Factorization of Unfoldings for Distributed Tile Systems Part 1 : Reduced Interaction Case

Eric Fabre

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Factorization of Unfoldings for Distributed Tile Systems Part 1 : Reduced Interaction Case

Eric Fabre

N°4829

Mai 2003

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--4829--FR+ENG

Factorization of Unfoldings for Distributed Tile Systems Part 1 : Reduced Interaction Case

Eric Fabre

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Sigma 2

Rapport de recherche n ° 4829 — Mai 2003 — 35 pages

Abstract: We consider products of transition systems, that we represent as tile systems. Tile systems can be viewed as Petri nets : places are replaced by (state) variables, and transitions change the value of part of these variables. A tile system is said to be distributed when it is formed of several components interacting through shared variables. We provide runs of these systems with true concurrency semantics. The unfolding technique, a convenient tool to represent runs with concurrent events, has been shown to apply to such models by Esparza and Römer in [4]. In this paper, we propose an even more compact representation of the system behavior. Specifically, we show that the unfolding of the global system can be expressed as a product of local branching processes, one per component. We also describe a modular procedure to build these local branching processes, based on exchanges of information between interacting components, which avoids any use of global information such as the global unfolding. We believe this result could open a new way to modular model checking techniques, in the spirit of previous work of the authors about modular diagnosis algorithms.

Key-words: discrete event system, modular system, concurrency, unfolding, factorization, turbo algorithm, graphical model of interactions

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Factorisation de dépliages pour des systèmes de pièces distribués Première partie : interactions limitées

Résumé : On considère des produits de systèmes de transitions, représentés sous forme de systèmes de pièces. Ces systèmes de pièces sont très proches des réseaux de Petri : les places sont remplacées par des variables (d'état), et les transitions (les pièces) modifient la valeur d'un sous-ensemble de ces variables. Un système de pièces est dit distribué lorsqu'il est formé de plusieurs composants interagissant à travers leurs variables partagées. On muni les trajectoires de ces systèmes de pièces d'une sémantique de concurrence vraie. La technique de dépliage est alors un outil commode et compact pour représenter l'ensemble des trajectoires. Par rapport au graphe de marquage, le dépliage permet de s'affranchir en partie de l'explosion combinatoire du nombre de trajectoires, due à la concurrence des événements. Dans ce rapport, on propose une représentation encore plus compacte de l'ensemble des trajectoires. Plus précisément, on montre que le dépliage d'un système distribué peut s'exprimer comme le produit des dépliages de ses composants. On propose ensuite un algorithme permettant de calculer les facteurs minimaux dans une telle représentation en produit du dépliage global. Cet algorithme calcule les facteurs de façon modulaire, par échanges d'informations entre composants ; en particulier, le dépliage global n'est jamais calculé. D'autres travaux ont montré que des algorithmes similaires permettaient aussi de résoudre des problèmes de diagnostic distribué (du type programmation dynamique), pour un système dynamique modulaire, sans jamais manipuler l'état global du système. La représentation factorisée de l'ensemble des trajectoires, associée à ces algorithmes modulaires, semble prometteuse pour d'autres problèmes, par exemple la vérification de modèles.

Mots-clé : système à événements discrets, système distribué, concurrence, dépliage, factorisation, algorithme turbo, modèle graphique d'interaction

Contents

1 Introduction

Discrete event dynamic systems with asynchronous interactions have long been challenging to standard model checking techniques. The difficulty comes from the fact that several (concurrent) transitions may be independently enabled and fire in any order, which generates an explosion of the number of possible trajectories, due to interleavings. This drawback was addressed by several authors [1, 2], who proposed the notion of *unfolding* as a more appropriate representation for runs of systems with a high level of concurrency. In these "true concurrency semantics," a run appears as a causality graph (i.e. a partial order) of events, instead of a sequence. In other words, a run becomes a class of sequences of events, modulo the interleaving of concurrent events. This notion has proved to significantly reduce the combinatorial explosion of runs [3], and to be amenable to standard model checking methods [7, 8, 9].

In this paper, we propose to further "compress" the representation of possible runs of a system, still in true concurrency semantics, when this system decomposes as a product of smaller elements. In simple terms, the idea is the following : a run of the global system decomposes as a product of local runs, one in each component. It is very likely that a given local run in some component participates to several runs of the global system. Therefore, there may be few local runs in each component which generate many global runs, which suggests a factorization phenomenon. The purpose of this paper is precisely to evidence this property on the unfolding of a compound system : this unfolding decomposes as a product of local branching processes, one per component, and the collection of these local branching processes may be more compact than the unfolding of the global system itself. A first approach[1] in this direction was proposed by Couvreur *et al* [10] (section 4) : They showed that the unfolding of a compound system relates to unfoldings of its components through partial homomorphisms, although no effective product of branching processes was derived.

This result immediately raises two questions. The first one concerns the computation of the factorized representation of an unfolding, which should be simpler than the computation of the unfolding itself. We provide a modular algorithm to do so, inspired from previous work [12, 14, 15]. This algorithm is based on local computations performed by each component, and on messages exchanged between components ; it only handles local branching processes. The second question concerns the possibility to use the factorized representation of an unfolding as the support of efficient methods to deal with compound systems. Our experience is limited to using unfoldings

[1] The only paper addressing factorization properties of unfoldings, to the knowledge of the author.

for failure diagnosis in distributed asynchronous systems [13], but again the modular asynchronous algorithms mentioned above provide efficient solutions [15]. The possibility to extend them to model checking problems remains an open question, but there is good hope that the answer is positive.

Developements of this paper consider modular systems under the form of tile systems. Tile systems are closely related to safe Petri nets, as we evidence below, and the results we present could be expressed directly on the latter. However, we prefer to emphasize the notion of variable (instead of place) : beyond some technical simplifications, this allows a more direct link with previous work [12], and prepares the way to symbolic unfoldings, where tiles take the form of functions operating on variables.

The paper is organized as follows. Section 2 defines tile systems, their composition, and recalls standard definitions and results about unfoldings. Section 3 defines the product of branching processes and states the factorization property for the unfolding of a compound system. Section 4 studies in detail properties of factors of a branching process, with an emphasis on the notion of minimal factors. Finally, section 5 is devoted to modular algorithms for the computation of minimal factors of an unfolding. This requires to define the projection of a branching process on a component, and to check that projection and product of branching processes satisfy an axiomatic framework previously developed in [12], from which algorithms and convergence results derive naturally. Projections are easy to define for a limited type of interactions between components ; the technical extension to the general case will be addressed in a forthcoming paper.

2 Basic notions

This section defines tile systems, and relates them to Petri nets. Their major interest is to emphasize the notion of local state variable (playing the part of a place in Petri nets). We will rely on these variables to define the notion of local interaction. Tile systems are closely related to the synchronous product of transition systems as defined by Arnold in [11] and revisited by Esparza in [4]. The former can actually be viewed as a "compiled version" of the latter. We then adapt the notions of occurence net and unfolding to tile systems, following [4].

2.1 Tile systems

Systems. A *tile system* $\mathcal{S}$ is a tuple $(\mathcal{V}, \mathcal{T}, \mathbf{v}^0, \alpha, \beta, \gamma)$ where $\mathcal{V}$ and $\mathcal{T}$ are finite sets of variables and tiles respectively, and $\mathbf{v}^0$ is the initial state of the system. Variables are denoted by capital letters $A, B, V_1, V_2, \ldots$, and take values $a, b, v_1, v_2, \ldots$ in the finite domain $\mathcal{D}$. A *state* of $\mathcal{S}$ is a function $\mathbf{v} : \mathcal{V} \to \mathcal{D}$. Tiles are generically denoted by $\mathbf{t}, \mathbf{t}', \mathbf{t}_1, \mathbf{t}_2$, etc. Function $\gamma : \mathcal{T} \to 2^{\mathcal{V}}$ associates to each tile the variable set on which it operates: variables $\gamma(\mathbf{t})$ impacted by tile $\mathbf{t}$ are also denoted by $\mathcal{V}_\mathbf{t}$ for short. In the same way, there exists partial functions $\alpha, \beta : \mathcal{T} \times \mathcal{V} \to \mathcal{D}$ defining for each tile $\mathbf{t}$ and each variable $V \in \mathcal{V}_\mathbf{t}$ the value of V respectively before and after $\mathbf{t}$ fires. $\mathbf{t}$ is enabled by state $\mathbf{v}$, denoted by $\mathbf{v}[\mathbf{t}\rangle$, iff $\forall V \in \mathcal{V}_\mathbf{t}, \mathbf{v}(V) = \alpha(\mathbf{t}, V)$. $\mathbf{t}$ can thus fire, which yields the new state $\mathbf{v}'$ defined by $\forall V \in \mathcal{V}_\mathbf{t}, \mathbf{v}'(V) = \beta(\mathbf{t}, V)$, and $\forall V \in \mathcal{V} \setminus \mathcal{V}_\mathbf{t}, \mathbf{v}'(V) = \mathbf{v}(V)$. We adopt the standard notation $\mathbf{v}[\mathbf{t}\rangle\mathbf{v}'$.

Tile systems are very similar in nature to Petri nets, with places replaced by variables, and transitions by tiles. Apart from the fact that they don't handle tokens, the major structural difference is that the same variables appear in the pre-set and in the post-set of a tile.

Composition. We give two definitions for the composition of tile systems. In the first one, components interact only by sharing variables. This definition is very convenient to keep the interaction span under control, and will be used for the standard definition of compound systems in section 5. In the general definition of composition, systems are allowed to interact also by shared tiles, which captures features of a usual synchronous product.

Let $\mathcal{S}_1 = (\mathcal{V}_1, \mathcal{T}_1, \mathbf{v}_1^0, \alpha_1, \beta_1, \gamma_1)$ and $\mathcal{S}_2 = (\mathcal{V}_2, \mathcal{T}_2, \mathbf{v}_2^0, \alpha_2, \beta_2, \gamma_2)$ be two tile systems, with *distinct* tile sets, we define the composition of $\mathcal{S}_1$ and $\mathcal{S}_2$ by

$$\mathcal{S}_1 \| \mathcal{S}_2 \quad \triangleq \quad (\mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{T}_1 \cup \mathcal{T}_2, \mathbf{v}_1^0 \wedge \mathbf{v}_2^0, \alpha, \beta, \gamma) \tag{1}$$

hence the two components interact through shared variables $\mathcal{V}_1 \cap \mathcal{V}_2$. The initial state $\mathbf{v}^0 = \mathbf{v}_1^0 \wedge \mathbf{v}_2^0$ is given by $\mathbf{v}^0(V) = \mathbf{v}_i^0(V)$ if $V \in \mathcal{V}_i$, $i = 1, 2$, and thus requires that $\mathbf{v}_1^0$ and $\mathbf{v}_2^0$ coincide on $\mathcal{V}_1 \cap \mathcal{V}_2$, which we define as the *coherence assumption*. $\gamma(\mathbf{t})$ is defined by $\mathbf{t} \in \mathcal{T}_i \Rightarrow \gamma(\mathbf{t}) = \gamma_i(\mathbf{t})$, which is unambiguous as far as tile sets are disjoint. Partial functions α and β are defined in the same way. Definition (1) extends to $N \geq 2$ components in a straightforward manner.

We now consider tile systems $\mathcal{S}_i = (\mathcal{V}_i, \mathcal{T}_i, \mathbf{v}_i^0, \alpha_i, \beta_i, \gamma_i)$, $1 \leq i \leq N$, which can both share variables and *tiles*. Composition can be extended to this case provided some extra *coherence* properties are satisfied: for every pair of tile systems $\mathcal{S}_i, \mathcal{S}_j, i \neq j$, we require

1. $\forall \mathbf{t} \in \mathcal{T}_i \cap \mathcal{T}_j,\ \gamma_i(\mathbf{t}) \cap \mathcal{V}_j = \gamma_j(\mathbf{t}) \cap \mathcal{V}_i$
2. $\forall \mathbf{t} \in \mathcal{T}_i \cap \mathcal{T}_j,\ \forall V \in \mathcal{V}_i \cap \mathcal{V}_j,\ \alpha_i(\mathbf{t}, V) = \alpha_j(\mathbf{t}, V)$ and $\beta_i(\mathbf{t}, V) = \beta_j(\mathbf{t}, V)$ on pairs $(\mathbf{t}, V)$ where these functions are defined.

Point 1 expresses that two components must declare the same shared variables for a common tile, and point 2 that on these shared variables, pre- and post-conditions must also be the same. In other words, common tiles only differ by the variables they impact in the components where they appear. So there exists a "global" $\gamma = \cup_i \gamma_i$ such that $\forall 1 \leq i \leq N, \forall \mathbf{t} \in \mathcal{T}_i,\ \gamma_i(\mathbf{t}) = \gamma(\mathbf{t}) \cap \mathcal{V}_i$. Similarly, there exist partial functions α and β on $(\cup_i \mathcal{V}_i) \times (\cup_i \mathcal{T}_i)$, the restrictions of which define the α_i and β_i.

With this coherence property, and the associated definition of α, β, γ, the composition of $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N$ is simply defined as in (1)

$$\mathcal{S}_1 \| \mathcal{S}_2 \| \ldots \| \mathcal{S}_N \quad \triangleq \quad (\cup_i \mathcal{V}_i,\ \cup_i \mathcal{T}_i,\ \wedge_i \mathbf{v}_i^0,\ \alpha,\ \beta,\ \cup_i \gamma_i) \tag{2}$$

The major difference with the previous case appears in the last term $\gamma = \cup_i \gamma_i$: it stresses the fact that if a given tile "name" appears in various tile sets, then the corresponding local definitions of this tile must be glued to form a bigger tile (i.e. involving more variables), as in a standard synchronous product. In the sequel, by abuse of notations, we may drop the index i in α_i, β_i and γ_i when defining and using coherent components. It will be only maintained in γ_i in cases where confusions must be avoided, i.e. when components share tiles.

Coherence of components can be checked recursively, in an associative manner. In the same way, composition $\|$ of coherent tile systems is associative.

Remark : The second definition of composition makes $\|$ similar to a synchronous product : it "glues" transitions together. There is a major difference, however, with a standard synchronous product. Usually, distinct local transition sets are assumed, one per component, and an extra labeling function indicates which local transitions must be glued ; then a name is chosen for the compound global transition. Here, we directly give global names to local transitions in each component, at the expense of larger local systems. We will come back to this aspect at the end of section 3.

2.2 Occurence nets and unfoldings of tile systems

We briefly recall vocabulary and state notations of this paper for standard notions like nets, homomorphisms, etc.

Nets. We define a *net* as a triple $\mathcal{N} = (P, T, \rightarrow)$, where P and T are distinct *place* and *transition* sets, and $\rightarrow \subseteq (P \times T) \cup (P \times T)$ is the *flow relation*. A *labelled net* is a net augmented with a labeling map $\lambda : P \cup T \rightarrow \Lambda$, where Λ is a finite set of labels. Elements of $P \cup T$ are also called *nodes*. The transitive and reflexive closure of the flow relation $\rightarrow$ on nodes is denoted by $\preceq$ ($\prec$ for the irreflexive part). For a node x, its *preset* and *postset* in the net are respectively defined as ${}^\bullet x = \{y : y \rightarrow x\}$ and $x^\bullet = \{y : x \rightarrow y\}$. A net *homomorphism* from $\mathcal{N}$ to $\mathcal{N}'$ is a map $\phi : P \cup T \rightarrow P' \cup T'$ such that 1/ $\phi(P) \subseteq P'$, $\phi(T) \subseteq T'$, and 2/ for every transition $\mathbf{t}$ of $\mathcal{N}$, ϕ restricted to ${}^\bullet\mathbf{t}$ defines a bijection with ${}^\bullet\phi(\mathbf{t})$, and ϕ restricted to $\mathbf{t}^\bullet$ defines a bijection with $\phi(\mathbf{t})^\bullet$. If $\mathcal{N}$ and $\mathcal{N}'$ are labeled nets, ϕ is also required to preserve the labeling.

In a net, two nodes x, x' are said to be *in conflict*, denoted by $x \# x'$, iff there exists two transitions $\mathbf{t}, \mathbf{t}' \in T$ such that $\mathbf{t} \preceq x$, $\mathbf{t}' \preceq x'$ and ${}^\bullet\mathbf{t} \cap {}^\bullet\mathbf{t}' \neq \emptyset$.

Notice that a tile system $\mathcal{S} = (\mathcal{V}, \mathcal{T}, \mathbf{v}^0, \alpha, \beta, \gamma)$ can be easily transformed into an equivalent net by taking $P = \mathcal{V} \times \mathcal{D}$, $T = \mathcal{T}$, and defining $\rightarrow$ by $\forall \mathbf{t} \in \mathcal{T}$, ${}^\bullet\mathbf{t} = \{(V, \alpha(\mathbf{t}, V)), V \in \mathcal{V}_\mathbf{t}\}$ and $\mathbf{t}^\bullet = \{(V, \beta(\mathbf{t}, V)), V \in \mathcal{V}_\mathbf{t}\}$. This allows the use of notations ${}^\bullet\mathbf{t}$ and $\mathbf{t}^\bullet$ for tiles. Equivalence trivially extends to runs of the two models if places $(V, \mathbf{v}^0(V))$ of the net are initially marked. Observe that the equivalent net of $\mathcal{S}$ is safe by construction. Conversely, one easily checks that a safe net can be turned into an equivalent tile system, for example by complementing of places and making a binary variable of each pair (place,complement place), so the two model families have the same expression power.

Occurence nets. A net $\mathcal{O} = (C, E, \rightarrow)$ is said to be an *occurence net* iff it satisfies the following properties

1. $\forall x \in C \cup E, \neg(x \prec x)$: $\preceq$ is a partial order (or $\rightarrow$ defines a directed acyclic graph (DAG) on nodes),
2. $\forall x \in C \cup E$, $|\{y : y \preceq x\}| < \infty$: partial order $\preceq$ is well founded,
3. $\forall c \in C$, $|{}^\bullet c| \leq 1$: each place has at most one input transition,
4. $\forall e \in E$, $|{}^\bullet e| \geq 1$: each transition has at least one input place,
5. $\forall x \in C \cup E, \neg(x \# x)$: no node is in self-conflict

In an occurence net, places (C) are called *conditions*, and transitions (E) are called *events*. Observe that minimal nodes of $\mathcal{O}$ are conditions: $\min(\mathcal{O}) \subseteq C$. The partial order $\preceq$ defines the *causality relation*. Nodes x, x' are said to be *concurrent*, denoted by $x \perp\!\!\!\perp x'$, iff neither $x \# x'$ nor $x \preceq x'$ nor $x' \preceq x$ holds. A *co-set* $X \subseteq C$ is a

set of pairwise concurrent conditions. Maximal co-sets (for set inclusion) are called *cuts*. A *configuration* κ is a sub-net[2] of $\mathcal{O}$ (or more generally a subset of nodes) which is both conflict-free, causally-closed (on the left, i.e. on the "past" side), and satisfying $\forall e \in \kappa \cap E, e^\bullet \subseteq \kappa$.

Branching processes and unfoldings. Occurence nets are useful to represent runs of a tile system in the so-called *true concurrency semantics*. The labeled occurence net $\mathcal{O} = (C, E, \rightarrow, \lambda)$ is a *branching process* of the tile system $\mathcal{S} = (\mathcal{V}, \mathcal{T}, \mathbf{v}^0, \alpha, \beta, \gamma)$ iff

1. λ is a net homomorphism between $(C, E, \rightarrow)$ and the equivalent net of $\mathcal{S}$; in particular, conditions are labeled by pairs (variable,value of that variable), and events are labeled by tiles,
2. λ defines a bijection between $\min(\mathcal{O})$ and $\{(V, \mathbf{v}^0(V)), V \in \mathcal{V}\}$,
3. $\forall e, e' \in E$, ${}^\bullet e = {}^\bullet e'$ and $\lambda(e) = \lambda(e')$ together imply $e = e'$.

Remark : With this definition, a branching process of tile system $\mathcal{S}$ is nothing more than a branching process of its equivalent safe net. However, by contrast with general safe nets, a tile system imposes structure to this net : each variable impacted by a tile appears in the pre-set and in the post-set of each event. In the sequel, we consider restrictions of a branching process to sets of variables. This structure prevents having disconnected events.

A configuration κ of a branching process of $\mathcal{S}$ encodes a run of $\mathcal{S}$ in the following way. Let $e_1, e_2, \ldots$ be *any* linear extension of partial order $\preceq$ reduced to events of κ. Then the sequence of tiles $\lambda(e_1), \lambda(e_2), \ldots$ is firable at the initial state $\mathbf{v}^0$ of $\mathcal{S}$. Moreover, whatever the chosen sequence, the final state of $\mathcal{S}$ is the same. Therefore, configurations of a branching process encode *equivalence classes* of sequences of events. In other words, a sequence of events is "unique" up to the permutation of two consecutive concurrent events. It is precisely this use of concurrency which reduces the set of possible runs of a system and makes unfolding techniques attractive for systems with reduced interactions.

$\mathcal{O}'$ is a *prefix* of $\mathcal{O}$, denoted by $\mathcal{O}' \sqsubseteq \mathcal{O}$, if it is a (left) causally-closed sub-net of $\mathcal{O}$. By convention in this paper, we also require that a prefix contain $\min(\mathcal{O})$, so the prefix of a branching process of $\mathcal{S}$ remains a branching process of $\mathcal{S}$. Two branching

[2] We choose to define configurations as subnets, for a matter of homogeneity. This contrasts with the standard definition which rather considers subsets of events.

processess $\mathcal{O}_1$ and $\mathcal{O}_2$ have a maximal common prefix, defined up to an isomorphism. This common prefix and the corresponding labeled net isomorphism can be defined recursively, starting at $\min(\mathcal{O}_1)$ and $\min(\mathcal{O}_2)$, and checking the presence of equally labeled successor events for every pair of isomorphic co-sets of conditions.

Lemma 1 *Two branching processes $\mathcal{O}_1, \mathcal{O}_2$ of $\mathcal{S}$ are isomorphic iff each configuration of $\mathcal{O}_1$ is isomorphic to a configuration of $\mathcal{O}_2$, and conversely.*

One has to show that the isomorphisms relating pairs of configurations of $\mathcal{O}_1 \times \mathcal{O}_2$ can be "glued" into a larger isomorphism between $\mathcal{O}_1$ and $\mathcal{O}_2$. This is made possible by the fact that two isomorphic configurations of a branching process are necessarily identical, because of point 3 in the definition. Details of the proof are left to the reader. Observe that the result still holds, with the same proof, if, instead of checking all configurations of a branching process, one checks only a covering set of configurations (for example the set of maximal configurations). In the sequel, we do not distinguish isomorphic branching processes, and we prove the identity of branching processes by constructing an isomorphism between them.

The union of branching processes $\mathcal{O}_1$ and $\mathcal{O}_2$ is defined as the minimal occurence net having $\mathcal{O}_1$ and $\mathcal{O}_2$ as prefixes. It can again be defined recursively, with a procedure similar to the search for a common prefix. There exists a unique branching process having all branching processes as prefixes; it is called the *unfolding* of the tile system $\mathcal{S}$ (see theorem 23 in [2]).

A branching process $(C, E, \rightarrow, \lambda)$ of $\mathcal{S}$ can be constructed by the following

Procedure 1

- initialization :
  - $E = \emptyset$
  - create $|\mathcal{V}|$ initial conditions in C such that $\forall V \in \mathcal{V}, \exists! c \in C : \lambda(c) = (V, \mathbf{v}^0(V))$
- recursion :
  - choose co-set $X \subseteq C$ and tile $\mathbf{t} \in \mathcal{T}$ such that $\lambda(X) = {}^\bullet\mathbf{t}$,
  - create e in E such that ${}^\bullet e = X$ and $\lambda(e) = \mathbf{t}$ (if there is no such event),
  - then $\forall V \in \mathcal{V}_\mathbf{t}$, create a c in C such that ${}^\bullet c = e$ and $\lambda(c) = (V, \beta(\mathbf{t}, V))$

The unfolding of $\mathcal{S}$, denoted by $\mathcal{U}_\mathcal{S}$, is the unique supremum of all finite branching processes obtained by procedure 1, and is thus the unique stable point of procedure 1.

3 Unfolding factorization

This section defines the synchronous product of branching processes, and states a factorization result on the unfolding of a compound system.

3.1 Extended components

Given a tile system $\mathcal{S} = (\mathcal{V}, \mathcal{T}, \mathbf{v}^0, \alpha, \beta, \gamma)$ and a subset $\mathcal{V}' \subseteq \mathcal{V}$ of variables, we define the *restriction of $\mathcal{S}$ to variables $\mathcal{V}'$* as

$$\mathcal{S}_{|\mathcal{V}'} \triangleq (\mathcal{V}', \mathcal{T}', \mathbf{v}^0_{|\mathcal{V}'}, \alpha_{|\mathcal{T}'\times\mathcal{V}'}, \beta_{|\mathcal{T}'\times\mathcal{V}'}, \gamma')$$

where $\mathcal{T}' = \{\mathbf{t} \in \mathcal{T} : \mathcal{V}_\mathbf{t} \cap \mathcal{T}' \neq \emptyset\}$ and $\forall t \in \mathcal{T}'$, $\gamma'(t) = \gamma(t) \cap \mathcal{V}'$. In other words, $\mathcal{S}'$ keeps the tiles of $\mathcal{S}$ *truncated* to variables $\mathcal{V}'$. For simplicity, we write $\mathcal{S}_{|\mathcal{V}'} = (\mathcal{V}', \mathcal{T}', \mathbf{v}^0_{|\mathcal{V}'}, \alpha, \beta, \gamma')$.

Lemma 2 *For a given $\mathcal{S}$ with variable set $\mathcal{V}$, let $\mathcal{V}_i \subseteq \mathcal{V}$ for $1 \leq i \leq N$ and define $\mathcal{S}_i^e = \mathcal{S}_{|\mathcal{V}_i}, 1 \leq i \leq N$. Then the $\mathcal{S}_i^e$ are coherent and $\mathcal{S}_1^e \| \ldots \| \mathcal{S}_N^e = \mathcal{S}_{|\cup_i \mathcal{V}_i}$.*

The proof is straightforward and left to the reader. Assume the $\mathcal{V}_i$ cover $\mathcal{V}$, i.e. $\mathcal{V} = \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_N$. In general, there exist several representations of $\mathcal{S}$ as $\mathcal{S}_1 \| \ldots \| \mathcal{S}_N$ where sub-systems $\mathcal{S}_i$ operate on variable sets $\mathcal{V}_i$. The $\mathcal{S}_i^e$ give a canonical one, where each component knows "everything that could happen on its variables in $\mathcal{S}$." In particular, for every decomposition $\mathcal{S} = \mathcal{S}_1 \| \ldots \| \mathcal{S}_N$, $\mathcal{S}_i^e$ contains all tiles of $\mathcal{S}_i$ and thus appears as an extension of it. Figure 1 gives an example of compatible components $\mathcal{S}_1, \mathcal{S}_2$ and their extended versions, under the form of equivalent (safe) nets, with $\mathcal{V}_1 = \{A, C\}$ and $\mathcal{V}_2 = \{B_1, B_2, C\}$. Initial states/markings $\mathbf{v}_1^0$ and $\mathbf{v}_2^0$ appear in bold. Dashed arrows represent the truncated tiles of the other component.

In terms of unfoldings, observe that $\mathcal{U}_{\mathcal{S}_i^e}$ describes more configurations than $\mathcal{U}_{\mathcal{S}_i}$, because of the extra transitions borrowed to other components. But it also describes more behaviors than one could find in $\mathcal{U}_\mathcal{S}$ restricted[3] to $\mathcal{S}_i^e$, since extra transitions are "free of use" in $\mathcal{S}_i^e$, whereas in $\mathcal{S}$ they are constrained by private variables of other components.

3.2 Product of branching processes

Let $\mathcal{S} = (\mathcal{V}, \mathcal{T}, \mathbf{v}^0, \alpha, \beta, \gamma)$ be a tile system, select $\mathcal{V}_i \subseteq \mathcal{V}$ for $1 \leq i \leq N$, and define components $\mathcal{S}_i^e = \mathcal{S}_{|\mathcal{V}_i} = (\mathcal{V}_i, \mathcal{T}_i^e, \mathbf{v}_i^0, \alpha, \beta, \gamma_i)$. Let $\mathcal{O}_i = (C_i, E_i, \rightarrow_i, \lambda_i)$ be a

[3] we specify this operation below

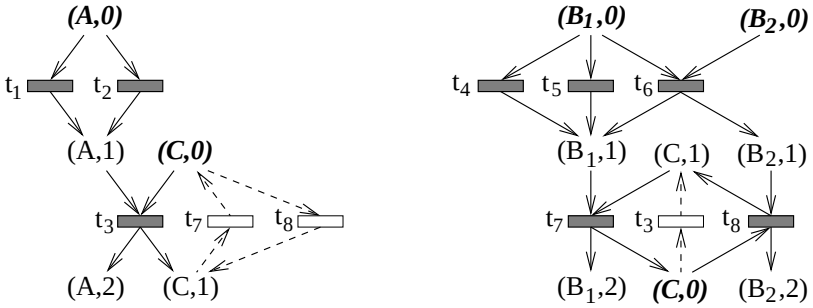

Figure 1: *Two components (solid arrows) and their extended versions (solid+dashed arrows).*

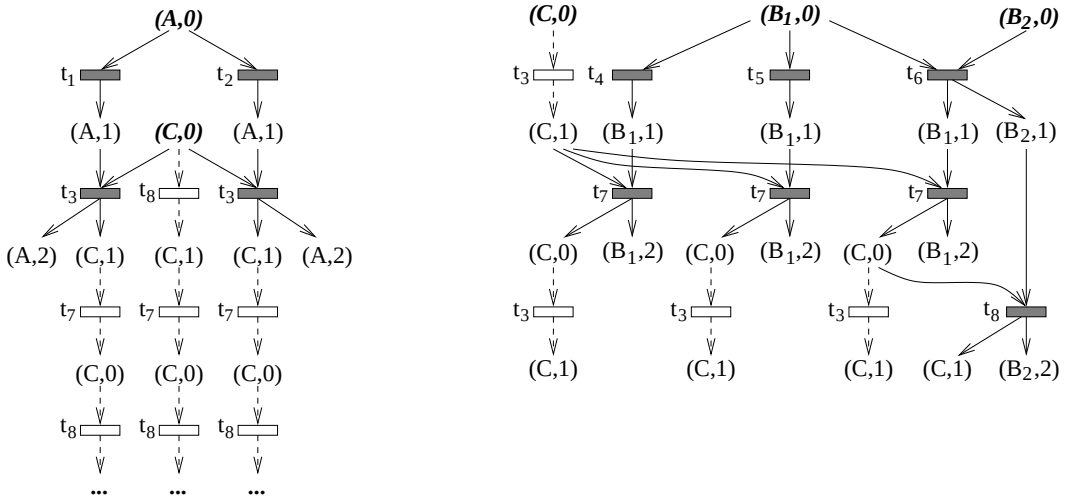

Figure 2: *Unfoldings of the extended components of fig. 1.*

branching process of $\mathcal{S}_i^e$. The product $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$ is defined as the labeled occurence net $(C, E, \rightarrow, \lambda)$ satisfying

1. events are labeled by tiles of $\cup_i \mathcal{T}_i^e$, and conditions by pairs of $(\cup_i \mathcal{V}_i) \times \mathcal{D}$,
2. $\forall i \in \{1, \ldots, N\}$, there exists a partial homomorphism ψ_i between $\mathcal{O}$ and $\mathcal{O}_i$: $\psi_i : \lambda^{-1}(\mathcal{V}_i \times \mathcal{D}) \cup \lambda^{-1}(\mathcal{T}_i^e) \rightarrow C_i \cup E_i$
3. each restriction[4] $\psi_i : \min(\mathcal{O}) \rightarrow \min(\mathcal{O}_i)$ is bijective,
4. $\forall X$ co-set of C, $\forall \mathbf{t} \in (\cup_i \mathcal{T}_i^e)$ and $I = \{i : 1 \leq i \leq N, \mathbf{t} \in \mathcal{T}_i^e\}$, if $\forall i \in I$, $\exists e_i \in E_i$ such that $\lambda_i(e_i) = \mathbf{t}$ and $\psi_i(X) = {}^\bullet\mathbf{e}_i$, then $\exists! e \in E : {}^\bullet e = X$, $\lambda(e) = \mathbf{t}$, and this event e satisfies $\forall i \in I$, $\psi_i(e) = e_i$ and restriction $\psi_i : e^\bullet \rightarrow e_i^\bullet$ is bijective

A crucial point in this definition is that a single node of $\mathcal{O}$ may be labeled by several ψ_i, and thus related to several nodes in the $\mathcal{O}_i$. This is how local events of the $\mathcal{O}_i$ are glued together, and how local conditions are merged. Observe that points 1, 2, 3 ensure that configurations appearing in $\mathcal{O}$ have a legal image (by ψ_i) in every $\mathcal{O}_i$, whereas point 4 guarantees the "maximality" of $\mathcal{O}$. Procedure 2 below rephrases this definition of $\wedge$ in a constructive manner ; it emphasizes in particular that $\wedge$ basically performs a synchronous product of the $\mathcal{O}_i$, under the constraint that the resulting labeled net remains a DAG.

Procedure 2

- initialization :
  - $E = \emptyset$
  - create $|\cup_i \mathcal{V}_i|$ initial conditions in C, with labels satisfying $\lambda(C) = \cup_i \lambda_i(\min(\mathcal{O}_i))$
  - for $1 \leq i \leq N$, define the partial maps $\psi_i : C \rightarrow \min(\mathcal{O}_i)$ so that they are surjective and preserve λ
- recursion :
  for X a co-set of C, $\mathbf{t} \in (\cup_i \mathcal{T}_i)$, $I = \{i : 1 \leq i \leq N, \mathbf{t} \in \mathcal{T}_i^e\}$ and $\{e_i : i \in I, e_i \in E_i\}$ a set of events labeled by tile $\mathbf{t}$, satisfying $\forall i \in I$, $\psi_i(X) = {}^\bullet e_i$,
  - create e in E such that ${}^\bullet e = X$ and $\lambda(e) = \mathbf{t}$ (if there is no such event in E), and define $\psi_i(e) = e_i$ for $i \in I$,

[4] When talking about the restriction of a partial function ψ_i to a set S, we consider only its domain of definition in that set, i.e. $S \cap \lambda^{-1}(\mathcal{T}_i^e \cup (\mathcal{V}_i \times \mathcal{D}))$.

– then create $|\cup_i \gamma_i(\mathbf{t})|$ new conditions c in C, with ${}^\bullet c = e$, and assign labels to have $\lambda(e^\bullet) = \cup_{i\in I}\lambda_i(e_i^\bullet)$; extend partial maps ψ_i, $i \in I$, so that $\psi_i : e^\bullet \to e_i^\bullet$ is λ preserving and surjective.

This procedure guarantees that points 1, 2 and 3 are satisfied at any time, and extensions aim at satisfying the maximality of point 4. Anticipating on lemma 3, procedure 2 builds finite branching processes of $\|_i \mathcal{S}_i^e$; their union still satisfies points 1, 2 and 3, and improve the satisfaction of point 4, so $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$ appears as the supremum of all of them, and is thus the unique stable point of procedure 2. This proves existence and uniqueness of the object described by points 1 to 4.

Lemma 3 *Let $\mathcal{S}$ be a tile system with variable set $\mathcal{V}$. Let $\mathcal{V}_i \subseteq \mathcal{V}$ for $1 \leq i \leq N$, and define (coherent) components $\mathcal{S}_i^e = \mathcal{S}_{|\mathcal{V}_i}$. Let $\mathcal{O}_i$ be a branching process of $\mathcal{S}_i^e$, $1 \leq i \leq N$. Then $\mathcal{O} = \wedge_i \mathcal{O}_i$ is a branching process of $\|_i \mathcal{S}_i^e$.*

The proof is straightforward, by recursion in procedure 2.

Lemma 4 *Let $\mathcal{O}_1, \mathcal{O}_2$ be branching processes of the* same *system $\mathcal{S}$, then $\mathcal{O}_1 \wedge \mathcal{O}_2$ is the largest common prefix of $\mathcal{O}_1$ and $\mathcal{O}_2$. Similarly, for $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{S}' = \mathcal{S}_{|\mathcal{V}'}$, let $\mathcal{O}$ be a branching process of $\mathcal{S}$, then $\mathcal{O} \wedge \mathcal{U}_{\mathcal{S}'} = \mathcal{O}$.*

The proof is immediate and left to the reader (hint : check points 1 to 4 are satisfied by the largest common prefix of $\mathcal{O}_1$ and $\mathcal{O}_2$).

Lemma 5 *Consider a system $\mathcal{S}$, the product $\wedge$ defined on branching processes of restrictions of $\mathcal{S}$ is associative.*

Proof. Since commutativity is built in the definition of $\wedge$, we only have to show $(\mathcal{O}_1 \wedge \mathcal{O}_2) \wedge \mathcal{O}_3 = \mathcal{O}_1 \wedge \mathcal{O}_2 \wedge \mathcal{O}_3$. Observe that $\mathcal{O}_1 \wedge \mathcal{O}_2$ is a branching process of $\mathcal{S}_1^e \| \mathcal{S}_2^e = \mathcal{S}_{|\mathcal{V}_1 \cup \mathcal{V}_2}$ by lemmas 3 and 2, so $(\mathcal{O}_1 \wedge \mathcal{O}_2) \wedge \mathcal{O}_3$ is well defined.

Let $\mathcal{O} = (C, E, \to, \lambda) = (\mathcal{O}_1 \wedge \mathcal{O}_2) \wedge \mathcal{O}_3$ and $\mathcal{O}' = (C', E', \to', \lambda') = \mathcal{O}_1 \wedge \mathcal{O}_2 \wedge \mathcal{O}_3$, we have to prove the existence of an isomorphism ϕ between $\mathcal{O}$ and $\mathcal{O}'$. By construction of $\mathcal{O}'$, conditions and events are related to elements of $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ by partial functions $\psi_1', \psi_2', \psi_3'$ respectively. In the same way, let ψ_{12} and ψ_3 be the partial functions relating elements of $\mathcal{O}$ to those of $\mathcal{O}_1 \wedge \mathcal{O}_2$ and $\mathcal{O}_3$. Since elements of $\mathcal{O}_1 \wedge \mathcal{O}_2$ are in turn labeled by elements of $\mathcal{O}_1$ and $\mathcal{O}_2$ through ψ_1 and ψ_2 respectively, we can directly write that ψ_1, ψ_2, ψ_3 label elements of $\mathcal{O}$. Observe that, by construction, elements of $\mathcal{O}$ (resp. $\mathcal{O}'$) having the same collection of ψ-labels (resp. ψ'-labels) are necessarily identical.

Let us define $\phi : C \cup E \to C' \cup E'$, associating to a node x the unique node x' (if it exists) such that the ψ labels are preserved, i.e. $\forall i, \psi_i(x) = \psi'_i(x')$. ϕ is a partial function mapping events to events and conditions to conditions. ϕ is also injective on its definition set. We have to show that it is defined at all nodes, and is a (one to one) homomorphism from $\mathcal{O}$ to $\mathcal{O}'$. We proceed by recursion, assuming procedure 2 is building $\mathcal{O}$ from $\mathcal{O}_1 \wedge \mathcal{O}_2$ and $\mathcal{O}_3$, and drives in parallel the construction of part of $\mathcal{O}'$. ϕ is obviously a homomorphism between $\min(\mathcal{O})$ and $\min(\mathcal{O}')$ at initialization of procedure 2. Let X be a co-set of C, and $X' = \phi(X)$ its image co-set in C'. Let $\mathbf{t}$ be a tile of $\cup_i \mathcal{T}_i^e$, lying in all $\mathcal{T}_i^e$ for example, and such that $\lambda(X) = {}^\bullet\mathbf{t} = \lambda'(X')$. Assume there exists events $e_{12} \in \mathcal{O}_1 \wedge \mathcal{O}_2$ and $e_3 \in \mathcal{O}_3$, labeled by $\mathbf{t}$ and such that ${}^\bullet e_{12} = \psi_{12}(X)$, ${}^\bullet e_3 = \psi_3(X)$. By definition of $\mathcal{O}_1 \wedge \mathcal{O}_2$ there exists $e_1 = \psi_1(e_{12}) \in E_1$ and $e_2 = \psi_2(e_{12}) \in E_2$, both labeled by $\mathbf{t}$, and such that $\psi_1(X) \triangleq \psi_1(\psi_{12}(X)) = {}^\bullet e_1$, $\psi_2(X) \triangleq \psi_2(\psi_{12}(X)) = {}^\bullet e_2$. By the recursion assumption, we also have $\psi'_i(X') = {}^\bullet e_i$ for $i = 1, 2, 3$. In procedure 2, e_{12} and e_3 can be merged into event e of $(\mathcal{O}_1 \wedge \mathcal{O}_2) \wedge \mathcal{O}_3$ with ${}^\bullet e = X$. In the same way, e_1, e_2, e_3 can be merged into event e' of $\mathcal{O}_1 \wedge \mathcal{O}_2 \wedge \mathcal{O}_3$ with ${}^\bullet e' = X'$. So we have $\phi(e) = e'$: the injective morphism ϕ extends to the new event e. In the same way, one can establish a bijective correspondance between $e^\bullet$ and $e'^\bullet$. A similar reasoning applies to all other shapes of tiles.

By a symmetrical reasoning, there exists an injective homomorphism ϕ' from $\mathcal{O}'$ to $\mathcal{O}$ preserving the ψ-labels. It is obtained recursively, with procedure 2 building $\mathcal{O}'$ from $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and driving the simultaneous construction of part of $\mathcal{O}_1 \wedge \mathcal{O}_2$ and of part of $\mathcal{O}$.

We have proved the existence of injective homomorphisms ϕ from $\mathcal{O}$ to $\mathcal{O}'$, and ϕ' from $\mathcal{O}'$ to $\mathcal{O}$. Since they are both ψ-preserving, they are easily seen to be inverses one of the other. This is more than enough to see that ϕ is actually bijective, and thus is an isomorphism. □

As a subproduct of this proof, we have that the partial homomorphism ψ_{12} from $\mathcal{O}$ to $\mathcal{O}_1 \wedge \mathcal{O}_2$ can be regarded as the pair (ψ_1, ψ_2) relating elements of $\mathcal{O}'$ to $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively. This property is detailed in section 4.

3.3 Unfolding factorization

Theorem 1 *Let the $\mathcal{S}_i^e$ be restrictions of a given tile system $\mathcal{S}$, $1 \leq i \leq N$. Then $\mathcal{U}_{\mathcal{S}_1^e \| \ldots \| \mathcal{S}_N^e}$ is isomorphic to $\mathcal{U}_{\mathcal{S}_1^e} \wedge \ldots \wedge \mathcal{U}_{\mathcal{S}_N^e}$.*

Proof. It is enough to prove the result for $N = 2$; it extends to $N > 2$ by associativity of $\|$ and $\wedge$.

By lemma 3, $\mathcal{U}_{\mathcal{S}_1^e} \wedge \mathcal{U}_{\mathcal{S}_2^e}$ is a branching process of $\mathcal{S}_1^e \| \mathcal{S}_2^e$, so there exists an injective homomorphism $\phi : \mathcal{U}_{\mathcal{S}_1^e} \wedge \mathcal{U}_{\mathcal{S}_2^e} \to \mathcal{U}_{\mathcal{S}_1^e \| \mathcal{S}_2^e}$. We only have to show that ϕ is surjective.

Assume ϕ is not surjective. Let κ' be a finite configuration of $\mathcal{U}_{\mathcal{S}_1^e \| \mathcal{S}_2^e}$ containing a node without inverse image by ϕ. Let $\kappa = \phi^{-1}(\kappa')$, then κ is a configuration of $\mathcal{U}_{\mathcal{S}_1^e} \wedge \mathcal{U}_{\mathcal{S}_2^e}$ and $\phi(\kappa)$ is a prefix of κ', using the fact that ϕ is an injective homomorphism. Let $Y = \max(\kappa)$ and $Y' = \max(\phi(\kappa)) = \phi(Y)$ (recall that Y and Y' are cuts, i.e. maximal co-sets of conditions). If $Y' = \max(\kappa')$, then $\phi(\kappa) = \kappa'$, whence a contradiction. So there exists an event $e' \in \kappa' \setminus \phi(\kappa)$ such that ${}^\bullet e' = X' \subseteq Y'$. Let $\mathbf{t} = \lambda(e')$, and $X = \phi^{-1}(X')$. X is a co-set of conditions in $\mathcal{U}_{\mathcal{S}_1^e} \wedge \mathcal{U}_{\mathcal{S}_2^e}$, and $\lambda(X) = \lambda(X')$. For $i = 1, 2$, if $\mathbf{t} \in \mathcal{T}_i^e$, then $\lambda(\psi_i(X)) = {}^\bullet(\mathbf{t})$ in $\mathcal{S}_i^e$, so, by definition of unfoldings, there exists an event e_i in $\mathcal{U}_{\mathcal{S}_i^e}$ such that $\lambda_i(e_i) = \mathbf{t}$ and ${}^\bullet e_i = \psi_i(X)$. As a consequence, by definition of $\wedge$, and whatever the shape of $\mathbf{t}$, there exists an event e in $\mathcal{U}_{\mathcal{S}_1^e} \wedge \mathcal{U}_{\mathcal{S}_2^e}$ with ${}^\bullet e = X$ and $\lambda(e) = \mathbf{t}$. It is easily seen that the homomorphism ϕ must satisfy $\phi(e) = e'$, which contradicts the fact that e' has no inverse image by ϕ. □

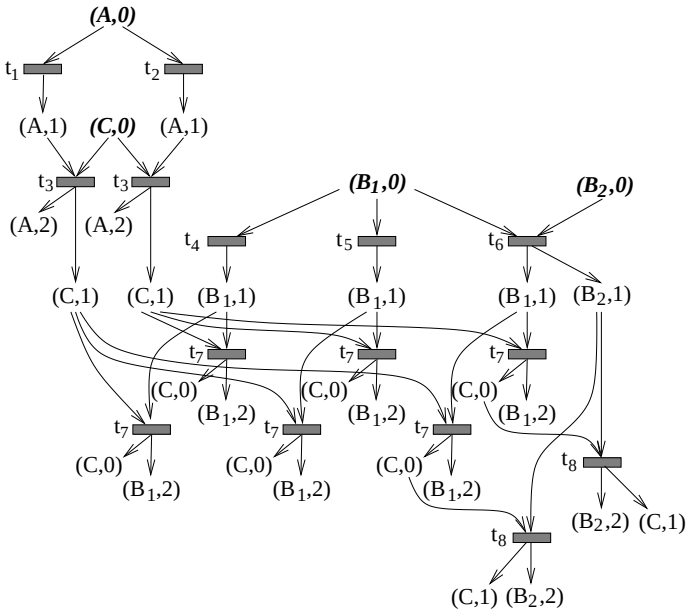

Figure 3: *Unfolding of* $\mathcal{S} = \mathcal{S}_1 \| \mathcal{S}_2$.

The unfolding of our running example $\mathcal{S} = \mathcal{S}_1 \| \mathcal{S}_2$ is depicted on fig. 3. It can be checked that $\mathcal{U}_\mathcal{S}$ is obtained as $\mathcal{U}_{\mathcal{S}_1^e} \wedge \mathcal{U}_{\mathcal{S}_2^e}$. Observe that some configurations of $\mathcal{U}_{\mathcal{S}_1^e}$ are shortened to a prefix (in particular, the infinite sequences changing the value of variable C). The interest of the factorized representation is already suggested by this example: there are $n = 2$ ways to produce condition $(C, 1)$ in $\mathcal{S}_1$, and $m = 3$ ways to use it in $\mathcal{S}_2$. This corresponds to n and m maximal configurations in $\mathcal{U}_{\mathcal{S}_1^e}$ and $\mathcal{U}_{\mathcal{S}_2^e}$ respectively (we put apart the local maximal configuration of $\mathcal{U}_{\mathcal{S}_1^e}$ containing only changes on the value of C by $\mathbf{t}_8$ and $\mathbf{t}_7$). However, in $\mathcal{U}_S$, their combinations generate $n \times m$ maximal configurations, whence a greater complexity. One could naturally object that the difference in complexity may not be favorable to the factorized representation, since the $\mathcal{U}_{\mathcal{S}_i^e}$ exhibit behaviors that are not possible in $\mathcal{S}$. Reducing local factors to behaviors appearing in $\mathcal{S}$ is precisely the objective of the next section.

Before, it is worth relating results of this section to previous work on the topic. The idea of partial homomorphisms relating the unfolding of a compound system to unfoldings of its components was already presented in [10], although no product of branching processes was defined there. However, a crucial difference lies in the definition of the compound system itself: $\mathcal{S}$ is chosen to be a standard synchronous product of its components (driven by transition labels), by contrast with (1) which is a simple union. In other words, components handle local transition names in [10], and global names in the present paper. As already mentioned (end of section 2.1), our approach corresponds to a "compiled version" of the synchronous product, which yields larger components, and thus larger local unfoldings. Anticipating on the next section, consider a configuration κ of $\mathcal{U}_\mathcal{S}$; its images by partial homomorphisms ψ_i yield configurations κ_i of components $\mathcal{S}_i$ in both cases. In the present paper, $\wedge_i \kappa_i$ yields κ without ambiguity, whereas in [10], local configurations κ_i may very well produce several other configurations by synchronous product. Or, in a better formulation, there may be several configurations κ, κ', etc. in $\mathcal{U}_\mathcal{S}$ with the same set of images through the ψ_i. It is precisely to avoid this phenomenon that the authors introduce the "non-reentrance" assumption on $\mathcal{S}$.

4 Minimal factors

In this section, without loss of generality, we assume $\mathcal{S}_1^e \| \ldots \| \mathcal{S}_N^e$ yields $\mathcal{S}$, or equivalently that $\cup_i \mathcal{V}_i = \mathcal{V}$; $\mathcal{O}_i$ denotes a generic branching process of $\mathcal{S}_i^e$. We examine properties related to the definition of $\wedge$, and in particular to the partial homomorphisms ψ_i. We show that they select in each $\mathcal{O}_i$ the minimal part allowing to recover

the same product $\mathcal{O}$. In terms of unfoldings, this defines the minimal parts of the $\mathcal{U}_{\mathcal{S}_i^e}$ necessary to get $\mathcal{U}_{\mathcal{S}}$, or, in other words, local runs of $\mathcal{S}_i^e$ that truly participate to a run of the global system $\mathcal{S}$. Section 5 will address the problem of computing these minimal factors without computing $\mathcal{U}_{\mathcal{S}}$ itself.

4.1 Monotonicity of ψ_i

Lemma 6 *Let $\bar{\mathcal{O}}_i$ be a prefix of $\mathcal{O}_i$, $1 \leq i \leq N$. Define $\bar{\mathcal{O}} = \wedge_i \bar{\mathcal{O}}_i$, with the $\bar{\psi}_i$ as associated partial homomorphisms, and $\mathcal{O} = \wedge_i \mathcal{O}_i$ associated to the ψ_i. Then $\bar{\mathcal{O}} \sqsubseteq \mathcal{O}$, and ψ_i coincides with $\bar{\psi}_i$ on $\bar{\mathcal{O}}$.*

The proof is straightforward, by recursion in procedure 2 building simultaneously $\mathcal{O}$ and $\bar{\mathcal{O}}$. This result allows a consistant use of notation ψ_i on any $\mathcal{O}$, a branching process of $\mathcal{S}$, viewed as a prefix of $\mathcal{U}_{\mathcal{S}} = \wedge_i \mathcal{U}_{\mathcal{S}_i^e}$. The reverse property holds also :

Lemma 7 *Let $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$, and $\bar{\mathcal{O}}$ be a prefix of $\mathcal{O}$. Then $\bar{\mathcal{O}}_i = \psi_i(\bar{\mathcal{O}})$ is a prefix of $\mathcal{O}_i$. If $\bar{\mathcal{O}}$ reduces to a single configuration κ, then $\kappa_i = \psi_i(\kappa)$ is a configuration of $\mathcal{O}_i$.*

Proof. By definition of a prefix, $\bar{\mathcal{O}}$ contains $\min(\mathcal{O})$, so by point 3 in the definition of $\wedge$, $\bar{\mathcal{O}}_i$ contains $\min(\mathcal{O}_i)$. We have to show that $\bar{\mathcal{O}}_i$ is causally-closed for $\rightarrow_i$ in $\mathcal{O}_i$. Let x be a node of $\bar{\mathcal{O}}$, labeled by an element of $\mathcal{S}_i^e$, and let $x_i = \psi_i(x)$ be its image node in $\bar{\mathcal{O}}_i$. For every $x_i' \in {}^\bullet x_i$, we must show the existence of $x' \in {}^\bullet x$ in $\mathcal{O}$ such that $\psi_i(x') = x_i'$ (remark : if x' exists in $\mathcal{O}$, then it is necessarily a node of $\bar{\mathcal{O}}$, by the prefix assumption).

Consider first event nodes : $x = e \in E$, $x_i = \psi_i(e) = e_i \in E_i$ and pick condition $x_i' = c_i \in {}^\bullet e_i$. Let $X = {}^\bullet e$ in $\mathcal{O}$; by point 4 in the definition of $\wedge$, the restriction of $\psi_i : X \rightarrow {}^\bullet e_i$ is bijective, so there exists $c \in {}^\bullet e$ such that $\psi_i(c) = c_i$.

Consider now condition nodes : $x = c \in C$, $x_i = \psi_i(c) = c_i \in C_i$, and let $e_i = {}^\bullet c_i$ be the unique event preceeding c_i in $\mathcal{O}_i$, and similarly $e = {}^\bullet c$ in $\mathcal{O}$. Event e is necessarily labeled by a tile of $\mathcal{T}_i^e$, as a predecessor event of a condition labeled by an element of $\mathcal{V}_i \times \mathcal{D}$. So ψ_i is defined at e : let $e_i' = \psi_i(e)$. Using the homomorphism property, we have $\psi_i(e^\bullet) = \psi_i(e)^\bullet = e_i'^\bullet$. Since $c_i \in \psi_i(e^\bullet)$, we have $c_i \in e_i'^\bullet$, and thus $e_i' = e_i$ because c_i has a unique predecessor. Hence $\psi_i(e) = e_i$.

For the last part of the lemma, we already know that κ_i is a prefix of $\mathcal{O}_i$. We also have that the restriction $\psi_i : \kappa \rightarrow \kappa_i$ is bijective[5] (on its definition domain), and is thus an isomorphism. Assume two nodes are in conflict in κ_i, then their

[5] This can be easily seen in procedure 2 used to build the single configuration κ.

inverse images by ψ_i are also in conflict in κ, which contradicts the fact that κ is a configuration. □

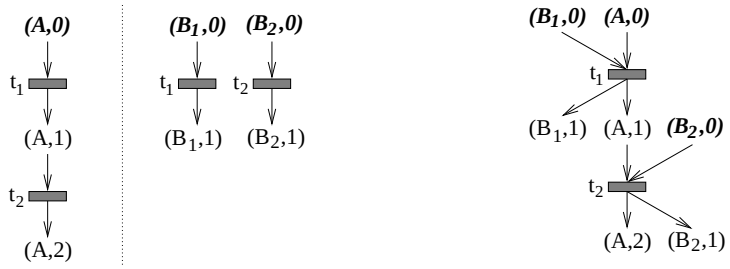

Figure 4: *Two components $\mathcal{S}_1^e, \mathcal{S}_2^e$ (left), sharing only tiles. These components are isomorphic to their unfoldings. The product of local unfoldings is represented on the right : $\mathcal{U}_{\mathcal{S}_1^e \| \mathcal{S}_2^e} = \mathcal{U}_{\mathcal{S}_1^e} \wedge \mathcal{U}_{\mathcal{S}_2^e}$. On this example, $\psi_i(\mathcal{U}_{\mathcal{S}_1^e \| \mathcal{S}_2^e}) = \mathcal{U}_{\mathcal{S}_i^e}$. Events associated to $\mathbf{t}_1$ and $\mathbf{t}_2$ are causally related in $\mathcal{U}_{\mathcal{S}_1^e \| \mathcal{S}_2^e}$, but this relation is lost by ψ_2.*

Lemma 7 actually reveals that causality relations $\rightarrow_i$ in configurations of $\mathcal{O}_i$ are still present in $\mathcal{O}$ (and may be reinforced by relations $\rightarrow_j$ from other components) ; these relations are recovered by the homomorphism ψ_i. This property will be emphasized in proposition 3. Observe that two events (or generally nodes) of $\mathcal{O}$ may be causally related, but have concurrent images in $\psi_i(\mathcal{O})$ (see fig. 4). In the same way, two events (or nodes) may be in conflict in $\mathcal{O}$, but appear as concurrent in $\psi_i(O)$ (see fig. 5). Roughly speaking, ψ_i filters out causalities or conflicts due to external variables/tiles of $\mathcal{S}_i^e$, i.e. not directly due to $\rightarrow_i$.

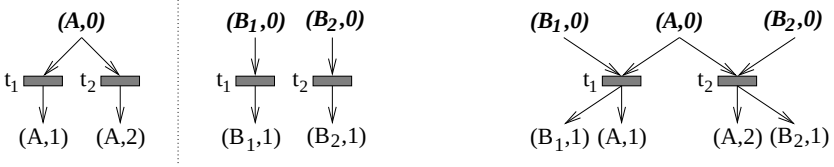

Figure 5: *Same situation as figure 4, but now a conflict relation is lost by ψ_2.*

4.2 Minimal factors of a product

Proposition 1 *Let $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$, and let $\mathcal{O}'_i = \psi_i(\mathcal{O})$, $1 \leq i \leq N$. Then $\mathcal{O} = \mathcal{O}'_1 \wedge \ldots \wedge \mathcal{O}'_N$.*

Proof. One already has that $\mathcal{O}' = \wedge_i \mathcal{O}'_i$ is a prefix of $\mathcal{O}$, as a product of prefixes of the $\mathcal{O}_i$ (see lemmas 7 and 6). Each node of $\mathcal{O}_i \setminus \mathcal{O}'_i$ (as well as all its successors) is actually useless to compute $\wedge_j \mathcal{O}_j$ since it appears in no node of $\mathcal{O}$. So it can be removed from $\mathcal{O}_i$ (as well as its successors) without changing the result of procedure 2. So $\mathcal{O}' = \mathcal{O}$. □

Notice also that every node of $\mathcal{O}'_i$ is useful to at least one element of $\mathcal{O}$, so the $\psi_i(\mathcal{O})$ define the "minimal factorization" of $\mathcal{O}$ as a product of branching processes of the $\mathcal{S}^e_i$.

Definition 1 *Assume $\mathcal{S} = \mathcal{S}^e_1 \| \ldots \| \mathcal{S}^e_N$, and thus $\mathcal{U}_\mathcal{S} = \wedge_i \mathcal{U}_{\mathcal{S}^e_i}$ by theorem 1. The minimal factors of $\mathcal{U}_\mathcal{S}$ in components $\mathcal{S}^e_i$ are the prefixes $\psi_i(\mathcal{U}_\mathcal{S})$ of $\mathcal{U}_{\mathcal{S}^e_i}$.*

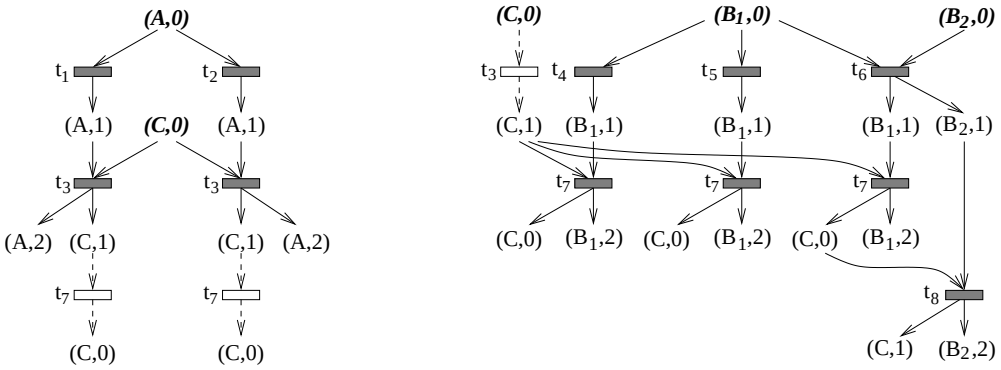

Figure 6: *Minimal factors of $\mathcal{U}_\mathcal{S}$ given on fig. 3. The infinite sequences of $\mathbf{t}_7$ and $\mathbf{t}_8$ on fig. 2 have disappeared. It is now clear that there are 2 maximal configurations on one side, and 3 on the other, for a combination of 6 on the unfolding of $\mathcal{S}$.*

On our running example, minimal factors of $\mathcal{U}_\mathcal{S}$ are given by fig. 6.

4.3 Minimal product covering

Not all branching processes of $\|_i \mathcal{S}^e_i$ have a product representation, but there exists a smaller product form containing a given branching process $\bar{\mathcal{O}}$ of $\mathcal{S}$. In effect, assume $\bar{\mathcal{O}} \sqsubseteq \mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$ and $\bar{\mathcal{O}} \sqsubseteq \mathcal{O}' = \mathcal{O}'_1 \wedge \ldots \wedge \mathcal{O}'_N$, then $\bar{\mathcal{O}} \sqsubseteq \mathcal{O} \wedge \mathcal{O}'$ which is the common prefix of $\mathcal{O}$ and $\mathcal{O}'$, and thus $\bar{\mathcal{O}} \sqsubseteq (\mathcal{O}_1 \wedge \mathcal{O}'_1) \wedge \ldots \wedge (\mathcal{O}_N \wedge \mathcal{O}'_N)$, where again factors $\mathcal{O}_i \wedge \mathcal{O}'_i$ are common prefixes of $\mathcal{O}_i$ and $\mathcal{O}'_i$.

Proposition 2 *Let $\bar{\mathcal{O}}$ be a branching process of $\mathcal{S}$, define*[6] *$\bar{\mathcal{O}}_i = \psi_i(\bar{\mathcal{O}}), 1 \leq i \leq N$, and build $\mathcal{O} = \bar{\mathcal{O}}_1 \wedge \ldots \wedge \bar{\mathcal{O}}_N$. Then $\bar{\mathcal{O}} \sqsubseteq \mathcal{O}$, and $\mathcal{O}$ is the smallest product branching process of $\mathcal{S}$ satisfying this property. In general, $\bar{\mathcal{O}}$ is a strict prefix of $\mathcal{O}$. If $\bar{\mathcal{O}}$ reduces to a single configuration $\bar{\kappa}$, equality holds.*

This result can be viewed as a refinement of proposition 1, where $\bar{\mathcal{O}}$ was limited to a product form.

Proof. Observe first that $\bar{\mathcal{O}}_i$ is a branching process of $\mathcal{S}_i^e$ by lemma 7 (take $\mathcal{O}_i = \mathcal{U}_{\mathcal{S}_i^e}$), so $\mathcal{O}$ is well defined. $\bar{\mathcal{O}}_i$ contains nodes of $\mathcal{U}_{\mathcal{S}_i^e}$ which are useful to at least one node of $\bar{\mathcal{O}}$ in procedure 2, so $\bar{\mathcal{O}} \sqsubseteq \mathcal{O}$. This argument is also sufficient to prove that $\mathcal{O}$ is the smallest product form containing $\bar{\mathcal{O}}$. But, for minimality, one can further observe that if a product form $\mathcal{O}' = \mathcal{O}'_1 \wedge \ldots \wedge \mathcal{O}'_N$ satisfies $\bar{\mathcal{O}} \sqsubseteq \mathcal{O}'$, then $\bar{\mathcal{O}}_i = \psi_i(\bar{\mathcal{O}}) \sqsubseteq \psi_i(\mathcal{O}') \sqsubseteq \mathcal{O}'_i$ by lemma 7, so $\mathcal{O} = \wedge_i \bar{\mathcal{O}}_i \sqsubseteq \mathcal{O}'$ by lemma 6.

To prove $\bar{\mathcal{O}} \neq \mathcal{O}$ in general, a counter-example is enough. Consider again our running example (fig. 3) : there are two events labeled by $\mathbf{t}_3$. Take for $\bar{\mathcal{O}}$ the union of all configurations containing a given $\mathbf{t}_3$, plus *one* maximal configuration containing the second $\mathbf{t}_3$. $\bar{\mathcal{O}}$ is a strict prefix of $\mathcal{U}_{\mathcal{S}}$, but $\psi_i(\bar{\mathcal{O}})$ yields $\psi_i(\mathcal{U}_{\mathcal{S}})$, the minimal factors of $\mathcal{U}_{\mathcal{S}}$, so $\mathcal{O} = \mathcal{U}_S$.

Particular case $\bar{\mathcal{O}} = \bar{\kappa}$. Referring to lemma 7, $\bar{\kappa}_i = \psi_i(\bar{\kappa})$ is a configuration, and $\bar{\kappa}$ is a prefix of $\bar{\kappa}' = \wedge_i \bar{\kappa}_i$, obviously another configuration. Using the fact that restrictions of the ψ_i are bijective, $\bar{\kappa}$ is easily seen to be equal to $\bar{\kappa}'$. □

The minimal product covering is *not* stable by $\wedge$, i.e. for $\bar{\mathcal{O}}$ and $\bar{\mathcal{O}}'$ with respective product coverings $\bar{\mathcal{O}}_1 \wedge \ldots \wedge \bar{\mathcal{O}}_N$ and $\bar{\mathcal{O}}'_1 \wedge \ldots \wedge \bar{\mathcal{O}}'_N$, one has $\psi_i(\bar{\mathcal{O}} \wedge \bar{\mathcal{O}}') \sqsubset \bar{\mathcal{O}}_i \wedge \bar{\mathcal{O}}'_i$ in general. To illustrate this, we turn back again to our running example. Let $\mathbf{t}_3^1, \mathbf{t}_3^2$ denote the two events labeled by $\mathbf{t}_3$, take for $\bar{\mathcal{O}}$ all configurations containing $\mathbf{t}_3^1$ and one maximal configuration containing $\mathbf{t}_3^2$, and symmetrically for $\bar{\mathcal{O}}'$. Then $\bar{\mathcal{O}} \wedge \bar{\mathcal{O}}'$ is formed of a pair of maximal configurations, one containing $\mathbf{t}_3^1$ and the other $\mathbf{t}_3^2$, so $\psi_i(\bar{\mathcal{O}} \wedge \bar{\mathcal{O}}')$ is strictly smaller than $\bar{\mathcal{O}}_i \wedge \bar{\mathcal{O}}'_i = \psi_i(\mathcal{U}_{\mathcal{S}})$.

Lemma 8 *Let $\bar{\mathcal{O}} \sqsubseteq \mathcal{O}$ be branching processes of $\mathcal{S}$, then $\psi_i(\bar{\mathcal{O}}) \sqsubseteq \psi_i(\mathcal{O})$.*

This result refines lemma 7 by droping the product form assumption for $\mathcal{O}$.

[6] Recall that ψ_i is defined on $\mathcal{O}$ as a prefix of $\mathcal{U}_{\mathcal{S}} = \wedge_i \mathcal{U}_{\mathcal{S}^e_i}$.

Proof. Define $\mathcal{O}_i = \psi_i(\mathcal{O})$ and build $\mathcal{O}' = \wedge_i \mathcal{O}_i$, the smallest product form containing $\mathcal{O}$. Then $\bar{\mathcal{O}} \sqsubseteq \mathcal{O} \sqsubseteq \mathcal{O}'$ and lemma 7 applies : $\psi_i(\bar{\mathcal{O}})$ is a prefix of $\mathcal{O}_i = \psi_i(\mathcal{O})$. □

Remark : One could imagine that factors $\psi_i(\mathcal{O})$ could be obtained by restricting $\mathcal{O}$ to nodes of $\lambda^{-1}(\mathcal{T}_i^e \cup (\mathcal{V}_i \times \mathcal{D}))$. But the net obtained in that way fails to be a branching process of $\mathcal{S}_i^e$: point 3 in the definition is violated (the other points hold). In other words, to get a branching process, one would have to merge copies of the same event (i.e. events associated to the same tile, and following the same co-set). This is studied in details in section 5, where these factors are computed without an explicit link to the product form $\mathcal{U}_\mathcal{S} = \wedge_i \mathcal{U}_{\mathcal{S}_i^e}$ (needed to define the ψ_i).

4.4 Decomposition of a ψ_I

Let I be a subset of $\{1, \ldots, N\}$. By lemma 5, we know that $\wedge$ is associative, so $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N = (\wedge_{i\in I}\mathcal{O}_i) \wedge (\wedge_{i\notin I}\mathcal{O}_i)$ and there exists a partial homomorphism $\psi_I : \mathcal{O} \to \mathcal{O}_I$ where $\mathcal{O}_I = \wedge_{i\in I}\mathcal{O}_i$. As a consequence, the lemmas of the previous sections apply to ψ_I. For example, by lemma 8, if $\bar{\mathcal{O}} \sqsubseteq \mathcal{O}$ then $\psi_I(\bar{\mathcal{O}})$ is a prefix of $\mathcal{O}_I$. In the same way, for κ a configuration of $\mathcal{O}$, $\kappa_I = \psi_I(\kappa)$ is a configuration of $\mathcal{O}_I$ by lemma 7. Proposition 1 generalizes in the following way : let $I_1 \cup \ldots \cup I_L$ be a partition of $\{1, \ldots, N\}$, then $\mathcal{O} = \wedge_{l=1}^L \psi_{I_l}(\mathcal{O})$. Similarly, a configuration κ of $\mathcal{O}$ decomposes as $\wedge_{l=1}^L \kappa_{I_l}$ with $\kappa_{I_l} = \psi_{I_l}(\kappa)$.

The existence of ψ_I is of little practical use if, to determine $\psi_I(\mathcal{O})$, one has first to compute $\mathcal{O}_I$, and then to recompute $\mathcal{O}$ from $\mathcal{O}_I$. We show below that ψ_I can actually be obtained directly as a combination of the $\psi_i, i \in I$. We first define an intermediate object : the cartesian product of branching processes $\mathcal{O}_i$.

Let $\mathcal{O}_i = (C_i, E_i, \to_i, \lambda_i)$ be a branching process of $\mathcal{S}_i^e$, we define the label preserving *cartesian product* $\mathcal{O}_1 \dot{\times} \mathcal{O}_2 = (C_1 \dot{\times} C_2, E_1 \dot{\times} E_2, \to, \lambda)$ in the following way.

$$\begin{aligned} C_1 \dot{\times} C_2 \;\; = \;\; & \{(c_1, c_2) \,:\, c_1 \in C_1,\, c_2 \in C_2,\, \lambda_1(c_1) = \lambda_2(c_2)\} \\ & \cup \{(c_1, \epsilon) \,:\, c_1 \in C_1\} \cup \{(\epsilon, c_2) \,:\, c_2 \in C_2\} \end{aligned}$$

where ϵ is a special value. $E_1 \dot{\times} E_2$ is defined accordingly, and $\lambda(x_1, x_2)$ is the common label of non-ϵ nodes in the tuple. Finally, $\to$ is defined by $(x_1, x_2) \to (x_1', x_2')$ iff $x_1 \to_1 x_1'$ or $x_2 \to_2 x_2'$. Observe that $\mathcal{O}_1 \dot{\times} \mathcal{O}_2$ is not a DAG anymore. This product extends to N components and is associative.

Proposition 3 *Let $\mathcal{O} = (\wedge_{i\in I}\mathcal{O}_i) \wedge (\wedge_{i\notin I}\mathcal{O}_i)$ for $I \subseteq \{1, \ldots, N\}$, and $\psi_I(\mathcal{O})$ be its image in $\mathcal{O}_I = \wedge_{i\in I}\mathcal{O}_i$ by the partial homomorphism ψ_I. Then $\psi_I(\mathcal{O})$ is isomorphic to $(\psi_i, i \in I)(\mathcal{O})$, where the ψ_i are the partial homomorphisms between $\mathcal{O}$ and the $\mathcal{O}_i$ for $\mathcal{O}$ computed as $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$.*

Proof. Let the ψ'_i be the partial homomorphisms relating $\mathcal{O}_I = \wedge_{j\in I}\mathcal{O}_j$ to the factors $\mathcal{O}_i$. Given $\dot{\times}_{i\in I}\mathcal{O}_i$, one can define the map

$$\begin{array}{rccc}(\psi'_i, i\in I): & \mathcal{O}_I & \to & \dot{\times}_{i\in I}\mathcal{O}_i \\ & x & \mapsto & (\psi'_i(x), i\in I)\end{array}$$

with the convention that $\psi'_i(x) = \epsilon$ if ψ'_i is not defined at that node in $\mathcal{O}_I$. The map $(\psi'_i, i\in I)$ is defined everywhere. Moreover, $(\psi'_i, i\in I)$ is an injective homomorphism of labeled nets, and thus establishes an isomorphism between $\mathcal{O}_I$ and its image. Consequently, $(\psi'_i, i\in I)$ also defines an isomorphism between $\psi_I(\mathcal{O})$, a prefix of $\mathcal{O}_I$, and its image $(\psi'_i, i\in I)(\psi_I(\mathcal{O}))$ in $\dot{\times}_{i\in I}\mathcal{O}_i$.

Consider now $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$ and the associated homomorphisms ψ_i relating $\mathcal{O}$ to its factors $\mathcal{O}_i$. Again, $(\psi_i, i\in I)$ defines a partial[7] homomorphism between $\mathcal{O}$ and $\dot{\times}_{i\in I}\mathcal{O}_i$. Since $(\psi_i, i\in I)(\mathcal{O})$ coincides with $(\psi'_i, i\in I)(\psi_I(\mathcal{O}))$, we thus have that $(\psi_i, i\in I)(\mathcal{O})$ is actually isomorphic to $\psi_I(\mathcal{O})$. □

This result underlines that mapping ψ_I preserves only the causality relations due to the $\mathcal{S}^e_i$, $i\in I$, and erases causalities or conflicts due to other components in $\mathcal{O}$. We can go further in this direction.

In the proof of the previous lemma, we have seen that the ψ'_i relating $\mathcal{O}_I$ to the $\mathcal{O}_i$ coincide with the ψ_i on $\psi_I(\mathcal{O}) \sqsubseteq \mathcal{O}_I$ (observe that the former have a larger definition domain in general). Let $J\subseteq I$, then $\psi_J(\mathcal{O})$ is isomorphic to $(\psi_j, j\in J)(\mathcal{O})$, which we denote by $\psi_J(\mathcal{O}) \simeq (\psi_j, j\in J)(\mathcal{O})$. Let us decompose $\mathcal{O}$ as $\mathcal{O} = (\mathcal{O}_J \wedge \mathcal{O}_{I\setminus J}) \wedge (\wedge_{i\notin I}\mathcal{O}_i)$, then $\psi_I(\mathcal{O}) \simeq (\psi_i, i\in I)(\mathcal{O})$. Moreover, let ψ'_J be the mapping from $\mathcal{O}_I = \mathcal{O}_J \wedge \mathcal{O}_{I\setminus J}$ to $\mathcal{O}_J$, then

$$\begin{array}{rcl}\psi'_J(\psi_I(\mathcal{O})) & \simeq & (\psi'_j, j\in J)(\psi_I(\mathcal{O})) \\ & \simeq & (\psi_j, j\in J)(\psi_I(\mathcal{O})) \\ & \simeq & (\psi_j, j\in J)\circ(\psi_i, i\in I)(\mathcal{O}) \\ & \simeq & (\psi_j, j\in J)(\mathcal{O}) \\ & = & \psi_J(\mathcal{O})\end{array}$$

which evidences a chain rule.

Corollary 1 *Let $\mathcal{O} = \mathcal{O}_1 \wedge \ldots \wedge \mathcal{O}_N$ and $J\subseteq I\subseteq\{1,\ldots,N\}$. Then $\mathcal{O} \simeq (\psi_i, 1\le i\le N)(O)$, and $\psi_I(\mathcal{O})$ is obtained by cutting off dimensions not indexed by I in this cartesian product representation. As a consequence, one has the composition law $\psi_J\circ\psi_I(\mathcal{O}) = \psi_J(\mathcal{O})$.*

[7] By convention, $(\psi_i, i\in I)$ is defined on nodes of $\mathcal{O}$ on which at least one of the $\psi_i, i\in I$ is defined.

The chain rule can be extended to any pair I, J: for $J \not\subseteq I$ define $\psi_J(\mathcal{O}_I)$ as $\psi_{J\cap I}(\mathcal{O}_I)$. This yields $\psi_I \circ \psi_J(\mathcal{O}) = \psi_J \circ \psi_I(\mathcal{O}) = \psi_{I\cap J}(\mathcal{O})$, which holds for any branching process of $\mathcal{S}$ (not only product ones) since it holds on $\mathcal{U}_\mathcal{S}$.

5 Modular computation of minimal factors for tree-shaped systems

We have already mentioned some advantages of the product representation for the unfolding of a compound system: there may be a substantial gain in size / complexity, and minimal factors may also be a sufficient information, since they describe all possible behaviors of a given component once the latter is inserted in the global system. However, this compact representation is of little use if minimal factors can only be obtained as the image by some homomorphism of the global system unfolding. In this section, we address the problem of deriving minimal factors of a compound system without computing the global unfolding. We start by defining the projection of a branching process of $\mathcal{S}$ to a restriction $\mathcal{S}' = \mathcal{S}_{|\mathcal{V}'}$. In this paper, projections are tightly related to homomorphisms ψ of the previous section, although they are defined without reference to a product. We examine these relations in detail, and study in particular the interplay between projections and product $\wedge$ on branching processes. This will evidence the key result of this section: to determine $\psi_1(\mathcal{U}_{\mathcal{S}_1^e \| \mathcal{S}_2^e})$ one only needs to take the product of $\mathcal{U}_{\mathcal{S}_1^e}$ with the projection of $\mathcal{U}_{\mathcal{S}_2^e}$ on the *interface* $\mathcal{S}^e_{1;2} = \mathcal{S}_{|\mathcal{V}_1\cap\mathcal{V}_2}$ between the two components. This opens the way to modular processings of minimal factors.

5.1 Projection of a branching process

Let $\mathcal{O} = (C, E, \rightarrow, \lambda)$ be a branching process of the tile system $\mathcal{S} = (\mathcal{V}, \mathcal{T}, \mathbf{v}^0, \alpha, \beta, \gamma)$. For $\mathcal{V}' \subseteq \mathcal{V}$, define $\mathcal{S}' = \mathcal{S}_{|\mathcal{V}'} = (\mathcal{V}', \mathcal{T}', \mathbf{v}^0_{|\mathcal{V}'}, \alpha, \beta, \gamma')$, and consider the restriction of $\mathcal{O}$ to nodes labeled by elements of $\mathcal{S}'$, which we denote by $\mathcal{O}_{|\mathcal{S}'}$ or simply $\mathcal{O}_{|\mathcal{V}'}$ by abuse of notations[8]. $\mathcal{O}_{|\mathcal{V}'}$ is a labeled occurence net, which satisfies

Lemma 9 *Every configuration of $\mathcal{O}_{|\mathcal{V}'}$ is isomorphic to a configuration of $\mathcal{U}_{\mathcal{S}'}$.*

[8] This makes sense however: $\mathcal{O}_{|\mathcal{V}'}$ keeps conditions labeled by variables $\mathcal{V}'$ and all events connected to these nodes.

Proof. Every node of $\mathcal{O}$ labeled by $\mathcal{S}'$ is either minimal in $\mathcal{O}$ or has another node labeled by $\mathcal{S}'$ as predecessor, due to the structure of tiles which have the same variables appearing in their pre-set and in their post-set. So $\min(\mathcal{O}_{|\mathcal{V}'}) \subseteq \min(\mathcal{O})$.

Let κ be a configuration of $\mathcal{O}_{|\mathcal{V}'}$: we show that κ is a branching process of $\mathcal{S}'$. Obviously, $\min(\kappa) = \min(\mathcal{O}) \cap \lambda^{-1}(\mathcal{V}' \times \mathcal{D})$ is a set of conditions which corresponds to the initial state $\mathbf{v}^0_{|\mathcal{V}'}$, so point 2 in the definition of a branching process is satisfied. In $\mathcal{O}$, λ defines a net homomorphism between $\mathcal{O}$ and the equivalent net of $\mathcal{S}$. Restricting $\mathcal{O}$ to nodes labeled by $\mathcal{S}'$ obviously preserves this homomorphism property : λ is a homomorphism between $\mathcal{O}_{|\mathcal{V}'}$ and the equivalent net of $\mathcal{S}'$, so point 1 is statisfied. Finally, since κ is conflict-free, point 3 is straightforward : ${}^\bullet e = {}^\bullet e'$ in κ necessarily means $e = e'$ (otherwise e and e' would be in conflict). □

Despite lemma 9, $\mathcal{O}_{|\mathcal{V}'}$ itself is not a branching process of $\mathcal{S}'$ since it precisely fails at point 3 of the definition. Consider for example the unfolding of fig. 3 as $\mathcal{O}$, and take its restriction to nodes labeled by $\mathcal{S}^e_1 = \mathcal{S}_{|\mathcal{V}_1}$ for $\mathcal{O}_{|\mathcal{V}'}$. In $\mathcal{O}_{|\mathcal{V}'}$, each condition labeled by $(C, 1)$ has three events labeled by $\mathbf{t}_7$ as successors. Nevertheless, one can enforce point 3 in the following way :

Definition 2 *Relying on lemma 9, let $\mathcal{O}'$ be the prefix of $\mathcal{U}_{\mathcal{S}'}$ (i.e. branching process of $\mathcal{S}'$) such that there exists a surjective homomorphism $\phi : \mathcal{O}_{|\mathcal{V}'} \to \mathcal{O}'$. We call $\mathcal{O}'$ the* projection *of $\mathcal{O}$ to variables $\mathcal{V}'$, and denote it by $\Pi_{\mathcal{V}'}(\mathcal{O})$.*

$\Pi_{\mathcal{V}'}(\mathcal{O})$ is well defined, as the union of configurations of $\mathcal{U}_{\mathcal{S}'}$ which are isomorphic to a configuration of $\mathcal{O}_{\mathcal{V}'}$. $\Pi_{\mathcal{V}'}(\mathcal{O})$ can be easily obtained from $\mathcal{O}_{|\mathcal{V}'}$, by a recursive procedure starting at $\min(\mathcal{O}_{|\mathcal{V}'})$ and "merging" events e and e' which have the same pre-set and the same label (merging two events means merging also conditions of their respective post-sets). Details are left to the reader.

Taking for $\mathcal{O}_{|\mathcal{V}'}$ the restriction of $\mathcal{O}$ to a subset of nodes may erase causality or conflict relations between these nodes, which could then appear as unduely concurrent in $\mathcal{O}_{|\mathcal{V}'}$. This is obvious on the examples of figures 4 and 5, taking the global unfolding for $\mathcal{O}$ and considering its projection on variables $\mathcal{V}' = \{B_1, B_2\}$. As a consequence, there may be configurations of $\Pi_{\mathcal{V}'}(\mathcal{O})$ which are not the restriction of a configuration of $\mathcal{O}$. We will say that the projection $\Pi_{\mathcal{V}'}$ is *misleading* on $\mathcal{O}$ when this phenomenon occurs.

Definition 3 $\Pi_{\mathcal{V}'}(\mathcal{O})$ *is* not misleading *if, for every configuration κ' of $\Pi_{\mathcal{V}'}(\mathcal{O})$, there exists a configuration κ of $\mathcal{O}$ such that $\Pi_{\mathcal{V}'}(\kappa) = \kappa'$ and $\preceq_\kappa$ coincides with $\preceq_{\kappa'}$ on κ'.*

Lemma 10 *Projections on a single variable are never misleading.*

Proof. Let $\mathcal{S}$ be a system defined on a single variable (i.e. a standard automaton). On its unfolding, two nodes are either causally related or in conflict : concurrency is impossible, hence false concurrency due to the projection cannot occur. □

For $\mathcal{V}' \not\subseteq \mathcal{V}$ we define $\mathcal{S}_{|\mathcal{V}'}$ as $\mathcal{S}_{|\mathcal{V}'\cap\mathcal{V}}$, and in the same way, for $\mathcal{O}$ a branching process of $\mathcal{S}$ we define $\Pi_{\mathcal{V}'}(\mathcal{O})$ as $\Pi_{\mathcal{V}'\cap\mathcal{V}}(\mathcal{O})$.

Proposition 4 *The mappings $\Pi_{\mathcal{V}'}$ are projectors. More generally, they satisfy*

$$\forall \mathcal{V}_1, \mathcal{V}_2 \subseteq \mathcal{V}, \qquad \Pi_{\mathcal{V}_2} \circ \Pi_{\mathcal{V}_1} \;=\; \Pi_{\mathcal{V}_1\cap\mathcal{V}_2} \tag{3}$$

Proof. Since $\Pi_{\mathcal{V}_1}(\mathcal{O})$ is a branching process of $\mathcal{S}_{|\mathcal{V}_1}$, $\Pi_{\mathcal{V}_2} \circ \Pi_{\mathcal{V}_1}$ must be read as $\Pi_{\mathcal{V}_1\cap\mathcal{V}_2} \circ \Pi_{\mathcal{V}_1}$, so we must prove $\Pi_{\mathcal{V}_1\cap\mathcal{V}_2} \circ \Pi_{\mathcal{V}_1} = \Pi_{\mathcal{V}_1\cap\mathcal{V}_2}$. Let $\mathcal{O}$ be a branching process of $\mathcal{S}$, we have to prove that $\mathcal{O}_{1;2} = \Pi_{\mathcal{V}_1\cap\mathcal{V}_2}(\Pi_{\mathcal{V}_1}(\mathcal{O}))$ is isomorphic to $\mathcal{O}'_{1;2} = \Pi_{\mathcal{V}_1\cap\mathcal{V}_2}(\mathcal{O})$. We will rely on lemma 1.

The construction of $\mathcal{O}_{1;2}$ uses four steps. First, $\mathcal{O}$ is restricted to $\mathcal{V}_1$, then a surjective homomorphism $\phi_1 : \mathcal{O}_{|\mathcal{V}_1} \to \mathcal{O}_1 = \Pi_{\mathcal{V}_1}(\mathcal{O})$ maps it to a branching process of $\mathcal{S}_{|\mathcal{V}_1}$. $\mathcal{O}_1$ is in turn restricted to $\mathcal{V}_1 \cap \mathcal{V}_2$, then another surjective homomorphism $\phi_{1;2} : \mathcal{O}_{1|\mathcal{V}_1\cap\mathcal{V}_2} \to \mathcal{O}_{1;2} = \Pi_{\mathcal{V}_1\cap\mathcal{V}_2}(\mathcal{O}_1)$ maps it to a branching process of $\mathcal{S}_{|\mathcal{V}_1\cap\mathcal{V}_2}$. The construction of $\mathcal{O}'_{1;2}$ uses only two steps : $\mathcal{O}$ is directly restricted to $\mathcal{V}_1 \cap \mathcal{V}_2$, then mapped to a branching process of $\mathcal{S}_{|\mathcal{V}_1\cap\mathcal{V}_2}$ by a surjective homomorphism $\phi'_{1;2} : \mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2} \to \mathcal{O}'_{1;2} = \Pi_{\mathcal{V}_1\cap\mathcal{V}_2}(\mathcal{O})$.

Observe that homomorphism ϕ_1, defined over $\mathcal{O}_{|\mathcal{V}_1}$, also applies to its sub-net $\mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2}$. Moreover, one easily checks that the image of $\mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2}$ by ϕ_1 coincides with $\phi_1(\mathcal{O}_{|\mathcal{V}_1})_{|\mathcal{V}_1\cap\mathcal{V}_2} = \mathcal{O}_{1|\mathcal{V}_1\cap\mathcal{V}_2}$. This results in the following diagram :

$$\begin{array}{ccc}
\mathcal{O}_{|\mathcal{V}_1} & \xrightarrow{\phi_1} & \mathcal{O}_1 = \Pi_{\mathcal{V}_1}(\mathcal{O}) \\
\downarrow & & \downarrow \\
\mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2} & \xrightarrow{\phi_1} & \mathcal{O}_{1|\mathcal{V}_1\cap\mathcal{V}_2} \\
\phi'_{1;2} \downarrow & & \downarrow \phi_{1;2} \\
\mathcal{O}'_{1;2} = \Pi_{\mathcal{V}_1\cap\mathcal{V}_2}(\mathcal{O}) & & \mathcal{O}_{1;2} = \Pi_{\mathcal{V}_1\cap\mathcal{V}_2}(\mathcal{O}_1)
\end{array}$$

We first show that any configuration κ' of $\mathcal{O}'_{1;2}$ is isomorphic to a configuration κ of $\mathcal{O}_{1;2}$. We actually limit ourselves to a covering set of configurations. Let κ'

be the smallest configuration of $\mathcal{O}'_{1;2}$ containing some particular event e' (and thus all its predecessors). By definition of $\phi'_{1;2}$, there exists at least one configuration κ'_1 in $\mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2}$ such that $\phi'_{1;2}(\kappa'_1) = \kappa'$ ($\phi'_{1;2}$ makes κ'_1 and κ' isomorphic). e'_1, the inverse image of e' by $\phi'_{1;2}$, is a maximal event of κ'_1. There exists at least one configuration κ'_2 of $\mathcal{O}_{|\mathcal{V}_1}$ containing e'_1 as maximal event, and thus containing all κ'_1. One has $\kappa'_1 = \kappa'_{2|\mathcal{V}_1\cap\mathcal{V}_2}$. Since ϕ_1 is injective on κ'_2, it is also injective on κ'_1, so κ'_1 is isomorphic to $\phi_1(\kappa'_1)$, and thus to $\kappa = \phi_{1;2}(\phi_1(\kappa'_1))$ (again, $\phi_{1;2}$ restricted to a configuration is injective). Hence κ' and κ are isomorphic.

For the converse part, we use a similar argument. Let e be an event of $\mathcal{O}_{1;2}$, and κ be the smallest configuration containing all predecessors of e. Let κ_1 be a configuration of $\mathcal{O}_{1|\mathcal{V}_1\cap\mathcal{V}_2}$ such that $\phi_{1;2}(\kappa_1) = \kappa$ and let e_1 be the (maximal) event of κ_1 such that $\phi_{1;2}(e_1) = e$. Let[9] κ_2 be a configuration of $\mathcal{O}_1$ containing e_1 as maximal event ; κ_2 contains all nodes of κ_1 and $\kappa_{2|\mathcal{V}_1\cap\mathcal{V}_2} = \kappa_1$. There exists at least one κ'_2 in $\mathcal{O}_{|\mathcal{V}_1}$ satisfying $\phi_1(\kappa'_2) = \kappa_2$. Let κ'_1 be the restriction of κ'_2 to $\mathcal{V}_1 \cap \mathcal{V}_2$; κ'_1 is a configuration of $\mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2}$ and satisfies $\phi_1(\kappa'_1) = \kappa_1$ (isomorphism). Now, by $\kappa' = \phi'_{1;2}(\kappa'_1)$, one gets a configuration of $\mathcal{O}'_{1;2}$ which is isomorphic to κ. □

5.2 Projection of a product

Proposition 5 *Let $\mathcal{O}$ be a branching process of $\mathcal{S} = \mathcal{S}_1^e \| \mathcal{S}_2^e$, and let ψ_1 be the partial homomorphism relating $\mathcal{U}_\mathcal{S}$ to $\mathcal{U}_{\mathcal{S}_1^e}$. Then $\psi_1(\mathcal{O})$ is isomorphic to $\Pi_{\mathcal{V}_1}(\mathcal{O})$.*

Proof. As for proposition 4, we rely on lemma 1. We denote by $\phi_1 : \mathcal{O}_{|\mathcal{V}_1} \to \Pi_{\mathcal{V}_1}(\mathcal{O})$ the surjective homomorphism defining $\Pi_{\mathcal{V}_1}(\mathcal{O})$.

Let κ_1 be a configuration of $\psi_1(\mathcal{O})$, there exists κ, a configuration of $\mathcal{O}$, such that $\kappa_1 = \psi_1(\kappa)$. One easily checks that $\kappa'_1 = \kappa_{|\mathcal{V}_1}$ is isomorphic to κ_1. So $\phi_1(\kappa'_1)$ is a configuration of $\Pi_{\mathcal{V}_1}(\mathcal{O})$ which is isomorphic to κ_1.

For the converse part, we consider only a covering set of configurations of $\Pi_{\mathcal{V}_1}(\mathcal{O})$. Let e_1 be an event of $\Pi_{\mathcal{V}_1}(\mathcal{O})$ and form κ_1 as the smallest configuration containing all predecessors of e_1. By definition of ϕ_1, there exists at least one configuration κ'_1 in $\mathcal{O}_{|\mathcal{V}_1}$ such that $\phi_1(\kappa'_1) = \kappa_1$. Let e'_1 be the reverse image of e_1 in κ'_1. There exists a configuration κ of $\mathcal{O}$ with e'_1 as maximal event. Hence κ contains all κ'_1, and further $\kappa'_1 = \kappa_{|\mathcal{V}_1}$. Again, since $\kappa_{|\mathcal{V}_1}$ is isomorphic to $\psi_1(\kappa)$, there exists a configuration of $\psi_1(\mathcal{O})$ which is isomorphic to κ_1. □

[9] At this point, we cannot use $\phi_1 : \mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2} \to \mathcal{O}_{1|\mathcal{V}_1\cap\mathcal{V}_2}$ in the reverse direction : it is not proved that any configuration of $\mathcal{O}_{1|\mathcal{V}_1\cap\mathcal{V}_2}$ has an isomorphic inverse image in $\mathcal{O}_{|\mathcal{V}_1\cap\mathcal{V}_2}$ by ϕ_1. We only know this between $\mathcal{O}_1$ and $\mathcal{O}_{|\mathcal{V}_1}$, by definition of ϕ_1.

This result transfers properties of the ψ_i to projections $\Pi_{\mathcal{V}_i}$; for example, it provides another way to obtain proposition 4 with the help of the chain rule evidenced in corollary 1 (left to the reader as an exercise). Proposition 5 trivially extends to N components, and thus also reveals that the problem of finding minimal factors of $\mathcal{O}$ or of $\mathcal{U}_\mathcal{S}$ amounts to performing projections.

Structural assumption. *From now, we assume $\mathcal{S}$ decomposes as $\mathcal{S} = \mathcal{S}_1 \| \dots \| \mathcal{S}_N$ where components $\mathcal{S}_i$ have distinct tile sets. So interactions between components are only due to shared variables, and the "span" of each tile of $\mathcal{S}$ is limited to some $\mathcal{V}_i$ $(\forall \mathbf{t}, \exists i : \gamma(\mathbf{t}) \subseteq \mathcal{V}_i)$.*

Theorem 2 *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be branching processes of $\mathcal{S}_1^e$ and $\mathcal{S}_2^e$, and select variables $\mathcal{V}_3$ such that $\mathcal{V}_1 \cap \mathcal{V}_2 \subseteq \mathcal{V}_3$. If projections $\Pi_{\mathcal{V}_3}(\mathcal{O}_1)$ and $\Pi_{\mathcal{V}_3}(\mathcal{O}_2)$ are not misleading, then $\Pi_{\mathcal{V}_3}(\mathcal{O}_1 \wedge \mathcal{O}_2) = \Pi_{\mathcal{V}_3}(\mathcal{O}_1) \wedge \Pi_{\mathcal{V}_3}(\mathcal{O}_2)$.*

Theorem 2 has a central role in the sequel. Its application range covers at least the case where components interact through a single variable (lemma 10), whence the subtitle of the paper.

Proof.

By $\mathcal{O}_1 \wedge \mathcal{O}_2 = \Pi_{\mathcal{V}_1 \cup \mathcal{V}_2}(\mathcal{O}_1 \wedge \mathcal{O}_2)$ and proposition 4, $\Pi_{\mathcal{V}_3}(\mathcal{O}_1 \wedge \mathcal{O}_2) = \Pi_{\mathcal{V}_3 \cap (\mathcal{V}_1 \cup \mathcal{V}_2)}(\mathcal{O}_1 \wedge \mathcal{O}_2)$. Similarly, $\Pi_{\mathcal{V}_3}(\mathcal{O}_i) = \Pi_{\mathcal{V}_3 \cap \mathcal{V}_i}(\mathcal{O}_i)$, $i = 1, 2$, so we have to prove $\Pi_{\mathcal{V}_3 \cap (\mathcal{V}_1 \cup \mathcal{V}_2)}(\mathcal{O}_1 \wedge \mathcal{O}_2) = \Pi_{\mathcal{V}_3 \cap \mathcal{V}_1}(\mathcal{O}_1) \wedge \Pi_{\mathcal{V}_3 \cap \mathcal{V}_2}(\mathcal{O}_2)$. Without loss of generality, we can thus assume $\mathcal{V}_3 \subseteq \mathcal{V}_1 \cup \mathcal{V}_2$.

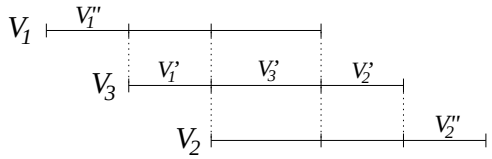

Figure 7: *Decomposition of $\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3$ for $\mathcal{V}_1 \cap \mathcal{V}_2 \subseteq \mathcal{V}_3 \subseteq \mathcal{V}_1 \cup \mathcal{V}_2$.*

We first prove $\Pi_{\mathcal{V}_3}(\mathcal{O}_1 \wedge \mathcal{O}_2) \sqsubseteq \Pi_{\mathcal{V}_3}(\mathcal{O}_1) \wedge \Pi_{\mathcal{V}_3}(\mathcal{O}_2)$. Decompose $\mathcal{V}_1 \cup \mathcal{V}_2$ as $\mathcal{V}_1'' \cup \mathcal{V}_1' \cup \mathcal{V}_3' \cup \mathcal{V}_2' \cup \mathcal{V}_2''$ where $\mathcal{V}_3' = \mathcal{V}_1 \cap \mathcal{V}_2$, $\mathcal{V}_i'' = \mathcal{V}_i \setminus \mathcal{V}_3$ and $\mathcal{V}_i' = \mathcal{V}_3 \setminus \mathcal{V}_j$, for $1 \leq i, j \leq 2$, $i \neq j$ (fig. 7). One has $\mathcal{V}_3 = \mathcal{V}_1' \cup \mathcal{V}_3' \cup \mathcal{V}_2'$. Let κ be a configuration of $\mathcal{O}_1 \wedge \mathcal{O}_2$, then $\kappa = \kappa_1 \wedge \kappa_2$ with $\kappa_i = \psi_i(\kappa) = \Pi_{\mathcal{V}_i}(\kappa), i = 1, 2$ by proposition 2 and proposition 5. With the same arguments, κ_1 can be further decomposed as $\kappa_1 = \kappa_1'' \wedge \kappa_1' \wedge \kappa_3'$ where $\kappa_1'' = \Pi_{\mathcal{V}_1 \setminus \mathcal{V}_3}(\kappa_1) = \Pi_{\mathcal{V}_1 \setminus \mathcal{V}_3}(\Pi_{\mathcal{V}_1}(\kappa)) = \Pi_{\mathcal{V}_1 \setminus \mathcal{V}_3}(\kappa) = \Pi_{\mathcal{V}_1''}(\kappa)$, and in the same way

$\kappa'_1 = \Pi_{\mathcal{V}'_1}(\kappa), \kappa'_3 = \Pi_{\mathcal{V}'_3}(\kappa)$. Symmetrically, κ_2 decomposes as $\kappa_2 = \kappa'_3 \wedge \kappa'_2 \wedge \kappa''_2$ with the *same* κ'_3. One easily derives that $\Pi_{\mathcal{V}_3}(\kappa_1) = \Pi_{\mathcal{V}_3 \cap \mathcal{V}_1}(\kappa_1) = \kappa'_1 \wedge \kappa'_3$, $\Pi_{\mathcal{V}_3}(\kappa_2) = \kappa'_3 \wedge \kappa'_2$ and $\Pi_{\mathcal{V}_3}(\kappa) = \kappa'_1 \wedge \kappa'_3 \wedge \kappa'_2$, which proves $\Pi_{\mathcal{V}_3}(\kappa_1) \wedge \Pi_{\mathcal{V}_3}(\kappa_2) = \Pi_{\mathcal{V}_3}(\kappa)$.

To prove $\Pi_{\mathcal{V}_3}(\mathcal{O}_1 \wedge \mathcal{O}_2) \sqsupseteq \Pi_{\mathcal{V}_3}(\mathcal{O}_1) \wedge \Pi_{\mathcal{V}_3}(\mathcal{O}_2)$, let κ_3 be a configuration of $\Pi_{\mathcal{V}_3}(\mathcal{O}_1) \wedge \Pi_{\mathcal{V}_3}(\mathcal{O}_2)$. κ_3 decomposes as $\bar{\kappa}_1 \wedge \bar{\kappa}_2$, where $\bar{\kappa}_i \in \Pi_{\mathcal{V}_3}(\mathcal{O}_i)$. As above, one has $\bar{\kappa}_1 = \kappa'_1 \wedge \kappa'_3$, $\bar{\kappa}_2 = \kappa'_3 \wedge \kappa'_2$ and $\kappa_3 = \kappa'_1 \wedge \kappa'_3 \wedge \kappa'_2$. Since $\Pi_{\mathcal{V}_3}(\mathcal{O}_i)$ is not misleading, there exists a $\kappa_i \sqsubseteq \mathcal{O}_i$ such that $\bar{\kappa}_i = \Pi_{\mathcal{V}_3}(\kappa_i)$, $i = 1, 2$. Again, κ_i decomposes as $\kappa''_i \wedge \bar{\kappa}_i$ with $\kappa''_i = \Pi_{\mathcal{V}''_i}(\kappa_i)$. Let us build $\kappa = \kappa_1 \wedge \kappa_2 = \kappa''_1 \wedge \kappa_3 \wedge \kappa''_2$ in $\mathcal{O}_1 \wedge \mathcal{O}_2$; we must show that $\bar{\kappa}_3 \triangleq \Pi_{\mathcal{V}_3}(\kappa)$ coincides with κ_3.

We rely on procedure 2 used to build $\kappa''_1 \wedge \kappa_3 \wedge \kappa''_2$. κ''_1 and κ''_2 are branching processes of $\mathcal{S}_{|\mathcal{V}_1 \setminus \mathcal{V}_3}$ and $\mathcal{S}_{|\mathcal{V}_2 \setminus \mathcal{V}_3}$ respectively, which share no variable by assumption on $\mathcal{V}_3$, and thus no tile by the structural assumption on $\mathcal{S}$. Hence events of these branching processes only synchronize with those of κ_3, in an exclusive manner. In the construction of κ, any order in which events of κ_3 have been used is valid both for the construction of $\kappa''_1 \wedge \kappa_3$ and for $\kappa''_2 \wedge \kappa_3$, by definition 3. Hence procedure 2 cannot block before all events of κ_3, κ''_1 and κ''_2 have appeared in κ, which proves $\Pi_{\mathcal{V}_3}(\kappa) = \kappa_3$. □

All conditions appearing in the theorem are important. For example, if $\mathcal{V}_1 \cap \mathcal{V}_2 \not\subseteq \mathcal{V}_3$, i.e. if components $\mathcal{S}^e_1, \mathcal{S}^e_2$ can interact "outside" $\mathcal{V}_3$, the result fails (counter-examples are easy to build; this is left to the reader). More interestingly, if projections are misleading, the result fails also. There may be several reasons for this; fig. 8 illustrates the case where antinomic causality relations are lost, which results in "false" configurations on shared variables.

Corollary 2 *With notations of theorem 2, whether projections are misleading or not, one always has* $\Pi_{\mathcal{V}_3}(\mathcal{O}_1 \wedge \mathcal{O}_2) \sqsubseteq \Pi_{\mathcal{V}_3}(\mathcal{O}_1) \wedge \Pi_{\mathcal{V}_3}(\mathcal{O}_2)$.

This corresponds to the first part of the proof, which doesn't use the non misleading assumption.

5.3 Modular computation of minimal factors

We now have all the elements to compute minimal factors of $\mathcal{U}_\mathcal{S}$ (or $\mathcal{O}$) without computing $\mathcal{U}_\mathcal{S}$ (resp. $\mathcal{O}$) itself. We start with an example: $\mathcal{S} = \mathcal{S}_1 \| \mathcal{S}_2$. Let $\mathcal{O}_1, \mathcal{O}_2$ be branching processes of $\mathcal{S}^e_1, \mathcal{S}^e_2$ respectively. The objective is to determine say

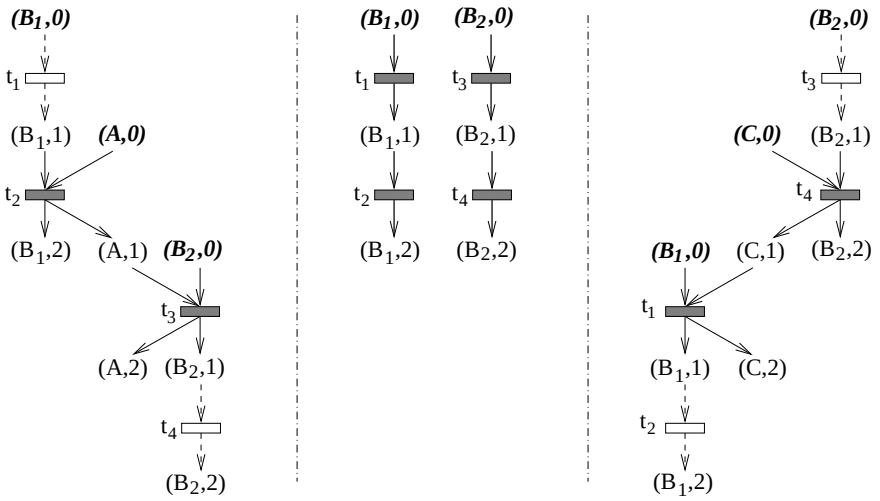

Figure 8: *Two branching processes $\mathcal{O}_1$ (left), BP of $\mathcal{S}_{|\{A,B_1,B_2\}}$, and $\mathcal{O}_2$ (right), BP of $\mathcal{S}_{|\{B_1,B_2,C\}}$. Projections $\Pi_{\{B_1,B_2\}}(\mathcal{O}_1)$ and $\Pi_{\{B_1,B_2\}}(\mathcal{O}_2)$ are misleading: they erase a causality relation. These projections are identical (center), thus coincide with $\Pi_{\{B_1,B_2\}}(\mathcal{O}_1) \wedge \Pi_{\{B_1,B_2\}}(\mathcal{O}_2)$, while the product of the two branching processes reduces to a single configuration: the initial state of $\mathcal{S}$...*

$\mathcal{O}'_1 = \psi_1(\mathcal{O}_1 \wedge \mathcal{O}_2) = \Pi_{\mathcal{V}_1}(\mathcal{O}_1 \wedge \mathcal{O}_2)$. One has

$$\begin{aligned} \mathcal{O}'_1 &= \Pi_{\mathcal{V}_1}(\mathcal{O}_1 \wedge \mathcal{O}_2) \\ &= \Pi_{\mathcal{V}_1}(\mathcal{O}_1) \wedge \Pi_{\mathcal{V}_1}(\mathcal{O}_2) \\ &= \mathcal{O}_1 \wedge \Pi_{\mathcal{V}_1 \cap \mathcal{V}_2}(\mathcal{O}_2) \end{aligned} \tag{4}$$

using theorem 2. The term $\mathcal{O}_{1;2} = \Pi_{\mathcal{V}_1 \cap \mathcal{V}_2}(\mathcal{O}_2)$ is a branching process of the *interface* $\mathcal{S}^e_{1;2} \triangleq \mathcal{S}_{|\mathcal{V}_1 \cap \mathcal{V}_2}$ between the two components. Hence in (4) the factors are relatively "small" with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$, and *a fortiori* with respect to $\mathcal{O}_1 \wedge \mathcal{O}_2$. Observe also that the product in (4) actually *reduces* $\mathcal{O}_1$ to a prefix.

To generalize these ideas to[10] $\mathcal{S} = \mathcal{S}_1 \| \dots \| \mathcal{S}_N$, we first gather properties of $\wedge$ and projectors Π. Since we only consider branching processes of restrictions of $\mathcal{S}$, we have

$$\forall \mathcal{O},\ \exists \mathcal{V}' \subseteq \mathcal{V} : \quad \mathcal{O} = \Pi_{\mathcal{V}'}(\mathcal{O}) \tag{5}$$

Moreover, by proposition 4, projectors satisfy

$$\forall \mathcal{V}', \mathcal{V}'' \subseteq \mathcal{V}, \quad \Pi_{\mathcal{V}'} \circ \Pi_{\mathcal{V}''} = \Pi_{\mathcal{V}' \cap \mathcal{V}''} \tag{6}$$

and by theorem 2, for branching processes $\mathcal{O}'$, $\mathcal{O}''$ of $\mathcal{S}_{|\mathcal{V}'}$, $\mathcal{S}_{|\mathcal{V}''}$,

$$\forall \mathcal{W} \supseteq \mathcal{V}' \cap \mathcal{V}'', \ \ \Pi_{\mathcal{W}}(\mathcal{O}' \wedge \mathcal{O}'') = \Pi_{\mathcal{W}}(\mathcal{O}') \wedge \Pi_{\mathcal{W}}(\mathcal{O}'') \tag{7}$$

Finally, $\wedge$ is a commutative and associative product on branching processes. Its unit element $\mathbb{I}$ corresponds to the unfolding of the trivial tile system (no variable, no tile) :

$$\forall \mathcal{O},\ \mathcal{O} \wedge \mathbb{I} = \mathcal{O} \qquad \text{and} \qquad \mathbb{I} = \Pi_{\emptyset}(\mathbb{I}) \tag{8}$$

These four properties coincide with the core of the axiomatic framework developed in [12] to design modular processings.

The second ingredient concerns the structure of the compound tile system $\mathcal{S}$. We associate a *connection graph* $\mathcal{G}^{cnx}$ to $\mathcal{S}$. $\mathcal{G}^{cnx}$ has $\{1, \dots, N\}$ as vertices, one vertex per component, and (i, j) is an edge of $\mathcal{G}^{cnx}$ iff $\mathcal{V}_i \cap \mathcal{V}_j \neq \emptyset$, i.e. iff components $\mathcal{S}_i, \mathcal{S}_j$ have common variables. A *communication graph* $\mathcal{G}^c$ is then derived from $\mathcal{G}^{cnx}$ by recursively removing *superfluous* edges, until this is not possible any more. An edge (i, j) is said to be superfluous on graph $\mathcal{G}$ iff there exists a path $(i, k_1, k_2, \dots, k_L, j)$ on $\mathcal{G}$ such that $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_{k_l}$ for every k_l, $1 \leq l \leq L$. In words, the direct interaction

[10] Recall that components of $\mathcal{S}$ are assumed to have no tile in common.

between $\mathcal{S}_i$ and $\mathcal{S}_j$ is captured by the path $(\mathcal{S}_i, \mathcal{S}_{k_1}, \ldots, \mathcal{S}_{k_L}, \mathcal{S}_j)$. In general, a system $\mathcal{S}$ has several communication graphs. However, if one of them is a tree, then all communication graphs of $\mathcal{S}$ are trees. *In this paper, we limit ourselves to systems living on a tree.*

We now have all elements to determine minimal factors $\Pi_{\mathcal{V}_i}(\mathcal{U}_\mathcal{S}) = \Pi_{\mathcal{V}_i}(\mathcal{U}_{\mathcal{S}_1} \wedge \ldots \wedge \mathcal{U}_{\mathcal{S}_N})$ by modular computations[11]. This takes the form of a message passing algorithm (MPA) run on a communication graph of $\mathcal{S}$. For each pair of neighbour components $\mathcal{S}_i, \mathcal{S}_j$ on $\mathcal{G}^c$, the MPA recursively updates a message $\mathcal{M}_{i,j}$ which progressively brings to $\mathcal{S}_j$ all necessary information from the subtree lying beyond $\mathcal{S}_i$. Observe that there are two messages per edge of $\mathcal{G}^c$, one in each direction. Denoting by $\mathcal{N}(i)$ the neigbours of vertex i on $\mathcal{G}^c$, the MPA is given by

Procedure 3

1. Initialization

$$\mathcal{M}_{i,j} \quad = \quad \mathbb{I}, \qquad \forall (i,j) \in \mathcal{G}^c \tag{9}$$

2. Until stability of messages, select an edge (i,j) and apply the update rule

$$\mathcal{M}_{i,j} \quad := \quad \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{U}_{\mathcal{S}_i^e} \wedge (\bigwedge_{k \in \mathcal{N}(i) \setminus j} \mathcal{M}_{k,i})] \tag{10}$$

3. Termination

$$\bar{\mathcal{U}}_{\mathcal{S}_i^e} \quad = \quad \mathcal{U}_{\mathcal{S}_i^e} \wedge (\bigwedge_{k \in \mathcal{N}(i)} \mathcal{M}_{k,i}), \qquad 1 \leq i \leq N \tag{11}$$

In procedure 3, the order of updates is left unspecified. Nevertheless, one can prove that the procedure converges to a unique stable point in a finite number of useful updates[12], where usefulness refers to the fact that the message does change in the update. Moreover, this stationary point yields $\bar{\mathcal{U}}_{\mathcal{S}_i^e} = \Pi_{\mathcal{V}_i}(\wedge_i \mathcal{U}_{\mathcal{S}_i^e}) = \Pi_{\mathcal{V}_i}(\mathcal{U}_\mathcal{S})$ (see theorem 1 in [12]).

[11] To compute minimal factors of $\mathcal{O} = \wedge_i \mathcal{O}_i$, $\mathcal{U}_{\mathcal{S}_i^e}$ must be replaced by $\mathcal{O}_i$ in all equations.

[12] One may object that each product $\wedge$ in the update step requires infinite time if infinite branching processes are considered. Nevertheless, in this paper we adopt this macro-granularity of time. Future work will focus on the recursive computation of a finite complete prefix of $\mathcal{U}_\mathcal{S}$.

5.4 Approximation of minimal factors

Other results obtained in the abstract setting of [12] have consequences on the computation of minimal factors. We briefly mention them below, without proof.

Computing minimal factors $\Pi_{\mathcal{V}_i}(\mathcal{U}_\mathcal{S})$ amounts to pruning the local unfolding $\mathcal{U}_{\mathcal{S}_i^e}$, by removing nodes which have no counterpart in neighbour unfoldings. We are thus solving a constraint system, with a distributed algorithm. This is evidenced by the so-called *involutivity* property on branching processes, a salient property of constraint systems. Involutivity expresses that composing a (constraint) system with part of itself doesn't change this system :

Lemma 11 *Let $\mathcal{O}$ be a branching process of $\mathcal{S}$, let $\mathcal{V}' \subseteq \mathcal{V}$, then $\mathcal{O} \wedge \Pi_{\mathcal{V}'}(\mathcal{O}) = \mathcal{O}$.*

Proof. One has $\mathcal{O} \sqsubseteq \Pi_{\mathcal{V}'}(\mathcal{O}) \wedge \Pi_{\mathcal{V}\setminus\mathcal{V}'}(\mathcal{O})$ by propositions 2 and 5. Since $\Pi_{\mathcal{V}\setminus\mathcal{V}'}(\mathcal{O}) \sqsubseteq \mathcal{U}_{\mathcal{S}_{|\mathcal{V}\setminus\mathcal{V}'}}$, one has $\mathcal{O} \sqsubseteq \Pi_{\mathcal{V}'}(\mathcal{O}) \wedge \mathcal{U}_{\mathcal{S}_{|\mathcal{V}\setminus\mathcal{V}'}}$ (lemma 6). On the other hand, one easily shows $\mathcal{O} = \mathcal{O} \wedge \mathcal{U}_{\mathcal{S}_{|\mathcal{V}'}} \wedge \mathcal{U}_{\mathcal{S}_{|\mathcal{V}\setminus\mathcal{V}'}}$ (theorem 1), so $\mathcal{O} \wedge \Pi_{\mathcal{V}'}(\mathcal{O}) = \mathcal{O} \wedge (\mathcal{U}_{\mathcal{S}_{|\mathcal{V}\setminus\mathcal{V}'}} \wedge \Pi_{\mathcal{V}'}(\mathcal{O}))$. Since $\mathcal{O}$ is a prefix of $\mathcal{U}_{\mathcal{S}_{|\mathcal{V}\setminus\mathcal{V}'}} \wedge \Pi_{\mathcal{V}'}(\mathcal{O})$, the right hand side term equals $\mathcal{O}$. □

Involutivity is a strong property. First, it allows to design an alternate and somehow simpler distributed algorithm to compute minimal factors. But, more interestingly, it reveals properties of procedure 3 when applied to systems *not living on a tree*. Convergence of the procedure to a unique stationary point can still be proved. The resulting branching processes $\bar{\mathcal{U}}_{\mathcal{S}_i^e}$ are still factors of $\mathcal{U}_\mathcal{S}$, so they have the desired minimal factors $\Pi_{\mathcal{V}_i}(\mathcal{U}_\mathcal{S})$ as prefixes, although they are generally larger. Extra properties of the $\bar{\mathcal{U}}_{\mathcal{S}_i^e}$ reveal that they are "close" to minimal factors $\Pi_{\mathcal{V}_i}(\mathcal{U}_\mathcal{S})$; We refer the reader to [12] for more details.

Still relying on involutivity, it can be proved that procedure 3 converges if (7) is replaced by

$$\forall \mathcal{W} \supseteq \mathcal{V}' \cap \mathcal{V}'', \;\; \Pi_\mathcal{W}(\mathcal{O}' \wedge \mathcal{O}'') \sqsubseteq \Pi_\mathcal{W}(\mathcal{O}') \wedge \Pi_\mathcal{W}(\mathcal{O}'') \tag{12}$$

whatever the structure of the connection graph. Corollary 2 thus reveals that procedure 3 can be applied even if projections are misleading. Again, one still obtains (non minimal) factors of $\mathcal{U}_\mathcal{S}$, i.e. no event is unduely discarded in the $\mathcal{U}_{\mathcal{S}_i^e}$.

6 Conclusion

We have proposed a reshaping of safe Petri nets under the form of tile systems, operating on local sets of variables instead of places. These systems are composable, by variable sharing, and their structure can be represented under the form of

an interaction graph. Under this form, modular systems are reminiscent of similar structures appearing in probability theory : Markov random fields (or Bayesian networks), which define the probability distribution of a large number of random variables through local specifications on part of these variables. One derives from this modular specification a set of conditional independance relations on the random variables of the field, and a factorization property of the overall probability distribution. On the side of dynamic systems, one rather specifies the behaviour of the compound system as the conjunction of local possible transitions. By analogy, we have shown that the unfolding of a compound system also satisfies a factorization property, and that the behaviors of two components are independent given that they coincide on the shared variables.

The unfolding of a system with concurrency takes advantage of concurrency to describe in a compact manner all possible behaviors of that system (in view of property verification for example, or for failure diagnosis [13]). The factorization we propose pushes further this idea by also taking advantage of local interactions between components. To compute this compact representation of the system behavior, one can again directly benefit of the analogy with Markov fields, for which many processings can be carried out by modular algorithms, based on local computations. We have proposed one of them to compute minimal factors of an unfolding, but this family of algorithms can capture many other problems (for example distributed diagnosis [15], [16]).

As mentioned in the introduction, the present work is limited to non-misleading projections. The general case requires extended notions of branching processes, of product $\wedge$ and of projections, but the main ideas of section 5 remain the same, up to these technical refinements. This will be presented in a follow-up of this paper. In the same way, we have only presented a computation of minimal factors by recursion "in space," leaving apart the usual recursion "in time" where tiles are connected one after the other. Actually, the two recursions can be merged into a single one, which avoids using extended components : each time a component connects a proper tile to its local unfolding, and if this tile impacts shared variables, it informs its neighbour(s) of the new event, which they can in turn incorporate to their own local unfolding. This procedure is already suggested in [15] and will be presented in a forthcoming paper.

Finally, let us risk the following idea. Another limitation of the present paper is related to the interaction structure of the compound system, required to be a tree. For general structures, minimal factors may be hard to obtain exactly. Actually, this complexity can be related to the number of cycles in the interaction structure. How-

ever, the modular algorithm we have proposed can be used to provide *approximate* minimal factors, with interesting properties, at a *low cost*. In particular, if some given behavior is forbidden by these approximate factors, it is certainly forbidden in the true system (while the converse is false). Although not exact, this method may thus provide useful information.

References

[1] M. Nielsen, G. Plotkin, G. Winskel, *Petri nets, event structures and domains*, Theoretical Computer Science 13(1), 1980, pp. 85-108.

[2] J. Engelfriet, *Branching Processes of Petri Nets*, Acta Informatica 28, 1991, pp. 575-591.

[3] K.L. McMillan, *Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits*, in Proc. 4th Workshop of Computer Aided Verification, Montreal, 1992, pp. 164-174.

[4] J. Esparza, S. Römer, *An unfolding algorithm for synchronous products of transition systems*, in proc. of CONCUR'99, LNCS 1664, Springer Verlag, 1999.

[5] J. Esparza, S. Römer, W. Vogler, *An improvement of McMillan's unfolding algorithm*, in Proc. of TACAS'96, LNCS 1055, pp. 87-106.

[6] J. Esparza, S. Römer, W. Vogler, *An Improvement of McMillan's Unfolding Algorithm*, Formal Methods in System Design 20(3), pp. 285-310, May 2002. Extended version of [5].

[7] J. Esparza, *Model checking using net unfoldings*, Science and Computer Programming 23, pp. 151-195, 1994.

[8] J. Esparza, C. Schröter, *Reachability Analysis Using Net Unfoldings*, Workshop of Concurrency, Specification and Programming, volume II of Informatik-Bericht 140, pp. 255-270, Humboldt-Universität zu Berlin, 2000.

[9] S. Melzer, S. Römer, *Deadlock checking using net unfoldings*, CAV'97, LNCS 1254, pp. 352-363.

[10] J.-M. Couvreur, S. Grivet, D. Poitrenaud, *Unfolding of Products of Symmetrical Petri Nets*, 22nd International Conference on Applications and Theory of Petri Nets (ICATPN 2001), Newcastle upon Tyne, UK, June 2001, LNCS 2075, pp. 121-143.

[11] A. Arnold, *Finite Transition Systems,* Prentice Hall, 1992.

[12] E. Fabre, *Convergence of the turbo algorithm for systems defined by local constraints,* Irisa research report no. PI 1510, april 2003.

[13] A. Benveniste, E. Fabre, S. Haar, C. Jard, *Diagnosis of asynchronous discrete event systems, a net unfolding approach,* Inria research report no. 4461, to appear in IEEE Trans. on Automatic Control, 2003.

[14] E. Fabre, *Compositional Models of Distributed and Asynchronous Dynamical Systems,* 41st Conf. on Detection and Control, Las Vegas, Dec. 2002, pp. 1-6.

[15] E. Fabre, V. Pigourier, *Monitoring distributed systems with distributed algorithms,* 41st Conf. on Detection and Control, Las Vegas, Dec. 2002, pp. 411-416.

[16] E. Fabre, *Distributed Diagnosis for Large Discrete Event Dynamic Systems,* in preparation.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

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