Modular construction of finite and complete prefixes of Petri net unfoldings

Agnes Madalinski, Eric Fabre

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Modular construction of finite and complete prefixes of Petri net unfoldings

Agnes Madalinski, Eric Fabre

N° 6412

January 2008

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6412--FR+ENG

Modular construction of finite and complete prefixes of Petri net unfoldings

Agnes Madalinski, Eric Fabre

Thème COM — Systèmes communicants
Projets DistribCom

Rapport de recherche n° 6412 — January 2008 — 20 pages

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 know 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 provides a more compact representation of the system trajectories, and 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 prefix" is based on deriving summaries of component behaviours w.r.t. their interfaces, that are communicated to the neighbouring components. The latter integrate them to 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.

Key-words: Petri net, unfolding, finite complete prefix, distributed system, pullback, category theory

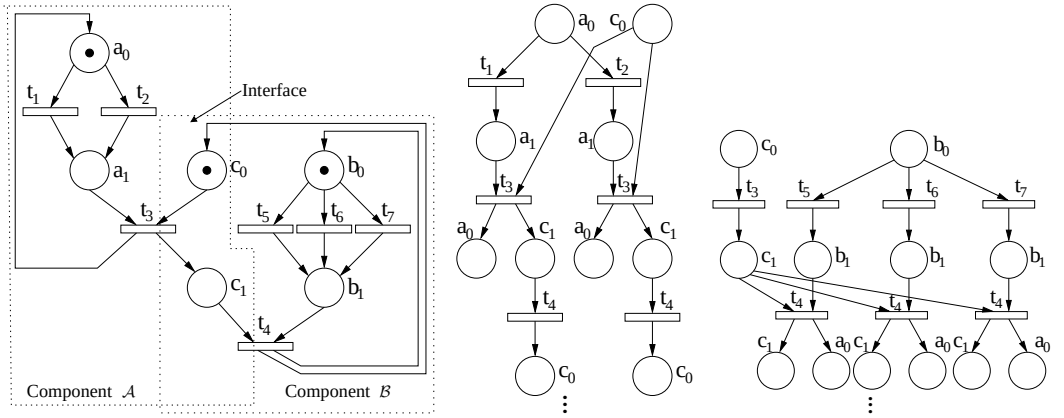
Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Construction modulaire de préfixes finis complets de dépliages

Résumé : Ce papier considère des systèmes distribués, définis comme des collections de composants reliés par des interfaces. Composants, interfaces et systèmes distribués sont modélisés par des réseaux de Petri. Il est bien connu que le dépliage d'un tel système distribué jouit d'une propriété de factorisation : il peut en effet s'exprimer comme la composition des dépliages des différents composants du système. Cette représentation factorisée du dépliage est en général plus compacte que le dépliage global, et permet en outre d'analyser le système par morceaux.

L'objectif de ce papier est la construction de préfixes finis complets (PFC) dans les dépliages de systèmes distribués. Plus précisément, on cherche à obtenir directement de tels préfixes sous forme factorisée, en évitant de construire d'abord un préfixe complet dans le dépliage du système global pour ensuite le factoriser. Cette construction de préfixes sous forme modulaire se fonde sur la notion de résumé du comportement d'un composant par rapport à son (ses) interface(s). Ces résumés sont communiqués aux composants voisins, qui les intègrent à leur propre dépliage pour construire des résumés vers les composants suivants, et ainsi de suite. Le tout prend la forme d'un algorithme à circulation de messages entre composants, où le système global n'est jamais manipulé.

Mots-clés : réseau de Petri, dépliage, préfixe fini complet, système distribué, produit fibré, théorie des catégories



(a) $\mathcal{N}$ (b) $Unf_{\mathcal{A}}$ (c) $Unf_{\mathcal{B}}$

Figure 1: Distributed system and the unfoldings of its components

1 Introduction

Petri nets (PN) form a widely used model class 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, 14] 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, however, but rather on a finite and complete prefix (FCP) of it [5, 11, 13]. 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 [10, 12].

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, 6, 7, 16]. 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, as illustrated in Figure 1. The system in Figure 1(a) consists of two components, $\mathcal{A}$ and $\mathcal{B}$, interacting through an interface. 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)), hence $m \cdot n = 6$ different combinations in the unfolding of the global system.

The factorisation property of unfoldings was first mentioned by Winskel in [16], but only found its first applications around 2005, in diagnosis problems [9]. A closely related problem consists in computing the minimal factorised form of an unfolding, and was considered at the same period [6, 7, 1]. 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 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 the derivation of tools for modular model checking. The problem amounts to building FCP of 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 labeled transition systems was presented in [4]. 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, taking into account the symmetries of components. However, the systems considered there are restricted to non-reentrant synchronisations, i.e. one transition of a component can synchronise only with one transition in another component. This restriction kills the compression gain of factorised forms: the collection of factors of the global unfolding has the same size as the global unfolding itself. In addition, the derivation of local prefixes uses global information. In this paper, the objective is to use only local information, i.e. the component model plus information communicated by 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, 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 (the end of the paper suggests how this paradigm can then be extended). The interface is assumed to be a simple automaton rather than a true PN, and the prefix construction assumes marking completeness, and 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 contribution of this paper: it presents the modular derivation of a finite and complete prefix in factorised form, focusing first on the simple case of two components, and then proposing the generalization to more complex networks of components.

2 Nets and unfoldings

In this section basic definitions concerning Petri nets and net unfoldings are presented. These are mainly adapted from [5, 11].

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*, and $P^0 : P \rightarrow$

$\mathbb{N} = \{0, 1, 2, ...\}$ is a multiset on P representing the *initial marking* of the net. 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 $x^\bullet = \{y \mid (x, y) \in \rightarrow\}$. In this work the nets are limited to *safe* nets, i.e. for every reachable marking M and every place $p \in P$, $M(p) \subseteq \{0, 1\}$.

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

Morphisms. A morphism $\phi : A \rightarrow B$ between nets $x = \left(P_x, T_x, \rightarrow_x, P_x^0\right)$ [15], where $x \in \{A, B\}$, is a pair $\left(\phi^P, \phi^T\right)$ with ϕ^P a relation on places and ϕ^T a partial function on transitions[1]. The initial marking is preserved by ϕ as follows: $P_B^0 = \phi\left(P_A^0\right)$ and $\forall p_B \in P_B^0, \exists! p_A \in P_A^0 : p_A \phi p_B$. If ϕ is defined on $p_A \in P_A$, then it is also defined on both ${}^\bullet p_A$ and $p_A^\bullet$; ϕ preserves the environment of each transition: $t_B = \phi(t_A)$ implies that restrictions $\phi^{op} : {}^\bullet t_B \rightarrow {}^\bullet t_A$ and $\phi^{op} : t_B^\bullet \rightarrow t_A^\bullet$ are both total functions, where ϕ^{op} denotes the reversed relation on places. Notice that net morphisms preserve runs.

For labeled nets $x = \left(P_x, T_x, \rightarrow_x, P_x^0, \lambda_x, \Lambda_x\right)$ the definition of a morphism $\phi : A \rightarrow B$ is reinforced by extra requirements: $\Lambda_B \subseteq \Lambda_A$ (the label set is reduced by ϕ), $Dom(\phi_T) = \lambda_A^{-1}(\Lambda_B) \subseteq T_A$ (ϕ is defined exactly on transitions having a shared label), and if $\phi_T(t_A) = t_B$ then $\lambda_A(t_A) = \lambda_B(t_B)$ (label preserving).

2.2 Branching processes

Two nodes of a net $\mathcal{N}$, y and y', are in *structural conflict*, denoted by $y \# y'$, if there exist distinct transitions $t, t' \in T$ such that ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$, and (t, y) and (t', y') are in the reflexive transitive closure of the flow relation $\rightarrow$, denoted by $\preceq$.

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: for every $c \in C$, $\mid {}^\bullet c \mid \leq 1$; for every $y \in C \cup E$, $\neg(y \# y)$ and there are finitely many y' such that $y' \prec y$, where $\prec$ denotes the *causal relation*, the transitive irreflexive closure of $\rightarrow$. Two nodes are *concurrent*, denoted $y \parallel y'$, if neither $y \# y'$ nor $y \preceq y'$ nor $y' \preceq y$.

Branching processes. A *branching process* of a net system $\mathcal{N}$ is a pair $\beta = (\mathcal{O}, f)$, where morphism $f : \mathcal{O} \rightarrow \mathcal{N}$ is a total function on $\mathcal{O}$, also called a *folding* of $\mathcal{O}$ into $\mathcal{N}$. This folding can be seen as a labeling function on events and conditions of $\mathcal{O}$, by which configurations of $\mathcal{O}$ represents runs of $\mathcal{N}$. It is further required that β satisfy a parsimony condition : for all $e_1, e_2 \in E$, if ${}^\bullet e_1 = {}^\bullet e_2$ and $f(e_1) = f(e_2)$ then $e_1 = e_2$. To define finite complete prefixes, it will be useful to consider a (virtual) initial event $\perp$ in β, which has an empty preset, C^0 as post-set and no label (i.e. no image by f).

A branching process (BP) β' of $\mathcal{N}$ is a *prefix* of a branching process β, denoted by $\beta' \sqsubseteq \beta$, if $\mathcal{O}'$ is a causally closed sub-net of $\mathcal{O}$ containing all initial conditions and such that: $\forall e \in E, e \in \mathcal{O}'$ implies $e^\bullet \subseteq \mathcal{O}'$ and f' is the restriction of f to $C' \cup E'$. For each net system $\mathcal{N}$ 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 $\mathcal{N}$. In the sequel, when foldings are unambiguous they will be omitted, so β will be identified with $\mathcal{O}$.

Given a configuration κ of a branching process β of $\mathcal{N}$, ϵ is called a *suffix* of κ in β iff there exists some configuration $\kappa' \sqsupset \kappa$ such that $\epsilon = \kappa' \setminus \kappa$.

[1] For simplicity, ϕ is written instead of ϕ^P or ϕ^T.

Configurations and cuts. A *configuration* of a branching process β is a finite set of events $\kappa \subseteq E$ such that for all $e, e' \in \kappa$, $\neg(e \# e')$ and, for every $e \in \kappa$, $e' \prec e$ implies $e' \in \kappa$; in addition it is required that $\perp \in \kappa$. For every event $e \in E$, the configuration $[e] \stackrel{\text{df}}{=} \{e' | e' \preceq e\}$ is called the *basic configuration*[2] of e, and $\langle e \rangle \stackrel{\text{df}}{=} [e] \setminus \{e\}$ denotes the set of *causal predecessors*. The set of all finite (resp. basic) configurations of a branching process β is denoted by $\boldsymbol{\kappa}^{\beta}_{fin}$ (resp. $\boldsymbol{\kappa}^{\beta}_{bas}$), and the superscript β is dropped when $\beta = Unf_{\mathcal{N}}$.

A *co-set* is a set of condition C' such that for all distinct $c, c' \in C'$, $c \parallel c'$, and a *cut* is a maximal co-set for the set inclusion. Let κ be a configuration then $Cut(\kappa) \stackrel{\text{df}}{=} \left(C^0 \cup \kappa^{\bullet}\right) \setminus {}^{\bullet}\kappa$ is a cut; furthermore, the set of places $f\left(Cut(\kappa)\right)$ is a reachable marking of $\mathcal{N}$, which is denoted by $Mark(\kappa)$. A marking M of $\mathcal{N}$ is *represented* in β if there is a configuration κ of β such that $M = Mark(\kappa)$. Every marking represented in β is reached in $\mathcal{N}$, and every reachable marking of $\mathcal{N}$ is represented in $Unf_{\mathcal{N}}$.

2.3 Finite and complete prefixes

An abstract parametric model has been introduced in [10, 11] to cope with different variants of truncating unfoldings. It uses parameters which determine the information intended to be preserved in the complete prefix (in the standard case, this is the set of reachable markings) and specify the circumstances under which an event can be designated as a cut-off event.

Definition 1 *A* cutting context *for $Unf_{\mathcal{N}}$ is a triple $\Theta = \left(\approx, \lhd, \{\boldsymbol{\kappa}_e\}_{e \in E}\right)$, where:*

1. *$\approx$ is an equivalence relation on $\boldsymbol{\kappa}_{fin}$.*
2. *$\lhd$, called an* adequate *order, is a strict well-founded partial order on $\boldsymbol{\kappa}_{fin}$ refining $\subset$, i.e. $\kappa' \subset \kappa''$ implies $\kappa' \lhd \kappa''$.*
3. *$\approx$ and $\lhd$ 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 $\kappa'' \lhd \kappa'$ then $\kappa'' \oplus \epsilon'' \lhd \kappa' \oplus \epsilon'$.*
4. *$\{\boldsymbol{\kappa}_e\}_{e \in E}$ is a family of subsets of $\boldsymbol{\kappa}_{fin}$.*

The adequate order specifies which configurations are preserved in the complete prefix; all $\lhd$-minimal configurations in each equivalent class of $\approx$ are preserved. The last parameter $\boldsymbol{\kappa}_e$ is needed to specify the set of configurations used later to decide whether an event can be designed as a cut-off event. The cutting context $\Theta_{ERV} = \left\{\approx_{mar}, \lhd_{tot}, \{\boldsymbol{\kappa}_e = \boldsymbol{\kappa}_{bas}\}_{e \in E}\right\}$ corresponds to the framework in [5], where $\approx_{mar}$ is the equivalence relation on reachable marking of $\mathcal{N}$, i.e. $\kappa' \approx_{mar} \kappa''$ iff $Mark(\kappa') = Mark(\kappa'')$, and $\lhd_{tot}$ is a total adequate order.

The static cut-off events are defined independently of an unfolding algorithm, where feasible events are events whose causal predecessors are not static cut-off events and thus are included in the prefix determined by those cut-off events.

Definition 2 *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 inductively, in the following way:*

1. *An event e is a feasible event if $\langle e \rangle \cap cut^{\Theta} = \emptyset$.*

[2] The 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."

2. *An event e is a static cut-off event if it is feasible, and if there is a (so called* corresponding*) configuration $\kappa \in \boldsymbol{\kappa}_e$ such that $\kappa \subseteq fsble^{\Theta} \setminus cut^{\Theta}$, $\kappa \approx [e]$, and $\kappa \lhd [e]$. (An event e' is referred as* corresponding event *if $\kappa = [e']$.)*

The branching process $Pref_{\mathcal{N}}^{\Theta}$ induced by the set of events $fsble_{\mathcal{N}}^{\Theta}$ is called the canonical prefix *of $Unf_{\mathcal{N}}$.*

Note that $Pref_{\mathcal{N}}^{\Theta}$ is uniquely determined by the cutting context Θ. Several fundamental properties of $Pref_{\mathcal{N}}^{\Theta}$ have been proven in [11]. In particular, $Pref_{\mathcal{N}}^{\Theta}$ is always complete w.r.t. $cut_{\mathcal{N}}^{\Theta}$, and it is finite if $\approx$ has finitely many equivalence classes and $\boldsymbol{\kappa}_e \supseteq \boldsymbol{\kappa}_{bas}$.

3 Distributed systems

Distributed systems are modeled by a set of interconnected components interacting through shared sub-systems, called interfaces. This modular structure is also called a "factorisation property" of the system, due to the use of products (more precisely of pullbacks) to connect components. This factorisation property is inherited by the unfolding of the distributed system, which makes it possible to process such systems in a modular manner [9].

3.1 Compound system

Pullbacks. A distributed system is formally expressed as a special case of a pullback [8].

Definition 3 *Let $\mathcal{A}, \mathcal{B}$ and C be labeled nets, where $\mathcal{A}$ and $\mathcal{B}$ are related to their interface C by morphisms $\phi_{\mathcal{A}} : \mathcal{A} \to C$ and $\phi_{\mathcal{B}} : \mathcal{B} \to C$ that are partial functions. Moreover, assume $\Lambda_C \supseteq \Lambda_{\mathcal{A}} \cap \Lambda_{\mathcal{B}}$. The* pullback *of this triple, or better the pullback of diagram $\mathcal{A} \overset{\phi_{\mathcal{A}}}{\to} C \overset{\phi_{\mathcal{B}}}{\leftarrow} \mathcal{B}$, is denoted by $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ and defined on places by*

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

and on transitions by

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

The flow relation follows accordingly as well as the definition of initial places.

Associated to this pullback construction, there exist canonical morphisms $\psi_{\mathcal{A}} : \mathcal{N} \to \mathcal{A}$ and $\psi_{\mathcal{B}} : \mathcal{N} \to \mathcal{B}$ that map elements of $\mathcal{N}$ to the corresponding elements in $\mathcal{A}$ and $\mathcal{B}$, respectively, and that satisfy $\phi_{\mathcal{A}} \circ \psi_{\mathcal{A}} = \phi_{\mathcal{B}} \circ \psi_{\mathcal{B}}$.

Distributed systems. A net $\mathcal{N}$ is said to be a *distributed system* if it can be expressed as a pullback of several components where the intermediary nets are interfaces. 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. For simplicity in this paper a distributed system $\mathcal{N}$ is limited to two components $\mathcal{A}$ and $\mathcal{B}$ and an interface C, which can then be extended to a graph. Figure 2 depicts $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ with associated morphisms.

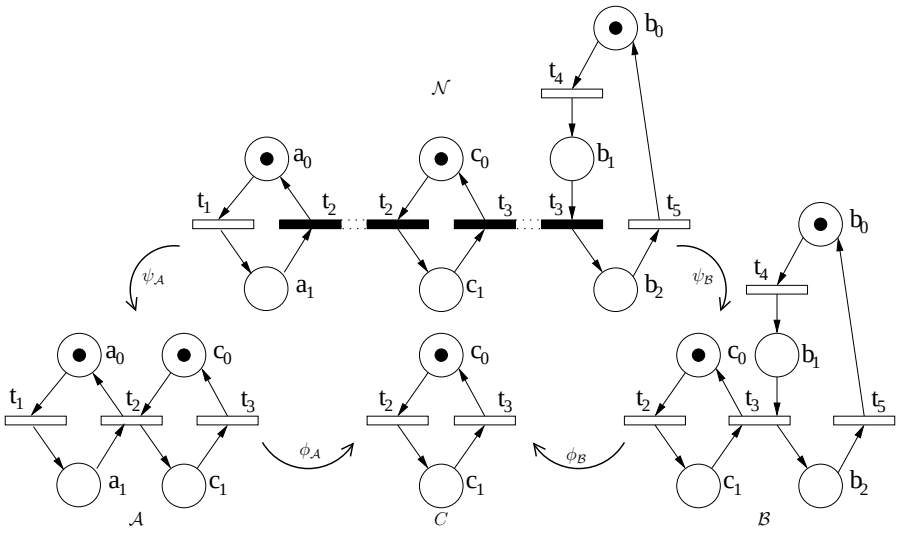


Figure 2: The pullback $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$

3.2 Factorisation of unfoldings

It was shown in [16] that the factorised form of a net system yields a factorised form of the unfolding of such a system. Given $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ one obtains

$$Unf_{\mathcal{N}} = Unf_{\mathcal{A}} \times_O^C Unf_{\mathcal{B}} \tag{3}$$

where $Unf_{\mathcal{N}}$, $Unf_{\mathcal{A}}$ and $Unf_{\mathcal{B}}$ are the unfoldings of $\mathcal{N}$, $\mathcal{A}$ and $\mathcal{B}$, and $\times_O^C$ denotes the pullback on branching processes, with Unf_C as interface. Thus, the unfolding of the global system can be expressed as the pullback of the unfoldings of its components (as illustrated in Figure 3).

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

$$\mathcal{O}_{\mathcal{A}} \times_O^C \mathcal{O}_{\mathcal{B}} = Unf\left(\mathcal{O}_{\mathcal{A}} \times_N^{Unf_C} \mathcal{O}_{\mathcal{B}}\right) \tag{4}$$

Moreover, it is easily checked that there exists a folding f from $\mathcal{O}_{\mathcal{A}} \times_O^C \mathcal{O}_{\mathcal{B}}$ that makes it a branching process of $\mathcal{A} \times_N^C \mathcal{B}$. Notice that (4) 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).

Projections. Given $Unf_{\mathcal{N}} = Unf_{\mathcal{A}} \times_O^C Unf_{\mathcal{B}}$, some nodes of $Unf_{\mathcal{N}}$ are labelled by places and transitions of C, through any of the (commuting) paths relating $Unf_{\mathcal{N}}$ to C in Figure 3. The *restriction* of $Unf_{\mathcal{N}}$ to these nodes is denoted by $(Unf_{\mathcal{N}})_{|C}$. By contrast, the *projection* of $Unf_{\mathcal{N}}$ on behaviours of C is defined as the image of $Unf_{\mathcal{N}}$ in Unf_C:

$$\Pi_C\left(Unf_{\mathcal{N}}\right) = \phi_{\mathcal{A}}^{\mathcal{O}} \circ \psi_{\mathcal{A}}^{\mathcal{O}}\left(Unf_{\mathcal{N}}\right) = \phi_{\mathcal{B}}^{\mathcal{O}} \circ \psi_{\mathcal{B}}^{\mathcal{O}}\left(Unf_{\mathcal{N}}\right). \tag{5}$$

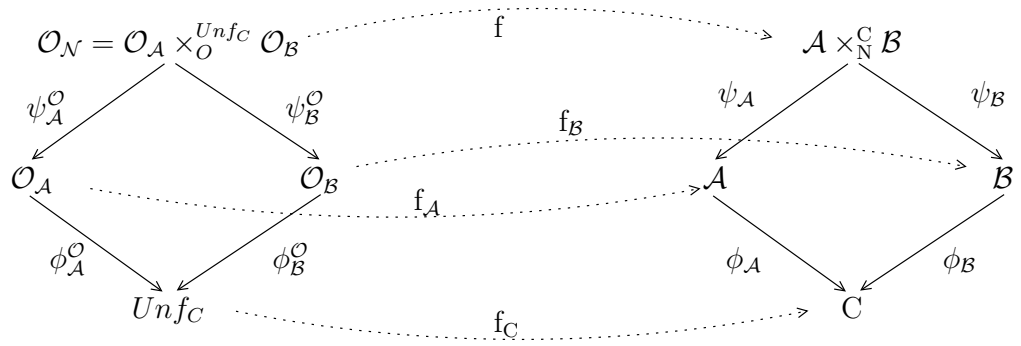


Figure 3: Commutative diagram of the pullbacks of branching processes and nets, and their relation by associated foldings.

Alternatively, this projection can be obtained by taking first $(Unf_\mathcal{N})_{|C}$ and then performing a *trimming,* i.e. by merging isomorphic configurations in $(Unf_\mathcal{N})_{|C}$ in order to get a valid branching process of C. Projections on $\mathcal{A}$ and $\mathcal{B}$ of $Unf_\mathcal{N}$ are defined similarly as its images by $\psi_\mathcal{A}^\mathcal{O}$ and $\psi_\mathcal{B}^\mathcal{O}$ respectively. Projections naturally extend to any BP of $\mathcal{N}$.

Notice that the restriction of $Unf_\mathcal{N}$ to a subset of nodes may erase causality or conflict relations between these nodes, which may then appear as concurrent in $(Unf_\mathcal{N})_{|C}$ whereas they were not in $Unf_\mathcal{N}$. The same phenomenon occurs as well with projections. Therefore a projection $\Pi_C(Unf_\mathcal{N})$ is said to be *non misleading* iff every configuration κ' in $\Pi_C(Unf_\mathcal{N})$ is the image of a configuration of κ in $Unf_\mathcal{N}$, and causality relations on events of κ' are not lost. Observe that projections on interfaces C that are automata, i.e. Petri nets without concurrency, are by definition 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 [7] 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 center 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 automata.

Minimal pullback covering. Projections allow us to build the *minimal pullback covering* of $Unf_\mathcal{N}$, which restricts the factors $Unf_\mathcal{A}$ and $Unf_\mathcal{B}$ to the behaviours of components $\mathcal{A}, \mathcal{B}$ that remain feasible in the entire system $\mathcal{N}$. One can write

$$Unf_\mathcal{N} = \Pi_\mathcal{A}(Unf_\mathcal{N}) \times_O^C \Pi_\mathcal{B}(Unf_\mathcal{N}) \tag{6}$$

and each *minimal factor* $\Pi_x(Unf_\mathcal{N})$, where $x \in \{\mathcal{A}, \mathcal{B}\}$, is a prefix of Unf_x. These factors are minimal in the sense that taking any strict prefix of them would prevent recovering the full $Unf_\mathcal{N}$ by pullback.

Minimal factors can be computed in a modular manner without computing $Unf_\mathcal{N}$ itself. One has

$$\Pi_\mathcal{A}(Unf_\mathcal{N}) = Unf_\mathcal{A} \times_O^C \Pi_C(Unf_\mathcal{B}) \tag{7}$$

(and symmetrically for $\Pi_\mathcal{B}(Unf_\mathcal{N})$) [6, 7]. 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 $\mathcal{N}$. (7) holds, and thus allows one to perform a modular computation of minimal factors, as soon as projections on C are non misleading. It is precisely

to obtain a similar property for any interface net C that *augmented branching processes* were developed in [7], while [1] later proposed the alternate technique of *interleaving structures*.

For more complex distributed systems, taking the shape of a network of components, 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 [6, 7] for details.

4 Modular canonical 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 the case of two components $\mathcal{A}$ and $\mathcal{B}$ interacting through a common interface C, which ensures $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ as it was seen above. At the end of this section, however, the generalisation to more complex (tree-shaped) systems will be examined. It is furthermore assumed that the interface C is an automaton, not a general safe net, in order to avoid the extra technical difficulty of dealing with misleading projections.

A trivial solution to obtain an 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 as follows [3]

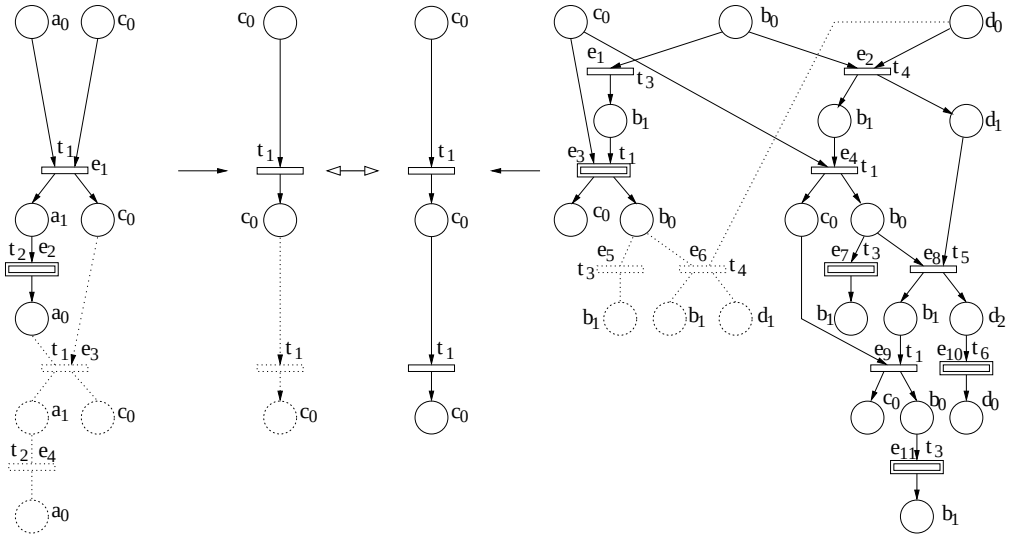
$$Pref_{\mathcal{N}} \sqsubseteq \Pi_{\mathcal{A}}\left(Pref_{\mathcal{N}}\right) \times_O^C \Pi_{\mathcal{B}}\left(Pref_{\mathcal{N}}\right) \tag{8}$$

However, this 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 to build *directly* the factors of a FCP of $Unf_{\mathcal{N}}$.

To obtain them, it is not enough 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 components that will also provide sufficiently rich behaviours on the interface 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.

A straightforward way to implement this idea is to start with locally complete prefixes, and then to transmit (and receive) the behaviours required on the interface, before recomputing local FCP with these extra requirements, etc. However, this might result in many iterations between the components. 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. $Pref_{\mathcal{A}}$ 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 avoid this situation, the behaviour of a component on its interface is captured 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.

[3] Note that in (8) one has equality if $Pref_{\mathcal{N}}$ is already given in pullback form, as in (6). But in general the minimal pullback covering of $Pref_{\mathcal{N}}$ is larger.



(a) $Pref_{\mathcal{A}}$ (b) interaction (c) $Pref_{\mathcal{B}}$

Figure 4: Illustrating required interaction via the interface between components

4.1 Extended canonical prefixes

The extended canonical prefix of a component $\mathcal{A}$ or $\mathcal{B}$ is built with regard to its interface. 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 *Let $\mathcal{A}$ be a component with an interface C. Then, w.r.t. the interface C, the cutting context $\Theta_C = \left(\approx_{mar}, \lhd_{tot}, \{\boldsymbol{\kappa}_e\}_{e \in E_{\mathcal{A}}}\right)$ is defined with $\forall e \in E_{\mathcal{A}}$,*

$$\boldsymbol{\kappa}_e = \begin{cases} \{\,[e'] \,:\, e' \in E_{\mathcal{A}},\ \Pi_C\,(e') \neq \emptyset\,\} & \textit{if}\ \ \Pi_C\,(e) \neq \emptyset \quad \textit{(a)} \\ \{\,[e'] \,:\, e' \in E_{\mathcal{A}},\ \Pi_C\,([e] \triangle [e']) = \emptyset\,\} & \textit{otherwise} \quad \textit{(b)} \end{cases} \tag{9}$$

where $\triangle$ is the symmetric set difference.

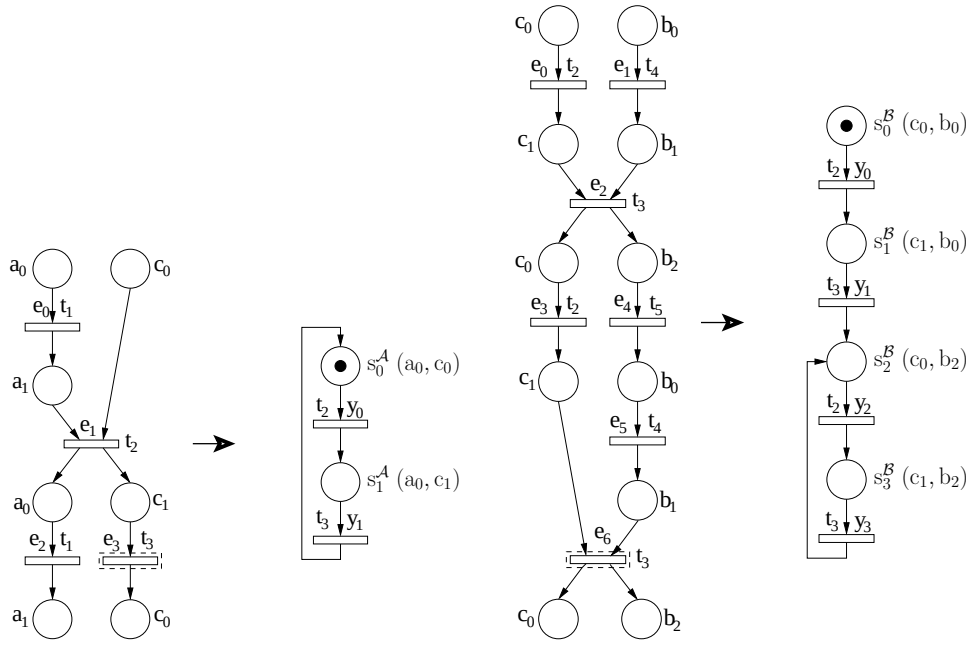
The restriction of the cutting context Θ_C in $\mathcal{A}$ means that:

(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 trantition of the interface) has to be chosen such that there are no interface events "between" e and e', i.e. in $\Pi_C\,([e] \triangle [e'])$. This condition is necessary to cut infinite chains of private events in the unfolding.

Definition 5 *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.*

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. $Pref_{\mathcal{A}}^{\Theta_C}$



(a) $Pref_{\mathcal{A}}^{\Theta_C}$ (b) $\hat{\Pi}_C(\mathcal{A})$ (c) $Pref_{\mathcal{B}}^{\Theta_C}$ (d) $\hat{\Pi}_C(\mathcal{B})$

Figure 5: Extended prefixes and their corresponding summary nets

coincides with the canonical prefix since the only cut-off event and its corresponding event are interface events. 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, which would be obtained by setting e_4 as a cut-off event since $[\bot] \approx_{mar} [e_4]$ and $[\bot] \lhd_{tot} [e_4]$. However, e_4 does not correspond to an extended cut-off event since it is a private event and $\Pi_C([e_4] \triangle [\bot]) = \{e_2, e_3\}$. 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 [10]). It has to be shown that it is finite.

Proposition 1 *$Pref_{\mathcal{A}}^{\Theta_C}$ is finite.*

Proof. By Proposition 2.10 in [10] 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, i.e. events which correspond to C, in the chain then the chain can be cut since the number of markings is finite and thus some marking is a final marking of several interface events.

Otherwise, there is only a finite number of interface events in the chain. Since the chain is infinite it contains an infinite tail of only private (non-interface) events. Since the number of markings is finite some marking is a final marking of several private events in the tail and thus the chain can be cut. □

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.

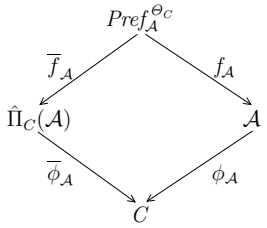


Figure 6: Canonical morphism relating the summary net $\hat{\Pi}_C(\mathcal{A})$ to the interface C.

Definition 6 *Let $Pref_{\mathcal{A}}^{\Theta_C}$ be the extended canonical prefix of $\mathcal{A}$ w.r.t. its interface C and let $cut_{\mathcal{A}}^{\Theta_C}$ be the set of extended cut-off events. Let $\hat{\Pi}_C(\mathcal{A}) = (S, Y, s_0, \rightarrow)$ denote the automaton $loop((Pref_{\mathcal{A}}^{\Theta_C})_{|C})$ with loop defined as*

$$\forall e \in (cut_{\mathcal{A}}^{\Theta_C})_{|C} \text{ merge } e^\bullet \text{ and } e'^\bullet$$

where e' is the corresponding event of e. The nodes of $\hat{\Pi}_C(\mathcal{A})$ correspond to the nodes of $(Pref_{\mathcal{A}}^{\Theta_C})_{|C}$ (up to the merge operation on states S), and s_0 corresponds to the minimal condition of $(Pref_{\mathcal{A}}^{\Theta_C})_{|C}$. Moreover, there exist canonical morphisms $\overline{f}_{\mathcal{A}} : Pref_{\mathcal{A}}^{\Theta_C} \rightarrow \hat{\Pi}_C(\mathcal{A})$ and $\overline{\phi}_{\mathcal{A}} : \hat{\Pi}_C(\mathcal{A}) \rightarrow C$ such that $\overline{\phi_{\mathcal{A}}} \circ \overline{f_{\mathcal{A}}} = \phi_{\mathcal{A}} \circ f_{\mathcal{A}}$ (see Fig. 6). In the sequel, $\hat{\Pi}_C(\mathcal{A})$ is called the summary net *of $\mathcal{A}$ w.r.t. its interface C.*

In addition, to facilitate discussions below, each event e of $\hat{\Pi}_C(\mathcal{A})$ 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 $s = e^\bullet$.

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 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 corresponding event that is also an interface event. Note also that, due to the restriction of interfaces to automata, the summary nets are also automata.

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 labeled by either c_0 or c_1, however, they are associated with different markings in $\mathcal{B}$ (given in brackets next to the states). It can be seen that summary nets carry minimal information about components since their states are linked with the markings reached by interface events.

Observe that the states of the summary net are associated with unique markings since the extended prefix, from which the summary net is derived, is defined using a *total* adequate order. This means, in terms of interface events, that having two interface events with equal final markings implies that one of them is a cut-off event, and thus they are merged by the *loop* operator.

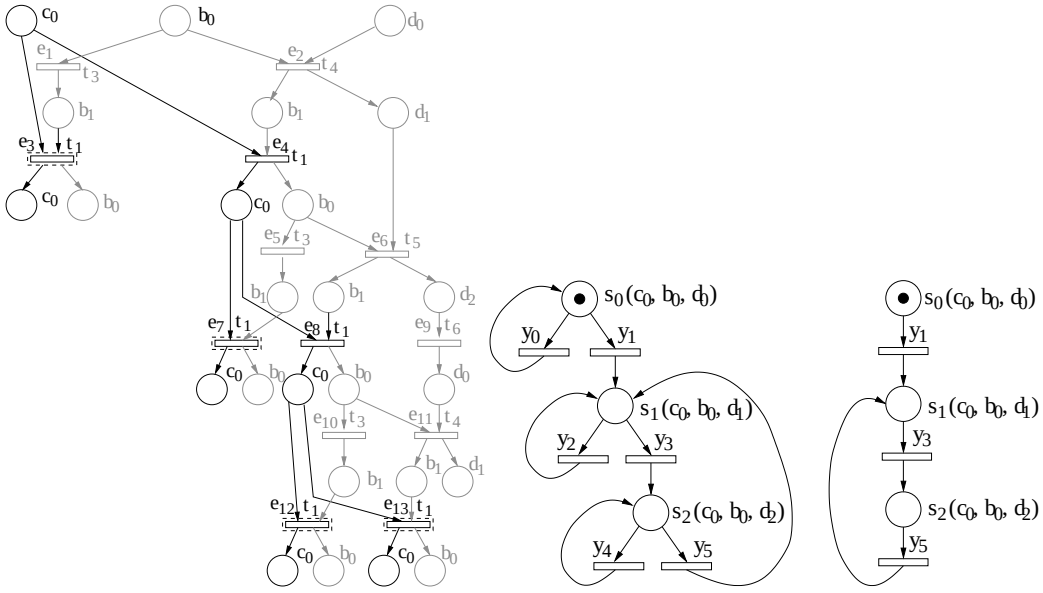
Due to the nature of the summary net the following relation holds.

Proposition 2 $\Pi_C(Unf_{\mathcal{A}}) = \Pi_C\left(Unf_{\hat{\Pi}_C(\mathcal{A})}\right)$.

In other words, the summary net is able to describe all possible behaviours of $\mathcal{A}$ on the interface.

Proof. Let us show $(Unf_{\mathcal{A}})_{|C} = Unf_{\hat{\Pi}_C(\mathcal{A})}$ first.

It is well know that the unfolding of $\mathcal{A}$ can be "refolded" on a canonical prefix $Pref_{\mathcal{A}}^{\Theta_C}$, or equivalently that $Pref_{\mathcal{A}}^{\Theta_C}$ is sufficient to recover the full unfolding $Unf_{\mathcal{A}}$. The idea is that



(a) $Pref_{\mathcal{B}}^{\Theta_C}$ (b) $\hat{\Pi}_C(\mathcal{B})$ (c) trimed $\hat{\Pi}_C(\mathcal{B})$

after any cut-off event e, with corresponding event e', one can "glue" a copy of $(Pref_{\mathcal{A}}^{\Theta_C})_{|e'+}$, the restriction of $Pref_{\mathcal{A}}^{\Theta_C}$ to nodes that are in the future of e', denoted by $e'+$. Doing that repeatedly and for all cut-off events allows to recover the full unfolding. This is a consequence of the completeness property. The same holds for $(Unf_{\mathcal{A}})_{|C}$ and $(Pref_{\mathcal{A}}^{\Theta_C})_{|C}$ because when a cut-off event is an interface event, its corresponding event is also an interface event.

Observe that the completion operation of $(Pref_{\mathcal{A}}^{\Theta_C})_{|C}$, as it was described above, amounts exactly to computing the unfolding of $\hat{\Pi}_C(\mathcal{A})$, by definition of $\hat{\Pi}_C(\mathcal{A})$. This proves that $(Unf_{\mathcal{A}})_{|C}$ is isomorphic to $Unf_{\hat{\Pi}_C(\mathcal{A})}$. So these two occurrence nets have the same projection on C (which simply amounts to trimming these objects, i.e. merging isomorphic configurations). □

Remark. A summary net $\hat{\Pi}_C(\mathcal{A})$ is obtained by looping the restriction instead of the projection of $Pref_{\mathcal{A}}^{\Theta_C}$ to the behaviours of the interface C (no trimming is applied). Recall that the trimming would merge isomorphic configuration of the restriction, and that $\hat{\Pi}_C(\mathcal{A})$ carries minimal information about $\mathcal{A}$ (i.e. the states of $\hat{\Pi}_C(\mathcal{A})$ are associated with markings in $Pref_{\mathcal{A}}^{\Theta_C}$). Thus, the trimming would merge states in the restriction which might be associated with different markings in $Pref_{\mathcal{A}}^{\Theta_C}$. This is illustrated in Figure 4.2 using the component in Figure 4(c). The configurations $[e_{12}]$ and $[e_{13}]$ are isomorphic in the restriction (without considering the greyed out nodes in the figure), and thus, would be merged although they correspond to different marking in the prefix $Mark([e_{12}]) = (c_0, b_0, d_2)$ and $Mark([e_{13}]) = (c_0, b_0, d_1)$. From the summary net in Figure 4.2(b) one can remove the loops containing y_0, y_2 and y_4 resulting in a 'trimed' net shown in 4.2(c). From the point of view of the component $\mathcal{A}$, which receives $\hat{\Pi}_C(\mathcal{B})$ from $\mathcal{B}$, it is necessary to know e.g. y_1 but not y_0 for the construction of the prefix. Note that all transitions in $\hat{\Pi}_C(\mathcal{B})$ correspond to occurrences of t_1. Having found a cut-off event containing y_1 there indirectly exist a cut-off event containing y_0 since the execution of y_0 reaches the same state. Generally, there exist also other cases where isomorphic configurations can be merge. In addition, one could

perform a further trimming by expanding the extended prefix, however, this would mean to trade-off between the width and the length of the resulting summary net.

4.3 Modular canonical prefix construction

Definition 7 *Given a distributed system $\mathcal{N} = \mathcal{A} \times_N^C \mathcal{B}$ the* canonical prefix factor *(CPF) of a component $\mathcal{A}$ is obtained by*

$$\overline{\mathcal{A}} = \mathcal{A} \times_N^C \hat{\Pi}_C(\mathcal{B}) \quad (10)$$

$$\overline{Pref}_{\mathcal{A}}^{\Theta_*} = \Pi_{\mathcal{A}}\left(Pref_{\overline{\mathcal{A}}}^{\Theta_*}\right) \quad (11)$$

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

Observe that in this set-up the selected adequate order is simply the set inclusion $\subset$ [4], 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 3.

Observe also that $\overline{Pref}_{\mathcal{A}}^{\Theta_*}$ is a prefix of $Unf_{\overline{\mathcal{A}}} = Unf_{\mathcal{A}} \times_O^C Unf_{\hat{\Pi}_C(\mathcal{B})}$ and so the projection on $\mathcal{A}$ can be applied to it, which results in a true branching process of $\mathcal{A}$.

For the running example of this paper, the canonical prefix factors of $\mathcal{N}$ and their composition into a prefix of $Unf_{\mathcal{N}}$ are depicted in Figure 7. Observe that the local cut-off decisions are quite conservative and may build branches of a CFP that are not 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 are performed in the definition of the two CFP. As a consequence of this conservative cut-off criterion, by back projecting the composition $\overline{Pref}_{\mathcal{A}}^{\Theta_*} \times_O^C \overline{Pref}_{\mathcal{B}}^{\Theta_*}$ on the two CFP $\overline{Pref}_{\mathcal{A}}^{\Theta_*}$ and $\overline{Pref}_{\mathcal{B}}^{\Theta_*}$ one may get smaller factors. In other words, the two CFP do not correspond to the mimimal covering of their composition.

Lemma 1 *If the summary net $\hat{\Pi}_C(\mathcal{B})$ does not reduce the behaviours of $\mathcal{A}$ (or more precisely of the interface C seen from $\mathcal{A}$), one has $Pref_{\mathcal{A}}^{\Theta_*} \sqsubseteq \overline{Pref}_{\mathcal{A}}^{\Theta_*}$.*

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

Proposition 3

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

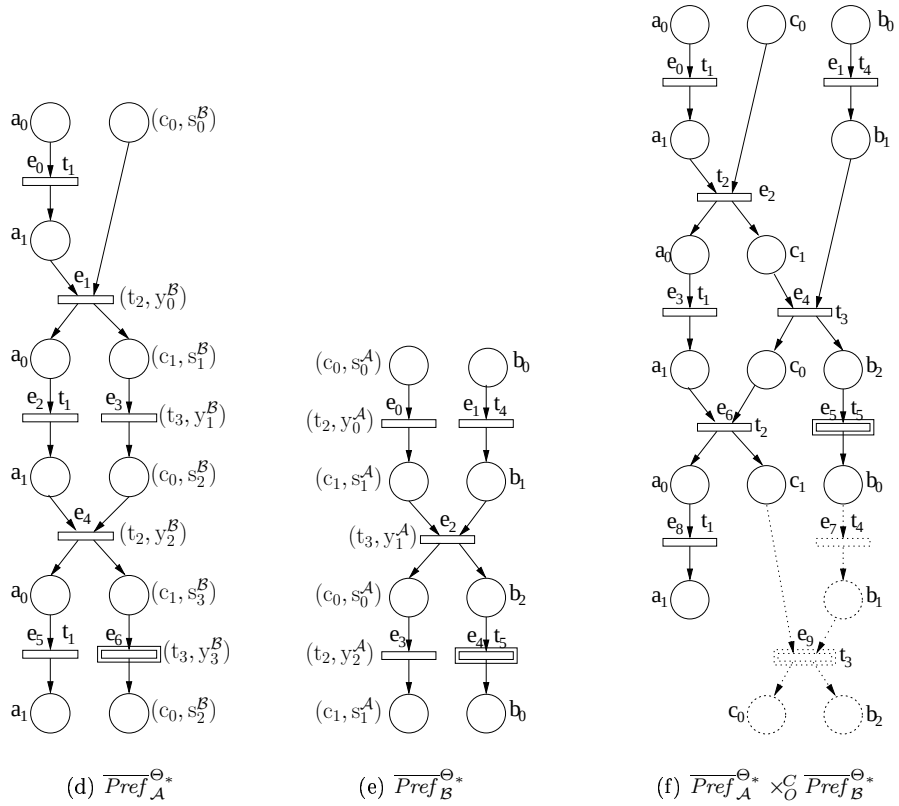
Recalling (8), this result expresses that the two CFP $\overline{Pref}_{\mathcal{A}}^{\Theta_*}$ and $\overline{Pref}_{\mathcal{B}}^{\Theta_*}$ form a pullback covering[5] of a finite complete prefix of $Unf_{\mathcal{N}}$.

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

$$Unf_{\mathcal{A}\times_N^C\hat{\Pi}_C(\mathcal{B})} = Unf_{\mathcal{A}} \times_O^C Unf_{\hat{\Pi}_C(\mathcal{B})} \quad (12)$$

[4] It was pointed out by Walter Vogler that the order $\subset$ is not adequate, and that the prefix build using $\subset$ is a canonical prefix (despite $\subset$ being not adequate) since $\subset$ is included in an adequate order.

[5] not necessarily minimal, as illustrated in the example above



(d) $\overline{Pref}^{\Theta_*}_{\mathcal{A}}$ (e) $\overline{Pref}^{\Theta_*}_{\mathcal{B}}$ (f) $\overline{Pref}^{\Theta_*}_{\mathcal{A}} \times^C_O \overline{Pref}^{\Theta_*}_{\mathcal{B}}$

Figure 7: Canonical prefix factors and their combination by pullback. Notice that event e_6 of factor (d) vanishes in the pullback. To preserve it, one would need the part of f in dashed lines, which is indeed useless to get a complete finite prefix of $Unf_{\mathcal{N}}$.

one has

$$\begin{aligned}
\Pi_{\mathcal{A}}\left(Unf_{\overline{\mathcal{A}}}\right) &= \Pi_{\mathcal{A}}\left(Unf_{\mathcal{A}\times^C_{\hat{N}}\hat{\Pi}_C(\mathcal{B})}\right) \\
&= Unf_{\mathcal{A}} \times^C_O \Pi_C\left(Unf_{\hat{\Pi}_C(\mathcal{B})}\right) \\
&= Unf_{\mathcal{A}} \times^C_O \Pi_C\left(Unf_{\mathcal{B}}\right) \\
&= \Pi_{\mathcal{A}}\left(Unf_{\mathcal{A}} \times^C_O Unf_{\mathcal{B}}\right) \\
&= \Pi_{\mathcal{A}}\left(Unf_{\mathcal{N}}\right) \qquad (13)
\end{aligned}$$

The second equality uses (7), the third one derives from Prop. 2, the fourth equality uses (7) again and the last one corresponds to (3). 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 $\mathcal{A}$.

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

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 (12): 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 $\Pi_{\mathcal{A}}(e) \neq \emptyset$.

Secondly, recall that the events of $Unf_{\hat{\Pi}_C(\mathcal{B})}$ are labeled by the marking of $\mathcal{B}$ they produced, as explained in the construction of summary nets (see the last sentence in Def. 6).

So in the pullback (12), synchronised events are thus associated to 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 (12) that are private to $\mathcal{A}$, provided one relies on the last synchronised event to get the marking produced on the 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. not strictly preceded by a cut-off event of the canonical FCP), and such that $\Pi_{\mathcal{A}}(e) \neq \emptyset$. The configuration $\kappa = [e]$ terminates with marking m of $\mathcal{N}$. By (13) 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 the first remark 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 $\hat{\Pi}_C(\mathcal{B})$, one can choose $\kappa' = [e']$ in such a way that the marking associated to 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 $Unf_{\overline{\mathcal{A}}}$. Otherwise, let $f' \neq e'$ be a cut-off event in κ' and let g' be its corresponding event, both being associated to marking m'. 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 to marking m', 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 that e is a feasible event. In summary, this proves that configuration $\Pi_{\mathcal{A}}([e])$ for the feasible event e of $Unf_{\mathcal{N}}$ is present in $\Pi_{\mathcal{A}}\left(Pref_{\overline{\mathcal{A}}}^{\Theta_*}\right)$, whence the result. □

Notice that the inclusion stated in Proposition 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 $\overline{Pref}_{\mathcal{A}}^{\Theta_*}$ *is finite.*

The proof is similar to the one in Proposition 1.

Proposition 5 $Pref_{\mathcal{N}}^{\Theta_*} \triangleq Pref_{\mathcal{A}\times_N^C\mathcal{B}}^{\Theta_*} \sqsubseteq \overline{Pref}_{\mathcal{A}}^{\Theta_*} \times_O^C \overline{Pref}_{\mathcal{B}}^{\Theta_*}$.

Proof. Immediate from (8) and Proposition 3. □

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, as it was noticed above. And secondly because the factors, computed separately, may not themselves satisfy

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

and symmetrically, as shown by the counter-example in Fig. 7.

4.4 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 $\mathcal{N} = \mathcal{A} \times_N^X \mathcal{B} \times_N^Y \mathcal{C}$ shown in Figure 8. It has three components which only interact through their interfaces between each other. Thus, there is no other interaction between the components $\mathcal{A}$ and $\mathcal{B}$ than the interface X, and similarly, there is no other interaction between the components $\mathcal{B}$ and $\mathcal{C}$

than the interface Y. In the propositions below, it is shown that the canonical prefix factors of this three component system can be computed recursively, with 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 shape.

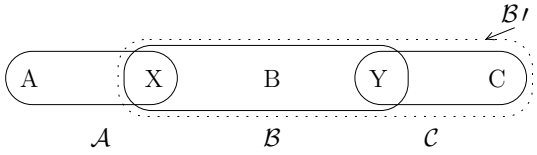


Figure 8: Distributed system $\mathcal{N} = \mathcal{A} \times_N^X \mathcal{B} \times_N^Y \mathcal{C}$.

Proposition 6 (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 canonical prefix factors

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

Then $\overline{Pref}_{\mathcal{A}}^{\Theta_*} = \overline{Pref}_{\mathcal{A}'}^{\Theta_*}$.

In other words, the summary nets can be computed recursively, which provides a way to obtain the canonical prefix factor in $\mathcal{A}$ with small scale computations (it is unnecessary to handle $\mathcal{B}'$).

Proof. One has $Unf_{\mathcal{B}'} = Unf_{\mathcal{B}} \times_O^Y Unf_{\mathcal{C}}$ by (3), whence $\Pi_{\mathcal{B}} (Unf_{\mathcal{B}'}) = Unf_{\mathcal{B}} \times_O^Y \Pi_Y (Unf_{\mathcal{C}})$ using properties of projections on unfoldings[6]. From Proposition 2 one has $\Pi_Y (Unf_{\mathcal{C}}) = \Pi_Y \left(Unf_{\hat{\Pi}_Y(\mathcal{C})} \right)$, so $\Pi_{\mathcal{B}} (Unf_{\mathcal{B}'}) = Unf_{\mathcal{B}} \times_O^Y \Pi_Y \left(Unf_{\hat{\Pi}_Y(\mathcal{C})} \right) = \Pi_{\mathcal{B}} \left(Unf_{\mathcal{B}} \times_O^Y Unf_{\hat{\Pi}_Y(\mathcal{C})} \right) = \Pi_{\mathcal{B}} \left(Unf_{\mathcal{B} \times_N^Y \hat{\Pi}_Y(\mathcal{C})} \right)$. This results in $\Pi_X (Unf_{\mathcal{B}'}) = \Pi_X \left(Unf_{\mathcal{B} \times_N^Y \hat{\Pi}_Y(\mathcal{C})} \right)$. 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 $\Pi_{\mathcal{A}} (Unf_{\overline{\mathcal{A}}}) = \Pi_{\mathcal{A}} (Unf_{\overline{\mathcal{A}'}})$, which entails $Pref_{\overline{\mathcal{A}}}^{\Theta_*} = Pref_{\overline{\mathcal{A}'}}^{\Theta_*}$. □

Proposition 7 (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 $\Pi_{\mathcal{B}} \left(Pref_{\mathcal{N}}^{\Theta_*} \right) \sqsubseteq \overline{Pref}_{\mathcal{B}}^{\Theta_*} = \Pi_{\mathcal{B}} \left(Pref_{\overline{\mathcal{B}}}^{\Theta_*} \right)$.*

So the two lateral summary nets are sufficient to compute the canonical prefix factor in the central component.

Proof. One can show that $\Pi_{\mathcal{B}} (Unf_N) = \Pi_{\mathcal{B}} (Unf_{\overline{\mathcal{B}}})$ from Proposition 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 $\mathcal{B}$.

[6] Actually, this is the propagation rule on unfoldings.

By similar arguments as in Proposition 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 $\Pi_{\mathcal{B}}\left(Pref_N^{\Theta_*}\right)$ is, in general, smaller than $Pref_{\overline{\mathcal{B}}}^{\Theta_*}$. □

The modular construction of canonical prefix factors described above on an elementary string of components extends naturally to nets having a tree shape. Computations are organised as follows. The first thing is to obtain summary nets from all neighbors at each components. Summary nets are first computed starting at the leaves of the structure, and progressing inwards, using the propagation rule. Specifically, 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 neighbors. Then the merge of all incoming summary nets at each component allows to compute the canonical prefix factor 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.

5 Conclusions

This paper has presented a novel approach to address the construction of finite complete prefixes of 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. These second operation can of course benefit from the first one, which reduces the 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. 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.

Although the technique was essentially described for a pair of components, it naturally extends to tree-shaped systems. The next efforts on this theme will focus on the consideration of other adequate orders, and to the derivation of a more appropriate notion of adequate order. The generalization to general nets as interfaces will also be examined.

Acknowledgements

Many thanks to Victor Khomenko and Walter Vogler for interesting discussions.

References

[1] Paolo Baldan, Stefan Haar, and Barbara König. Distributed Unfolding of Petri Nets. In *FoSSaCS*, pages 126–141, 2006.

[2] Jean-Michel Couvreur, Sébastien Grivet, and Denis Poitrenaud. Unfolding of products of symmetrical Petri nets. *Lecture Notes in Computer Science*, 2075:121–143, 2001.

[3] Joost Engelfriet. Branching Processes of Petri Nets. *Acta Informatica*, 28:575–591, 1991. NewsletterInfo: 38, 40.

[4] Javier Esparza and Stefan Romer. An unfolding algorithm for synchronous products of transition systems. In *International Conference on Concurrency Theory*, volume 1664 of *LNCS*, pages 2–20, 1999.

[5] Javier Esparza, Stefan Römer, and Walter Vogler. An Improvement of McMillan's Unfolding Algorithm. In *Formal methods in systems design*, pages 285–310, 2002.

[6] E. Fabre. Factorization of Unfoldings for Distributed Tile Systems, Part 1 : Reduced Interaction Case. Technical Report 1529, IRISA, April 2003.

[7] E. Fabre. Factorization of Unfoldings for Distributed Tile Systems, Part 2 : General Case. Technical Report 1606, IRISA, May 2004.

[8] E. Fabre. On the Construction of Pullbacks for Safe Petri Nets. In *"Applications and Theory of Petri Nets and other Models of Concurrency, ATPN'06, Turku, Finland"*, June 2006.

[9] E. Fabre, A. Benveniste, S. Haar, and C. Jard. Distributed Monitoring of Concurrent and Asynchronous Systems. *Journal of Discrete Event Systems, special issue*, pages 33–84, May 2005.

[10] V Khomenko. *Model Checking Based on Petri Net Unfolding Prefixes.* PhD thesis, University of Newcastle upon Tyne, 2002.

[11] Victor Khomenko, Maciej Koutny, and Walter Vogler. Canonical Prefixes of Petri Net Unfoldings. *Acta Informatica, Volume 40, Number 2*, pages 95–118, October 2003.

[12] K. L. McMillan. *Symbolic Model Checking: an approach to the state explosion problem.* PhD thesis, 1992.

[13] K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Proc. International Workshop on Computer Aided Verification*, pages 164–177, July 1992.

[14] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part I. *Theor. Computer Science*, 13(1):85–108, January 1980.

[15] Glynn Winskel. A New Definition of Morphism on Petri Nets. In *STACS '84: Proceedings of the Symposium of Theoretical Aspects of Computer Science*, pages 140–150, London, UK, 1984. Springer-Verlag.

[16] Glynn Winskel. Categories of models for concurrency. In *Seminar on Concurrency, Carnegie-Mellon University*, pages 246–267, London, UK, 1985. Springer-Verlag.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399