeness by considering global markings, and this might require the extension of the prefixes of these components until global truncation points are reached. In the end, this might result in many exchanges.

To solve these difficulties, we propose below to capture the behaviours of the interface required by a component under the form of a *summary net*, transmitted in “one shot.” The summary net is obtained from an extended canonical prefix by “refolding” its restriction to the interface.

4.1. Extended canonical prefixes

The extended canonical prefix of a component \mathcal{A} (or \mathcal{B}) is built with regard to its interface C . Such a prefix captures the behaviour of that interface in relation to its component. It is obtained by restricting the cutting context, in particular the set of configurations which are used for the cut-off criterion.

Definition 4.1. Let \mathcal{A} be a component with an interface C . Then, w.r.t. the interface C , the cutting context $\Theta_C = (\approx_{mar}, \triangleleft_{tot}, \{\kappa_e\}_{e \in E_{\mathcal{A}}})$ is defined by

$$\forall e \in E_{\mathcal{A}}, \quad \kappa_e = \begin{cases} \{[e'] : e' \in E_{\mathcal{A}}, \Pi_C(e') \neq \emptyset\} & \text{if } \Pi_C(e) \neq \emptyset & \text{(a)} \\ \{[e'] : e' \in E_{\mathcal{A}}, \Pi_C([e] \triangle [e']) = \emptyset\} & \text{otherwise} & \text{(b)} \end{cases} \quad (10)$$

where \triangle is the symmetric set difference.

The restriction of the cutting context Θ_C in \mathcal{A} means that:

- (a) An interface event (an event corresponding to a transition of the interface) can be designated as a cut-off event only if its corresponding event is also an interface event.
- (b) Whereas, the corresponding event e' of a private cut-off event e (i.e. an event which does not correspond to any transition of the interface) has to be chosen such that there are no interface events in $\Pi_C([e] \triangle [e'])$. This ensures that there is no interface event between e' and e when $e' \in [e]$. This condition is necessary to cut infinite chains of private events in the unfolding of a component, and obtain a finite prefix.

Definition 4.2. The branching process $Pref_{\mathcal{A}}^{\Theta_C}$ induced by the set of events $fsble_{\mathcal{A}}^{\Theta_C}$ is called the *extended canonical prefix* of \mathcal{A} w.r.t. its interface C . The elements of $cut_{\mathcal{A}}^{\Theta_C}$ are called *extended cut-off events*.

Figures 5(a) and (c) show the extended canonical prefixes of the components \mathcal{A} and \mathcal{B} in Figure 2. In the sequel the cut-off events are drawn as double boxes in figures, whereas in the case of extended cut-off events the outer box is drawn with a dashed line. $Pref_{\mathcal{A}}^{\Theta_C}$ coincides with the canonical prefix since the only cut-off event and its corresponding event (\perp) are interface events⁸. However, this is not the case for $Pref_{\mathcal{B}}^{\Theta_C}$ in Figure 5(c). The extended prefix is larger than the standard prefix, that would be obtained by setting e_4 as a cut-off event since $[\perp] \approx_{mar} [e_4]$ and $[\perp] \triangleleft_{tot} [e_4]$. However, e_4 does not correspond to an extended cut-off event since it is a private event and $\Pi_C([e_4] \triangle [\perp]) = \{e_0, e_2\}$. This applies also to the event e_5 . The event e_6 is an extended cut-off since itself and its corresponding event e_2 are interface events.

An extended canonical prefix is complete since it is a canonical prefix (see Proposition 2.9 in [13]). It has to be shown that it is finite.

Proposition 4.1. $Pref_{\mathcal{A}}^{\Theta_C}$ is finite.

⁸This is where the presence of a dummy initial event \perp in any branching process becomes useful, since it can be designated as corresponding event for some cut-off e , and thus reduces the size of complete prefixes.

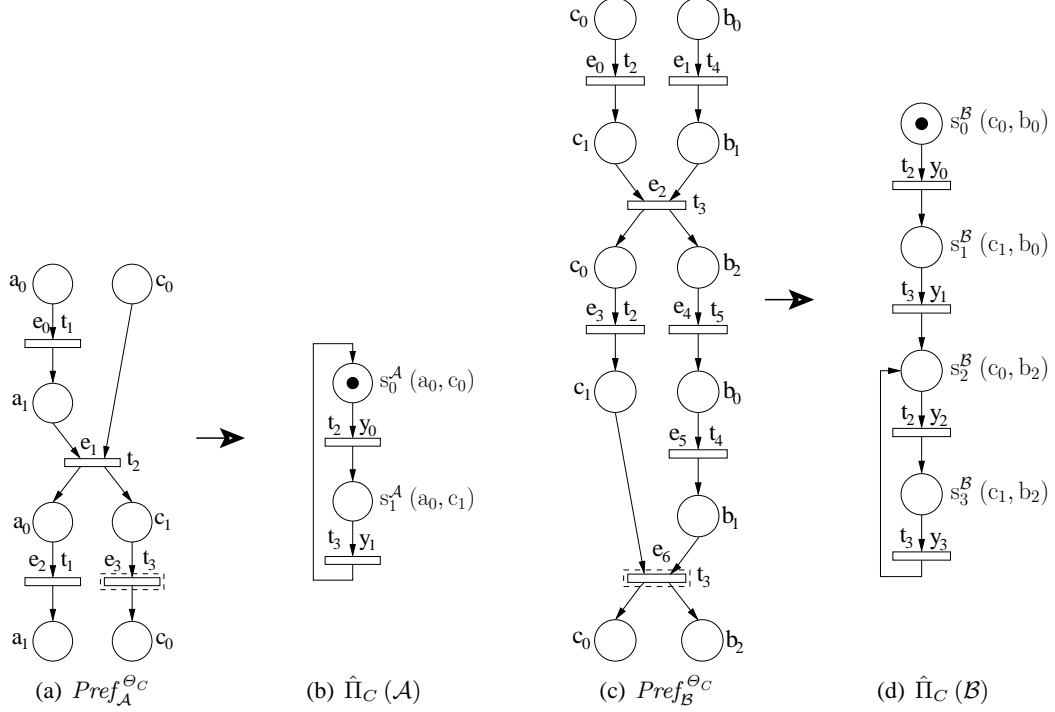


Figure 5. Extended prefixes of nets \mathcal{A}, \mathcal{B} given in Fig. 2 and their corresponding summary nets. The dummy initial events \perp are not represented.

Proof:

By Proposition 2.10 in [13] it is enough to show that each infinite \prec -chain in $Unf_{\mathcal{A}}$ can be cut. Two cases are considered.

If there are infinitely many interface events in the chain, *i.e.* events which correspond to C , then this chain necessarily contains an extended cut-off. There is indeed a finite number of markings in component \mathcal{A} , so considering only those reached by interface events, a repetition must occur, which defines an extended cut-off (case (a) in the definition).

Otherwise, there is only a finite number of interface events in the chain. Since the chain is infinite it contains an infinite tail formed of private (non-interface) events only. By the same arguments as above, there is necessarily a repetition in the markings reached by these private events, which proves the existence of an extended cut-off event in the chain (case (b) in the definition). \square

4.2. Summary nets

The summary net of a component captures the behaviour of a component w.r.t. its interface, and it is derived from its extended canonical prefix.

Definition 4.3. Let $Pref_{\mathcal{A}}^{\theta_C}$ be the extended canonical prefix of \mathcal{A} w.r.t. its interface C . The restriction $(Pref_{\mathcal{A}}^{\theta_C})|_C$ to events and conditions of interface C is a (bipartite) rooted oriented tree that can be

considered as a finite automaton (Q, Y, q_0, \rightarrow) with states Q given by the conditions, $q_0 = \perp^\bullet$, and transitions Y corresponding to the events without \perp .

Let relation R on states be defined by qRq' iff there exists events e, e' in $(Pref_{\mathcal{A}}^{\Theta_C})|_C$ such that e is an extended cut-off, e' is its (unique) corresponding event, and $q = e^\bullet, q' = e'^\bullet$. Relation R generates the equivalence relation \equiv on Q .

The *summary net* of \mathcal{A} w.r.t. its interface C is the automaton $\hat{\Pi}_C(\mathcal{A}) = (S, Y, s_0, \rightarrow)$ obtained as the quotient of (Q, Y, q_0, \rightarrow) by relation \equiv .

More informally, the summary net of \mathcal{A} w.r.t. its interface C is the automaton $\hat{\Pi}_C(\mathcal{A}) = loop((Pref_{\mathcal{A}}^{\Theta_C})|_C)$ where the *loop* operation is defined by

$$\forall e \in (cut_{\mathcal{A}}^{\Theta_C})|_C, \text{ merge } e^\bullet \text{ and } e'^\bullet \text{ where } e' \text{ is the (unique) corresponding event of } e.$$

The initial state s_0 corresponds to the minimal condition of $(Pref_{\mathcal{A}}^{\Theta_C})|_C$.

Remarks.

1. The *loop* operation folds the extended prefix restricted to the interface, i.e. the states reached by interface cut-off events e and their corresponding events e' are merged, which allows one to repeat the “sequence” from e' to e an arbitrary number of times. Recall that the cutting context Θ_C is designed in such a way that interface cut-off events have a (unique) corresponding event that is also an interface event. Notice as well the importance of requiring interfaces to be automata, in order to have a summary nets that are also automata.
2. The nodes of $\hat{\Pi}_C(\mathcal{A})$ correspond to the nodes of $(Pref_{\mathcal{A}}^{\Theta_C})|_C$ (up to the merge operation defining states S). They preserve their labelling by states and transitions of interface C . More precisely, there exist canonical morphisms $\bar{f}_{\mathcal{A}} : Pref_{\mathcal{A}}^{\Theta_C} \rightarrow \hat{\Pi}_C(\mathcal{A})$ and $\bar{\phi}_{\mathcal{A}} : \hat{\Pi}_C(\mathcal{A}) \rightarrow C$ such that $\bar{\phi}_{\mathcal{A}} \circ \bar{f}_{\mathcal{A}} = \phi_{\mathcal{A}} \circ f_{\mathcal{A}}$ (see Fig. 6). The morphism $\bar{\phi}_{\mathcal{A}}$ is a total function; it uniquely defines transition labels and will be used below to compose $\hat{\Pi}_C(\mathcal{A})$ by pullback.
3. In addition, to facilitate discussions below, each transition of $\hat{\Pi}_C(\mathcal{A})$ corresponds to an event e of $(Pref_{\mathcal{A}}^{\Theta_C})|_C$, and so it is associated with the marking $Mark([e])$ of component \mathcal{A} that it produces in $Pref_{\mathcal{A}}^{\Theta_C}$. In figures, this marking is represented as a label on the state $s = e^\bullet$. Observe that no two states of the summary net are associated with the same marking of \mathcal{A} : This is due to the fact that a *total* adequate order is used to define the extended prefix. For two events e, e' in $(Pref_{\mathcal{A}}^{\Theta_C})|_C$, if one had $Mark([e]) = Mark([e'])$, then one of these events would necessarily be a cut-off, the other would be its corresponding event, and e^\bullet would be merged with e'^\bullet by the *loop* operation.

Figure 5 depicts the summary nets of the components in Figure 2 together with their corresponding extended prefixes. The net $\hat{\Pi}_C(\mathcal{A})$ in Figure 5(b) coincides with the interface C . This is not the case with $\hat{\Pi}_C(\mathcal{B})$ in Figure 5(d). There are two states labelled by either c_0 or c_1 , however, they are associated with different markings in \mathcal{B} (given in brackets next to the states). Observe that summary nets carry some minimal information about the components they characterise, since their states are linked with the markings reached by interface events in the unfolding of the component.

The proposition below expresses that the summary net describes in a compact structure all possible behaviours of \mathcal{A} on the interface C .

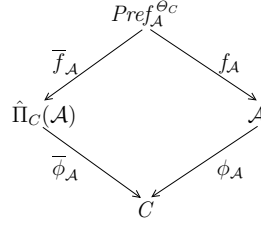


Figure 6. Canonical morphism relating the summary net $\hat{\Pi}_C(\mathcal{A})$ to the interface C .

Proposition 4.2. $\Pi_C(Unf_{\mathcal{A}}) = \Pi_C(Unf_{\hat{\Pi}_C(\mathcal{A})})$.

Proof:

Consider the two occurrence nets $(Unf_{\mathcal{A}})|_C$ and $Unf_{\hat{\Pi}_C(\mathcal{A})}$, that are labelled by places and transitions of the interface C (see Fig. 6). If one can show that they represent the same runs of C , it immediately follows that they have the same image in Unf_C , which proves the proposition. So we proceed by double inclusion.

Observe first that both $(Unf_{\mathcal{A}})|_C$ and $Unf_{\hat{\Pi}_C(\mathcal{A})}$ have no concurrent events, since the interface C is an automaton. Therefore these occurrence nets are simply trees.

The “inclusion” of $Unf_{\hat{\Pi}_C(\mathcal{A})}$ into $(Unf_{\mathcal{A}})|_C$, in terms of runs of C , is easy to obtain. This is actually the important part of the result for the sequel: it means that $\hat{\Pi}_C(\mathcal{A})$ is sound, *i.e.* that it does not show behaviours of the interface C that \mathcal{A} could not accomplish. Consider a configuration of $Unf_{\hat{\Pi}_C(\mathcal{A})}$, or equivalently a sequence $\sigma \in Y^*$ of transitions of the summary net, that is a sequence of events of the extended canonical prefix $Pref_{\mathcal{A}}^{\theta_C}$. This sequence can be split into subsequences as $\sigma = \sigma_1\sigma_2 \dots \sigma_n$ where each σ_i (excepted perhaps the last one σ_n) terminates by an element $e_i \in Y$ that is an extended cut-off in $Pref_{\mathcal{A}}^{\theta_C}$, and where no other element of a σ_i is such an extended cut-off. In this parsing of σ , the segment σ_1 correspond to a unique path in the extended canonical prefix $Pref_{\mathcal{A}}^{\theta_C}$, immediately following \perp , and its last event e_1 is associated with the marking $Mark([e_1])$ of \mathcal{A} (see remark 3 after Definition 4.3). By construction of the summary net, the segment σ_2 appears as a suffix of the corresponding event e'_1 of e_1 in $Pref_{\mathcal{A}}^{\theta_C}$. By definition of unfoldings, there exists a suffix of e_1 in $Unf_{\mathcal{A}}$ that is isomorphic to σ_2 , which proves that the sequence of transitions of C that $\sigma_1\sigma_2$ represents is present in $(Unf_{\mathcal{A}})|_C$. Applying the same argument recursively proves that the full sequence σ represents a sequence of transitions of C that \mathcal{A} can actually perform in some configuration.

The converse “inclusion” of $(Unf_{\mathcal{A}})|_C$ into $Unf_{\hat{\Pi}_C(\mathcal{A})}$, in terms of runs of C , is less obvious to derive. It proves that $\hat{\Pi}_C(\mathcal{A})$ is complete, *i.e.* that it describes *all* runs of interface C that component \mathcal{A} is able to perform. The proof is based on the completeness of the extended canonical prefix, and follows the same path as above, in the reverse direction. Let σ be a configuration in $(Unf_{\mathcal{A}})|_C$, and let σ_1 be the prefix of σ that stops at the first extended cut-off e_1 . This initial segment σ_1 is a configuration of the extended canonical prefix $Pref_{\mathcal{A}}^{\theta_C}$. Consider e'_1 the corresponding event of the cut-off e_1 . e'_1 is an event of $Pref_{\mathcal{A}}^{\theta_C}$. By definition of completeness (see point 2 in Definition 2.2), there exists a non-empty segment σ_2 in $\sigma = \sigma_1\sigma_2\sigma'$ that is isomorphic to a suffix σ'_2 of e'_1 in $Pref_{\mathcal{A}}^{\theta_C}$. Definition 2.2 expresses it for the first event after σ_1 in σ , but this naturally extends by recursion. Let us take for σ_2 the segment such that σ'_2 finishes at another extended cut-off e_2 of $Pref_{\mathcal{A}}^{\theta_C}$. By repeating this construction, one can

actually decompose σ into a sequence of segments σ_i that all appear in the extended canonical prefix $Pref_{\mathcal{A}}^{\Theta_C}$, related by jumps between cut-offs and their corresponding events. Consequently, the run σ will have an isomorphic counterpart as a run of the summary net. \square

Notice that the arguments that are used in this proof reproduce known properties of finite complete prefixes of unfoldings, namely the fact that an FCP $Pref_{\mathcal{N}}^{\Theta}$ allows to recover the full unfolding $Unf_{\mathcal{N}}$. As above, the idea is that after any cut-off event e in the FCP, with corresponding event e' , one can “glue” a copy of $(Pref_{\mathcal{N}}^{\Theta})|_{e'+}$, the restriction of $Pref_{\mathcal{N}}^{\Theta}$ to nodes that are in the future of e' , denoted by $e'+$. This introduces new “copies” of some cut-off events in the structure, that become new connection points. Doing that repeatedly and for all cut-off events allows to recover the full unfolding. This is a consequence of the completeness property.

4.3. Modular construction of prefix factors

Definition 4.4. Given a distributed system $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ the *modular prefix factor* (MPF) of a component \mathcal{A} is obtained by

$$\bar{\mathcal{A}} = \mathcal{A} \times_N^C \hat{\Pi}_C(\mathcal{B}) \quad (11)$$

$$\overline{Pref}_{\mathcal{A}}^{\Theta_*} = \Pi_{\mathcal{A}}(Pref_{\mathcal{A}}^{\Theta_*}) \quad (12)$$

where $\Theta_* = \{\approx_{mar}, \subset, \{\kappa_e = \kappa_{bas}\}_{e \in E}\}$. The symmetric relation is used to define $\overline{Pref}_{\mathcal{B}}^{\Theta_*}$.

Remarks.

1. The composition by pullback in (11) is a little specific because the morphism $\bar{\phi}_{\mathcal{B}} : \hat{\Pi}_C(\mathcal{B}) \rightarrow C$ is a total function. Therefore, all transitions of this summary net are shared and must find a counterpart in the “ C part” of \mathcal{A} in order to be preserved after composition.
2. Observe that in this set-up the selected adequate order is simply set inclusion \subset^9 , which allows the comparison of basic configurations both in $\overline{Pref}_{\mathcal{A}}^{\Theta_*}$, in $\overline{Pref}_{\mathcal{B}}^{\Theta_*}$ and in $Pref_{\mathcal{N}}^{\Theta_*}$ in an analogous manner. This property will be crucial in the proof of Proposition 4.3.
3. Finally, observe also that $Pref_{\mathcal{A}}^{\Theta_*}$ is a prefix of $Unf_{\mathcal{A}} = Unf_{\mathcal{A}} \times_O^C Unf_{\hat{\Pi}_C(\mathcal{B})}$ and so the projection on \mathcal{A} can be applied to it, which by definition results in a true branching process of \mathcal{A} .

For the running example of this paper, the prefix factors of \mathcal{N} and their composition into a prefix of $Unf_{\mathcal{N}}$ are depicted in Figure 7. Compare in particular the factor corresponding to component \mathcal{A} in Figure 7(a) to the complete (extended) prefix of \mathcal{A} alone given in Figure 5(a): there are now two consecutive firings of t_2 (events e_1, e_4), which were necessary to a prefix of $Unf_{\mathcal{B}}$ to reach completeness. However, observe that the local cut-off decisions are quite conservative and may build branches of a MPF that are not strictly necessary to the other factor. For example, the cut-off event e_6 of $\overline{Pref}_{\mathcal{A}}^{\Theta_*}$ in Figure 7(a) is not reachable in $\overline{Pref}_{\mathcal{A}}^{\Theta_*} \times_O^C \overline{Pref}_{\mathcal{B}}^{\Theta_*}$. This is due to the fact that independent truncations

⁹It was pointed out by Walter Vogler that the order \subset is not adequate, and that the prefix built using \subset is a canonical prefix (despite \subset being not adequate) since \subset is included in an adequate order. Thus, \subset can play the role of an adequate order.

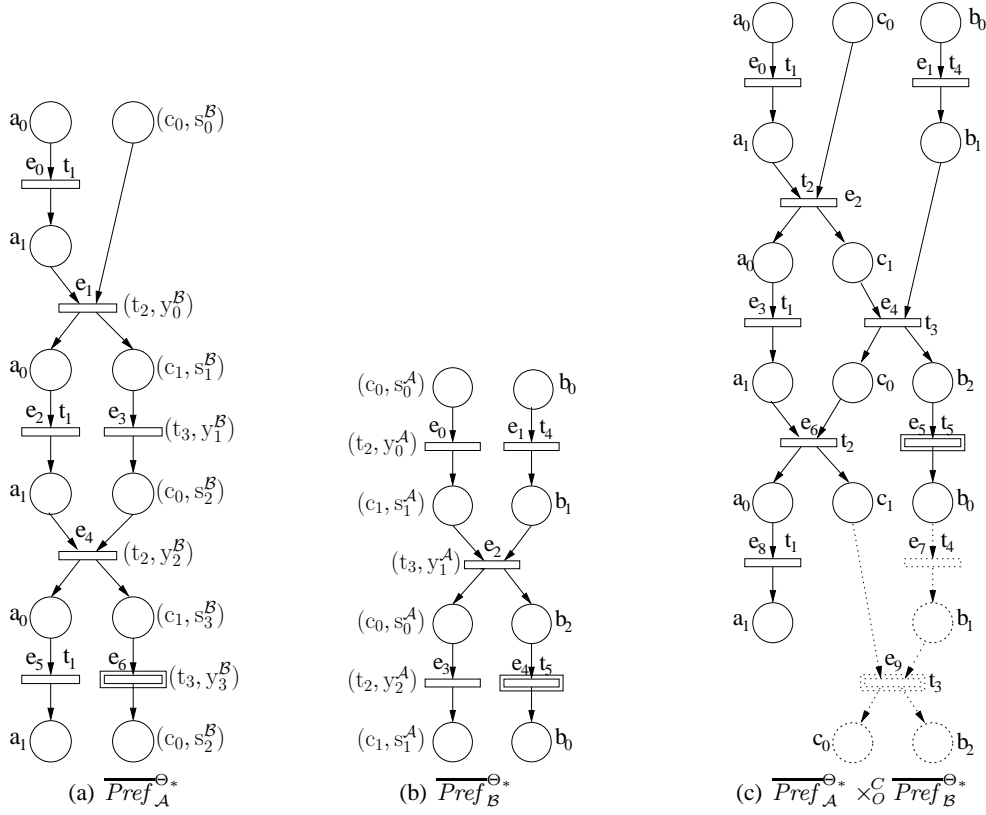


Figure 7. MPFs and their combination by pullback. Notice that event e_6 of factor (a) vanishes in the pullback. To preserve it, one would need the part of c in dashed lines, which is indeed useless to get a complete finite prefix of $Unf_{\mathcal{N}}$. Although the constructed factors are not strictly minimal, they nevertheless constitute a pullback covering of a complete prefix of \mathcal{N} .

are performed in the definition of the two MPF. As a consequence of this conservative cut-off criterion, by projecting back the composition $\overline{Pref}_A^{\Theta^*} \times_O^C \overline{Pref}_B^{\Theta^*}$ on the two MPF $\overline{Pref}_A^{\Theta^*}$ and $\overline{Pref}_B^{\Theta^*}$ one may get smaller factors. In other words, the two MPF do not correspond to the minimal pullback covering of their composition.

Lemma 4.1. If the summary net $\hat{\Pi}_C(\mathcal{B})$ does not reduce the behaviours of \mathcal{A} , that is if $Unf_{\mathcal{A}} = \Pi_{\mathcal{A}}(Unf_{\mathcal{A} \times_{\mathcal{N}}^C \hat{\Pi}_C(\mathcal{B})}) = \Pi_{\mathcal{A}}(Unf_{\mathcal{A}})$, then one has $Pref_{\mathcal{A}}^{\Theta^*} \subseteq \overline{Pref}_{\mathcal{A}}^{\Theta^*}$.

This is so because the summary net adds more constraints to compare markings in the construction of $Pref_{\mathcal{A}}^{\Theta^*}$, and thus to detect cut-off events. This result thus implies that when components \mathcal{A} and \mathcal{B} do not limit the behaviours of one another through the interface C , then the MPF of each component will be complete. However, in general this local completeness is not necessary to get the global completeness for \mathcal{N} , after composition of the two MPF. We now show that the two MPF do combine into a globally complete prefix of \mathcal{N} .

Proposition 4.3.

$$\Pi_{\mathcal{A}} \left(Pref_{\mathcal{N}}^{\Theta_*} \right) \triangleq \Pi_{\mathcal{A}} \left(Pref_{\mathcal{A} \times_N^C \mathcal{B}}^{\Theta_*} \right) \sqsubseteq \Pi_{\mathcal{A}} \left(Pref_{\mathcal{A} \times_N^C \hat{\Pi}_C(\mathcal{B})}^{\Theta_*} \right) \triangleq \Pi_{\mathcal{A}} \left(Pref_{\mathcal{A}}^{\Theta_*} \right) \triangleq \overline{Pref_{\mathcal{A}}^{\Theta_*}}$$

where \triangleq recalls relations deriving directly from definitions.

Recalling (9), this result implies that the two MPF $\overline{Pref_{\mathcal{A}}^{\Theta_*}}$ and $\overline{Pref_{\mathcal{B}}^{\Theta_*}}$ form a pullback covering¹⁰ of a finite complete prefix of $Unf_{\mathcal{N}}$.

Proof:

Let us first compare the projection on \mathcal{A} of the full unfoldings of \mathcal{N} and of $\overline{\mathcal{A}} = \mathcal{A} \times_N^C \hat{\Pi}_C(\mathcal{B})$, rather than their canonical prefixes. Given

$$Unf_{\mathcal{A} \times_N^C \hat{\Pi}_C(\mathcal{B})} = Unf_{\mathcal{A}} \times_O^C Unf_{\hat{\Pi}_C(\mathcal{B})} \quad (13)$$

one has

$$\begin{aligned} \Pi_{\mathcal{A}} (Unf_{\overline{\mathcal{A}}}) &= \Pi_{\mathcal{A}} \left(Unf_{\mathcal{A} \times_N^C \hat{\Pi}_C(\mathcal{B})} \right) \\ &= Unf_{\mathcal{A}} \times_O^C \Pi_C \left(Unf_{\hat{\Pi}_C(\mathcal{B})} \right) \\ &= Unf_{\mathcal{A}} \times_O^C \Pi_C (Unf_{\mathcal{B}}) \\ &= \Pi_{\mathcal{A}} (Unf_{\mathcal{A}} \times_O^C Unf_{\mathcal{B}}) \\ &= \Pi_{\mathcal{A}} (Unf_{\mathcal{N}}) \end{aligned} \quad (14)$$

The second equality uses (8), the third one derives from Prop. 4.2, the fourth equality uses (8) again and the last one corresponds to (4). This shows that $Unf_{\hat{\Pi}_C(\mathcal{B})}$ contains exactly the information about \mathcal{B} that is necessary to \mathcal{A} in order to determine what are its possible behaviours in \mathcal{N} . Equivalently, this result expresses that \mathcal{N} and $\overline{\mathcal{A}}$ have identical runs from the perspective of component \mathcal{A} .

Before comparing the projections of *canonical prefixes* of these unfoldings, let us make two remarks.

(a) Observe first that the events of $Unf_{\hat{\Pi}_C(\mathcal{B})}$ either synchronise with events of $Unf_{\mathcal{A}}$ or vanish in the pullback (13): in other words, the summary net $\hat{\Pi}_C(\mathcal{B})$ does not have private events, so for every event e of $Unf_{\overline{\mathcal{A}}}$ one has $\Pi_{\mathcal{A}}(e) \neq \emptyset$: no event of $Unf_{\overline{\mathcal{A}}}$ vanishes in the projection on \mathcal{A} . Consequently, configurations are preserved.

(b) Secondly, recall that the events of $Unf_{\hat{\Pi}_C(\mathcal{B})}$ are associated with the marking of \mathcal{B} they produced, as explained in the construction of summary nets (see remark 3 after Def. 4.3). So in the pullback (13), synchronised events are thus associated with a full marking of \mathcal{N} , obtained by merging the marking produced in \mathcal{B} to the marking produced in \mathcal{A} . This holds as well for events of (13) that are private to \mathcal{A} , provided one relies on the last synchronised event to get the marking produced on the \mathcal{B} part. Overall, every event of $Unf_{\overline{\mathcal{A}}}$ appears with the marking it would have produced in \mathcal{N} .

We now prove the inclusion of $\Pi_{\mathcal{A}} \left(Pref_{\mathcal{N}}^{\Theta_*} \right)$ into $\Pi_{\mathcal{A}} \left(Pref_{\overline{\mathcal{A}}}^{\Theta_*} \right)$.

Let e be a feasible event of $Unf_{\mathcal{N}}$, i.e. an event not strictly preceded by a cut-off event of the canonical FCP defined by cutting context Θ_* , and let e be such that $\Pi_{\mathcal{A}}(e) \neq \emptyset$. The configuration $\kappa = [e]$ in

¹⁰not necessarily minimal, as illustrated in the example above

$Unf_{\mathcal{N}}$ terminates with marking m of \mathcal{N} . By (14) there exists at least one configuration $\kappa' = [e']$ of $Unf_{\overline{\mathcal{A}}}$ (and possibly several) such that $\Pi_{\mathcal{A}}(\kappa) = \Pi_{\mathcal{A}}(\kappa')$. By remark (a) above, $\Pi_{\mathcal{A}}(\kappa')$ is isomorphic to κ' , so, with a light abuse of notations, one can write $\Pi_{\mathcal{A}}(\kappa) = \kappa'$. Moreover, by construction of the summary net $\tilde{\Pi}_C(\mathcal{B})$, one can choose $\kappa' = [e']$ in such a way that the marking associated with e' in $Unf_{\overline{\mathcal{A}}}$ is precisely equal to m , and similarly for all matching events in the isomorphism expressed by $\Pi_{\mathcal{A}}(\kappa) = \kappa'$. With this choice, event e' is necessarily feasible in $Unf_{\overline{\mathcal{A}}}$ for the cutting context Θ_* in $Unf_{\overline{\mathcal{A}}}$. Otherwise, let $f' \neq e'$ be a cut-off event in κ' and let g' be its corresponding event, both being associated with marking m' of \mathcal{N} . Since the cutting context Θ_* in $Unf_{\overline{\mathcal{A}}}$ relies on set inclusion as adequate order, one has $g' \in [f'] \sqsubset [e']$. f', g' are respectively related to events f, g in κ . So both f and g are associated with the same marking m' of \mathcal{N} , and moreover $f \in [g] \sqsubset [e]$. Set inclusion was also chosen to define the cutting context Θ_* in $Unf_{\mathcal{N}}$, which makes f a cut-off, provided f is feasible *i.e.* is not already preceded by a cut-off. In any case, this contradicts the fact that e is a feasible event in $Unf_{\mathcal{N}}$. In summary, this proves that configuration $\Pi_{\mathcal{A}}([e])$ for the feasible event e of $Unf_{\mathcal{N}}$ is present in $\Pi_{\mathcal{A}}(Pref_{\overline{\mathcal{A}}}^{\Theta_*})$, whence the result. \square

Notice that the inclusion stated in Proposition 4.3 may be strict. This happens for example when an event e of $Unf_{\mathcal{N}}$ is preceded by a cut-off event f that is private to \mathcal{B} . This cut-off makes e not feasible, but this phenomenon can not be seen in $Unf_{\overline{\mathcal{A}}}$, where events private to \mathcal{B} do not appear.

Proposition 4.4. $\overline{Pref}_{\overline{\mathcal{A}}}^{\Theta_*}$ is finite.

The proof is similar to the one of Proposition 4.1.

Theorem 4.1. $Pref_{\mathcal{N}}^{\Theta_*} \triangleq Pref_{\mathcal{A} \times_{\mathcal{N}}^C \mathcal{B}}^{\Theta_*} \sqsubseteq \overline{Pref}_{\mathcal{A}}^{\Theta_*} \times_O^C \overline{Pref}_{\mathcal{B}}^{\Theta_*}$.

Proof:

Immediate from (9) and Proposition 4.3. \square

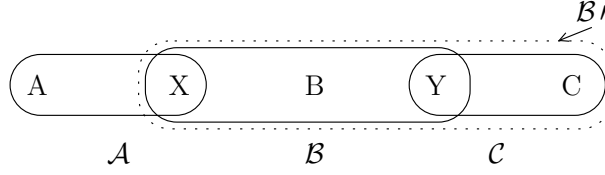
This result does not entail that the factors $\overline{Pref}_{\mathcal{A}}^{\Theta_*}$ and $\overline{Pref}_{\mathcal{B}}^{\Theta_*}$ provide the minimal pullback covering of $Pref_{\mathcal{N}}^{\Theta_*}$. First of all because one may have strict prefix inclusion in proposition 4.3, as it was noticed above. And secondly because the factors, computed separately, may not themselves satisfy

$$\overline{Pref}_{\mathcal{A}}^{\Theta_*} = \Pi_{\mathcal{A}}\left(\overline{Pref}_{\mathcal{A}}^{\Theta_*} \times_O^C \overline{Pref}_{\mathcal{B}}^{\Theta_*}\right)$$

and symmetrically for component \mathcal{B} , as shown by the counter-example in Fig. 7.

5. Tree-shaped systems

The above approach can be generalised to more complex tree-shaped systems having many components. This will be illustrated on the distributed system $\mathcal{N} = \mathcal{A} \times_{\mathcal{N}}^X \mathcal{B} \times_{\mathcal{N}}^Y \mathcal{C}$ shown in Figure 8, and that was already mentioned in Section 3.1 to illustrate the associativity of the pullback. System \mathcal{N} has three components which only interact through pairwise interfaces. This means that $\Lambda_{\mathcal{A}} \cap \Lambda_{\mathcal{B}} = \Lambda_X$, so there is no other interaction between the components \mathcal{A} and \mathcal{B} than the in interface X , and similarly

Figure 8. Distributed system $\mathcal{N} = \mathcal{A} \times_N^X \mathcal{B} \times_N^Y \mathcal{C}$.

$\Lambda_B \cap \Lambda_C = \Lambda_Y$, i.e. interface Y captures all interactions between \mathcal{B} and \mathcal{C} . Moreover, we assume $\Lambda_A \cap \Lambda_C \subseteq \Lambda_B$, so there is no direct interaction between \mathcal{A} and \mathcal{B} .

In the propositions below, it is shown that modular prefix factors of this three component system can be computed recursively using two rules: the so-called propagation rule and the merge rule. These rules can be combined to extend the computation procedure to any system with a tree interaction structure.

Proposition 5.1. (propagation rule)

Let $\mathcal{N} = \mathcal{A} \times_N^X \mathcal{B} \times_N^Y \mathcal{C} = \mathcal{A} \times_N^X \mathcal{B}'$ where $\mathcal{B}' = \mathcal{B} \times_N^Y \mathcal{C}$. Let us define the nets

$$\begin{aligned} \overline{\mathcal{A}} &= \mathcal{A} \times_N^X \hat{\Pi}_X \left(\left[\mathcal{B} \times_N^Y \hat{\Pi}_Y (\mathcal{C}) \right] \right) \\ \overline{\mathcal{A}'} &= \mathcal{A} \times_N^X \hat{\Pi}_X (\mathcal{B}') \end{aligned}$$

and the associated MPFs

$$\begin{aligned} \overline{Pref_{\mathcal{A}}^{\Theta^*}} &= \Pi_{\mathcal{A}} \left(Pref_{\overline{\mathcal{A}}}^{\Theta^*} \right) \\ \overline{Pref_{\mathcal{A}'}^{\Theta^*}} &= \Pi_{\mathcal{A}} \left(Pref_{\overline{\mathcal{A}'}}^{\Theta^*} \right) \end{aligned}$$

Then $\overline{Pref_{\mathcal{A}}^{\Theta^*}} = \overline{Pref_{\mathcal{A}'}^{\Theta^*}}$.

In other words, the summary nets can be computed recursively: The summary net sent from \mathcal{C} to \mathcal{B} on Y helps computing the next summary net from \mathcal{B}' to \mathcal{A} on X . This provides a way to obtain the MPF in \mathcal{A} with small scale computations, performed at the scale of a component: it is unnecessary to handle the big net \mathcal{B}' .

Proof:

One has $Unf_{\mathcal{B}'} = Unf_{\mathcal{B}} \times_O^Y Unf_{\mathcal{C}}$ by (4), whence $\Pi_{\mathcal{B}} (Unf_{\mathcal{B}'}) = Unf_{\mathcal{B}} \times_O^Y \Pi_Y (Unf_{\mathcal{C}})$ using properties of projections on unfoldings¹¹. From Proposition 4.2 one has $\Pi_Y (Unf_{\mathcal{C}}) = \Pi_Y (Unf_{\hat{\Pi}_Y(\mathcal{C})})$, so $\Pi_{\mathcal{B}} (Unf_{\mathcal{B}'}) = Unf_{\mathcal{B}} \times_O^Y \Pi_Y (Unf_{\hat{\Pi}_Y(\mathcal{C})}) = \Pi_{\mathcal{B}} (Unf_{\mathcal{B}} \times_O^Y Unf_{\hat{\Pi}_Y(\mathcal{C})}) = \Pi_{\mathcal{B}} (Unf_{\mathcal{B} \times_N^Y \hat{\Pi}_Y(\mathcal{C})})$. This results in $\Pi_X (Unf_{\mathcal{B}'}) = \Pi_X (Unf_{\mathcal{B} \times_N^Y \hat{\Pi}_Y(\mathcal{C})})$. In other words, the runs of $\mathcal{B}' = \mathcal{B} \times_N^Y \mathcal{C}$ and of $\mathcal{B} \times_N^Y \hat{\Pi}_Y(\mathcal{C})$ are identical from the perspective of interface X , but of course the second net is generally smaller than the first one since the private places of \mathcal{C} vanish in the construction of the summary net. As a consequence, one has $\Pi_{\mathcal{A}} (Unf_{\overline{\mathcal{A}}}^{\Theta^*}) = \Pi_{\mathcal{A}} (Unf_{\overline{\mathcal{A}'}}^{\Theta^*})$, which entails $\overline{Pref_{\mathcal{A}}^{\Theta^*}} = \overline{Pref_{\mathcal{A}'}^{\Theta^*}}$. \square

¹¹Actually, this is the propagation rule on unfoldings.

Proposition 5.2. (merge rule)

Let $\mathcal{N} = \mathcal{A} \times_N^X \mathcal{B} \times_N^Y \mathcal{C}$ and $\overline{\mathcal{B}} = \hat{\Pi}_X(\mathcal{A}) \times_N^X \mathcal{B} \times_N^Y \hat{\Pi}_Y(\mathcal{C})$. Then $\Pi_{\mathcal{B}}(Pref_{\mathcal{N}}^{\Theta^*}) \sqsubseteq \overline{Pref}_{\overline{\mathcal{B}}}^{\Theta^*} = \Pi_{\mathcal{B}}(Pref_{\overline{\mathcal{B}}}^{\Theta^*})$.

So the two lateral summary nets are sufficient to compute the MPF of \mathcal{N} in the central component \mathcal{B} . This proposition is expressed for two lateral components to \mathcal{B} , but it remains valid for any number of them.

Proof:

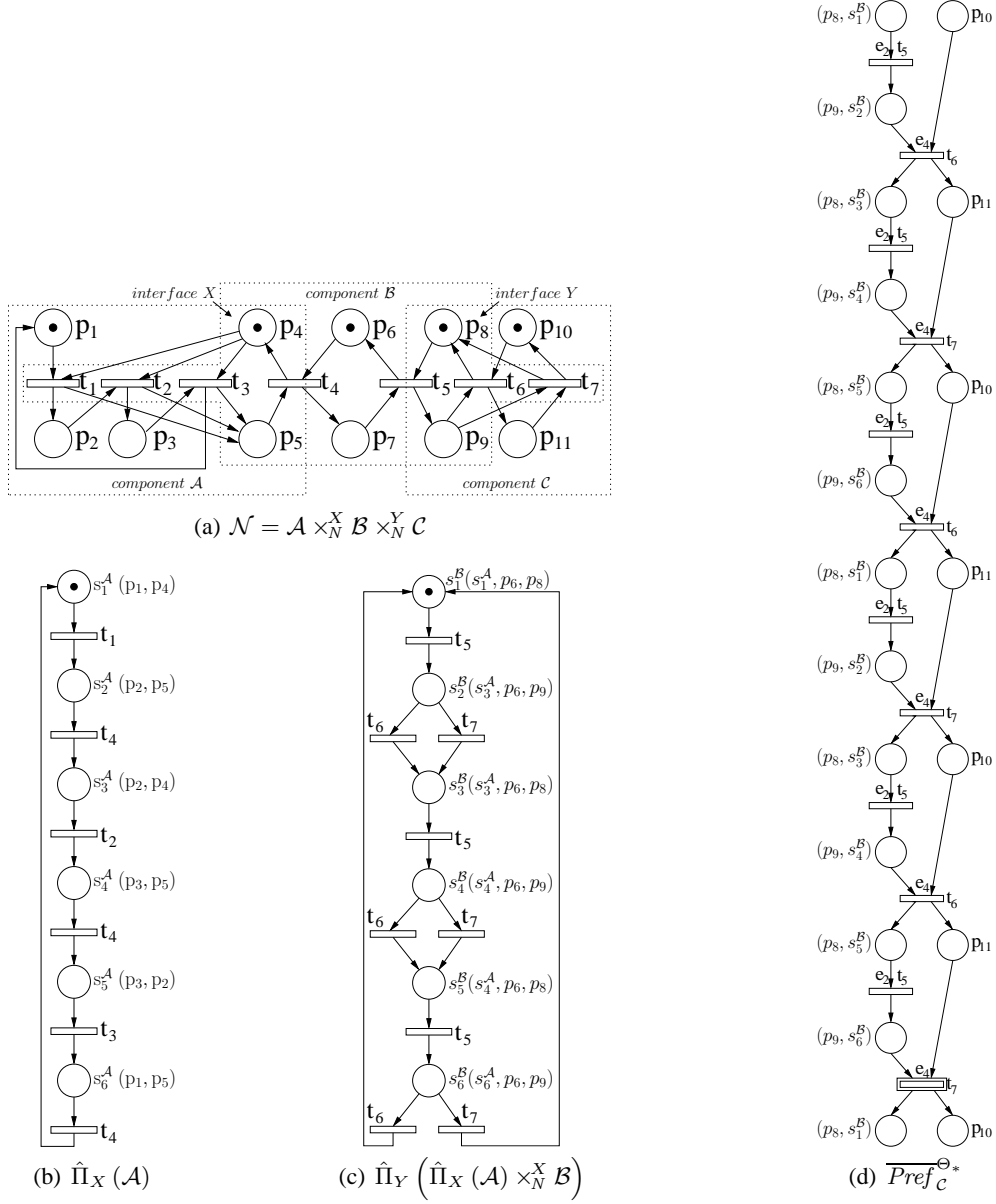
One can show that $\Pi_{\mathcal{B}}(Unf_{\mathcal{N}}) = \Pi_{\mathcal{B}}(Unf_{\overline{\mathcal{B}}})$ from Proposition 4.2 as follows. The summary nets received in \mathcal{B} on its interfaces X and Y give exactly the information of \mathcal{A} and \mathcal{C} , respectively, that is necessary to \mathcal{B} to recover its behaviours in \mathcal{N} . This means that \mathcal{N} and $\overline{\mathcal{B}}$ have identical runs from the point of view of \mathcal{B} .

By similar arguments as in Proposition 4.3, e.g. a cut-off event in $\Pi_{\mathcal{B}}(Unf_{\mathcal{N}})$ may occur before one in $\Pi_{\mathcal{B}}(Unf_{\overline{\mathcal{B}}})$, it is evident that $\Pi_{\mathcal{B}}(Pref_{\mathcal{N}}^{\Theta^*})$ is, in general, smaller than $Pref_{\overline{\mathcal{B}}}^{\Theta^*}$. \square

The modular construction of prefix factors is described above on an elementary string of components. But it extends naturally to nets having a tree-shape, for example with the classical message passing algorithm, employing two messages per edge, one in each direction. Specifically, computations are organised as follows. The objective is to obtain summary nets from all neighbours at each components. Summary nets are first computed starting at the leaves of the structure, and progressing inwards the tree, using first the propagation rule. Then, at each component, when summary nets are available on $n - 1$ interfaces out of the n of this component, the merge rule is applied (i.e. \mathcal{B} is replaced by $\overline{\mathcal{B}}$ as above), then the propagation rule yields the summary net to send on the last branch. This phase terminates when each component has received a summary net from each of its neighbours. It can then be checked that exactly two propagations have occurred on each branch, one in each direction. The final operation is a merge of all incoming summary nets at each component: this allows one to compute the MPF of this component. Note that at all steps in this computation are local and do not make use of any knowledge about the global system.

Figure 9(a) illustrates a simple tree-shaped system showing three components \mathcal{A} , \mathcal{B} , and \mathcal{C} , and their interfaces X and Y . It can also be seen as system having two components and an interface (consisting of places $\{p_6, p_7\}$ and transitions $\{t_4, t_5\}$). Observe that there is a length two cycle in the component \mathcal{C} , i.e. t_5 has to be executed twice to cover all reachable states in \mathcal{C} . Furthermore, this cycle is coupled to a length three cycle in \mathcal{A} , i.e. t_4 has to be executed three times. This propagation is illustrated with the computation of $\overline{Pref}_{\mathcal{C}}^{\Theta^*}$ in Figure 9(b)-(d).

The computation starts at the component \mathcal{A} . First, the summary net $\hat{\Pi}_X(\mathcal{A})$ is derived showing that three occurrences of t_4 are necessary for completeness (see Figure 9(b)). Then, $\hat{\Pi}_X(\mathcal{A})$ is sent to \mathcal{B} , where $\hat{\Pi}_Y(\hat{\Pi}_X(\mathcal{A}) \times_N^X \mathcal{B})$ is computed, which is depicted in Figure 9(c). Note that the three length cycle is also propagated; now t_5 needs three occurrences to be complete. Finally, as shown in Figure 9(d), $\overline{Pref}_{\mathcal{C}}^{\Theta^*}$ is computed using the received summary net $\hat{\Pi}_Y(\hat{\Pi}_X(\mathcal{A}) \times_N^X \mathcal{B})$. The component \mathcal{C} has to be executed six times to be globally complete. This is the result of the propagated three length cycle from $\hat{\Pi}_X(\mathcal{A})$, which is coupled with the two length cycle in \mathcal{C} .

Figure 9. Computing MPF of \mathcal{C} for the distributed system $\mathcal{N} = \mathcal{A} \times_N^X \mathcal{B} \times_N^Y \mathcal{C}$.

The remaining MPFs are computed by starting deriving the summary nets from \mathcal{C} . In that case $\hat{\Pi}_Y(\mathcal{C})$ carries the information that it needs two occurrences of Y , while $\hat{\Pi}_X(\mathcal{B} \times_N^Y \hat{\Pi}_Y(\mathcal{C}))$ need four occurrences of X to be complete. Then, the MPFs are computed; $\overline{Pref}_A^{\ominus*}$ has eight occurrences of t_4 , and $\overline{Pref}_B^{\ominus*}$ has seven occurrences of t_5 . In both cases, they are over complete (by one transition) when comparing with $\overline{Pref}_A^{\ominus*} \times_O^X \overline{Pref}_B^{\ominus*} \times_O^Y \overline{Pref}_C^{\ominus*}$ due to independent truncation.

6. Conclusions

This paper has presented a novel approach to address the construction of finite complete prefixes (FCP) of Petri net (PN) unfoldings. It specialises to distributed systems, i.e. PN expressed as networks of components, and provides the FCP in factorised form, that is it determines finite behaviours of each component that, put together, are sufficient to build a FCP of the global system. Moreover, the construction of these factors of the FCP takes the form of modular computations performed at the scale of a single component. Specifically, components communicate summary nets to their neighbours, that represent the behaviours that they allow on their interface with this neighbour.

Apparently, the computation scheme requires several rounds of unfolding in each component, one with the component alone, another one with the component coupled to the received summary net. The second round of unfoldings can of course benefit from the first one, which reduces the computation overhead. Interestingly, once summary nets are computed, some model-checking tasks may become simpler. For example when one component is replaced by another “implementation” the behaviours of the new version in the global system can be determined immediately, without re-unfolding the global system. The same happens when a new component is connected to a distributed system. This possibility to re-use previous computations after a local system update is of course extremely desirable, and one of the motivations for this approach.

The technique was essentially described for a pair of components, but it naturally extends to tree-shaped systems, as explained in the last section. The next efforts on this theme will follow three directions. A technical one, first: the generalisation to more complex interfaces, that would be nets instead of automata. A second direction will explore the existence of more appropriate notions of adequate order, that would be suited to distributed systems and would potentially provide smaller modular FCP. Finally, regarding complexity issues, most questions remain open. It is not clear how much one saves by distributed computations compared to the centralised computation of an FCP. But some hints and examples suggest progress can still be made. In particular, the summary nets used in this paper are obtained from *restrictions* of prefixes to the interface, not from *projections* on the interface. This means that the trimming operation that merges isomorphic configurations in the projection is not performed. Consequently, some isomorphic events that are in conflict in distant components may still be both propagated through summary nets, although they do not give access to new behaviours. Our efforts will be dedicated to understanding better this phenomenon, in order to obtain the “slimmest” possible summary net.

Acknowledgements

Many thanks to Victor Khomenko and Walter Vogler for fruitful discussions related to this topic. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions.

References

- [1] Baldan, P., Haar, S., König, B.: Distributed Unfolding of Petri Nets, *FoSSaCS'06*, LNCS 3921, pp. 126–141, 2006.
- [2] Couvreur, J.-M., Grivet, S., Poitrenaud, D.: Unfolding of Products of Symmetrical Petri Nets, *Lecture Notes in Computer Science*, **2075**, 2001, pp. 121–143.
- [3] Engelfriet, J.: Branching Processes of Petri Nets., *Acta Informatica*, **28**, 1991, pp. 575–591, NewsletterInfo: 38, 40.
- [4] Esparza, J., Heljanko, K.: *Unfoldings: a Partial-Order Approach to Model Checking*, EATCS monographs on Theoretical Computer Science, Springer-Verlag, 2008.
- [5] Esparza, J., Römer, S.: An Unfolding Algorithm for Synchronous Products of Transition Systems, *International Conference on Concurrency Theory*, LNCS 1664, pp. 2-20, 1999.
- [6] Esparza, J., Römer, S., Vogler, W.: An Improvement of McMillan's Unfolding Algorithm, *Formal methods in systems design*, 2002.
- [7] Fabre, E.: *Factorization of Unfoldings for Distributed Tile Systems, Part 1 : Reduced Interaction Case*, Technical Report 1529, IRISA, April 2003.
- [8] Fabre, E.: *Factorization of Unfoldings for Distributed Tile Systems, Part 2 : General Case*, Technical Report 1606, IRISA, May 2004.
- [9] Fabre, E.: Distributed Diagnosis based on Trellis Processes, *44th Conf. on Decision and Control (CDC), Seville, Spain*, December 2005.
- [10] Fabre, E.: On the Construction of Pullbacks for Safe Petri Nets, *Applications and Theory of Petri Nets and other Models of Concurrency, ATPN'06, Turku, Finland*, June 2006.
- [11] Fabre, E.: Bayesian Networks of Dynamic Systems, *Habilitation Thesis*, available at http://www.irisa.fr/distribcom/Personal_Pages/fabre/Papiers/hdr.pdf, June 2007.
- [12] Fabre, E., Benveniste, A., Haar, S., Jard, C.: Distributed Monitoring of Concurrent and Asynchronous Systems, *Journal of Discrete Event Systems, special issue*, May 2005, pp. 33–84.
- [13] Khomenko, V.: *Model Checking Based on Petri Net Unfolding Prefixes*, Ph.D. Thesis, University of Newcastle upon Tyne, 2002.
- [14] Khomenko, V., Koutny, M., Vogler, W.: Canonical Prefixes of Petri Net Unfoldings, *Acta Informatica, Volume 40, Number 2*, October 2003, pp. 95–118.
- [15] McMillan, K. L.: *Symbolic Model Checking: an approach to the state explosion problem*, Ph.D. Thesis, 1992.
- [16] McMillan, K. L.: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits, *Proc. International Workshop on Computer Aided Verification*, July 1992.
- [17] Nielsen, M., Plotkin, G., Winskel, G.: Petri Nets, Event Structures and Domains, Part I, *Theor. Computer Science*, **13**(1), January 1980, pp. 85–108.
- [18] Winskel, G.: A New Definition of Morphism on Petri Nets, *STACS '84: Proceedings of the Symposium of Theoretical Aspects of Computer Science*, Springer-Verlag, London, UK, 1984, ISBN 3-540-12920-0.
- [19] Winskel, G.: Categories of Models for Concurrency, *Seminar on Concurrency, Carnegie-Mellon University*, Springer-Verlag, London, UK, 1985, ISBN 3-540-15670-4.
- [20] Winskel, G.: Petri Nets, Algebras, Morphisms, and Compositionality, *Information and Computation*, **72**, 1987, pp. 197–238.