

Trellis Processes : A Compact Representation for Runs of Concurrent Systems

Eric Fabre

Received: 28 June 2005 / Accepted: 12 October 2006 /
Published online: 16 March 2007
© Springer Science + Business Media, LLC 2007

Abstract The unfolding of a concurrent system represents in a compact manner all possible runs of this system. Unfoldings are used in many applications, ranging from model-checking (offline) to failure diagnosis (on-line). Their factorization properties form the basis of modular/distributed algorithms to achieve these tasks. The trellis structure proposed here is an alternate and more compact representation for the trajectory set of a concurrent system. In this structure, time is unfolded, but not the conflict relations. Trellis nets are the generalization to concurrent systems of the usual notion of trellis for an automaton. As for unfoldings, factorization properties are evidenced on trellises, which makes these more compact structures a possible candidate for distributed model checking or diagnosis algorithms. As an example, we show how trellises can be used for diagnosis purposes in a distributed observation setting.

Keywords Concurrent system · Petri net · True concurrency semantics · Branching process · Unfolding · Trellis · Category theory · Product · Factorization · Diagnosis

1 Introduction

There is currently a strong research effort in the discrete event systems (DEDS) community to extend known results and methods to large modular or networked systems. The first issue that pops up when considering such systems is obviously the problem of size, and the state space explosion phenomenon. Many contributions have addressed this difficulty through modular or distributed approaches, that would

This work is supported by RNRT project SWAN, funded by the French Ministry of Research.

E. Fabre (✉)
Irisa/Inria, Campus de Beaulieu, 35042 Rennes cedex, France
e-mail: fabre@irisa.fr

properly scale up to large systems (Baroni et al. 1999; Boel and Jiroveanu 2004; Boel and van Schuppen 2002; Contant and Lafortune 2004; Debouk et al. 2000; Genc and Lafortune 2003; Sampath et al. 1995, 1996; Su 2004; Su et al. 2002; Yoo and Lafortune 2002). Among applications, let us mention diagnosis issues, reachability analysis, controllability checking, etc.

In this paper, we rather focus on another (related) explosion phenomenon, that takes place in the trajectory space. Large systems generally exhibit concurrency, which means that several non causally related events could occur in any order, or even simultaneously. The usual sequential semantics represents runs of a DEDS as a sequence of events, and thus considers different orderings of concurrent events as different runs, whence an explosion in the number of possible runs... In the early 80's (Nielsen et al. 1981; Winskel 1983, 1985), the unfolding technique was introduced as a convenient way to represent runs of concurrent systems, in the so-called “true concurrency semantics.” The idea is to represent trajectories of concurrent systems as partial orders of events, rather than sequences, in order to capture only true causality relations. This semantics has the advantage of reducing drastically the number of relevant runs, since useless interleavings of concurrent events are not taken into account. The unfolding of a system can then be viewed as a compact data structure to describe all its possible runs.

Unfoldings were intensively revisited in the 1990s for model checking purposes, in particular with the notion of finite complete prefix (Engelfriet 1991; Esparza 1994; Esparza et al. 1996, 2002; Esparza and Römer 1999; Esparza and Schröter 2000; Khomenko et al. 2003; McMillan 1992, 1993; Melzer and Römer 1997). The idea is to obtain a small finite structure describing all relevant runs of a concurrent system, in order to check properties like accessibility of a state, deadlock freeness, etc. While originally developed in the theoretical computer science community, recent contributions have promoted the use of unfoldings in the (close) field of discrete event systems. For example in (Giua and Xie 2004, 2005), where unfoldings are used to design controllers for Petri nets. Similarly, (Benveniste et al. 2003) has shown that unfoldings could be used to solve diagnosis problems, or more generally state estimation problems, in concurrent systems. In this last application, one is rather interested in a structure describing non-bounded behaviors. More specifically, the issue is the on-line recovery of runs of the system that can explain observations produced by that system.

This “on-line monitoring approach” has been pushed forward to solve the diagnosis problem for *distributed* systems, i.e. concurrent systems made of several components with localized interactions. Here, the keystone is the *factorization property* of unfoldings. This property states that the unfolding of a compound system can be expressed as the product of unfoldings of its components, for an appropriate notion of product. This factorization provides another tool to compress further the data structure representing all runs of a concurrent system, since the factorized form is generally more compact than the “developed” one. But its major advantage is to open the way to modular (or distributed) processings, by a suitable coordination of several on-line diagnosis procedures performed at the scale of single components (Fabre 2003a, 2004; Fabre et al. 2005). This has the advantage to address also the state explosion phenomenon. And clearly, by working at the scale of components, a globally intractable problem may be turned into a tractable one. The factorization property of unfoldings can be derived directly, provided one is not

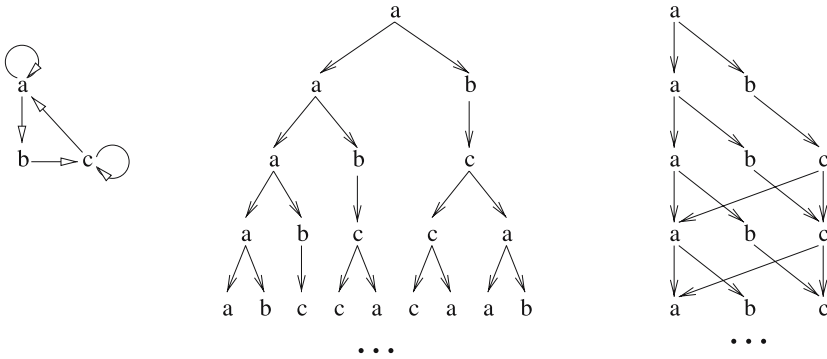


Fig. 1 A sequential machine (*left*) with *a* as initial state, its unfolding (*center*), and its trellis (*right*)

reluctant to heavy proofs. But the most simple and elegant derivation is probably due to (Winskel 1985), where it is expressed in the rather abstract (but powerful) framework of category theory.

Apparently though, unfoldings enjoy all the nice algebraic properties one could wish to deal with large concurrent systems. But a closer look reveals that they still suffer from a size problem, that could particularly penalize on-line monitoring algorithms. Specifically, the unfolding performs a *double expansion* of a system. First of all, *time* is unrolled; the unfolding is a partial order of events. But also *conflicts* are expanded: each time there is a choice between *n* possible events, *n* branches are created, *that will never meet each other again*, since conflicts are inherited in an unfolding. This justifies the name of *branching process* that is commonly used in the literature. As a consequence of this double expansion, the unfolding generally increases in width as one goes further in the direction of “increasing times.” This is striking in the case of an automaton, i.e. a sequential machine, for which the unfolding is simply the *tree* of all possible runs, viewed as sequences of events. In communities not concerned by concurrency aspects, another way of describing all possible runs of a system has prevailed up to now. Runs are represented on a *trellis*, which amounts to unfolding time but *not conflicts*, i.e. different (conflicting) runs ending in the same state at time *t* are merged, since they have the same future¹ (see Fig. 1). The resulting data structure representing runs grows along the time axis, but remains bounded in width.

The objective of this paper is thus to address the following natural questions. First of all, is it possible to extend this trellis representation to runs of *concurrent* systems? Specifically, one would like to capture graphically the concurrency of events, but abandon the infinite inheritance of conflicts and allow local merges of conflicting histories. The objective is of course to prevent an explosive number of possible histories to keep track of, for example in on-line monitoring algorithms.

¹To mention only one example, this approach is very popular in the digital communications community, to represent states of a convolutional encoder. It forms the basis of the Viterbi algorithm, a maximum likelihood estimation algorithm.

Secondly, does a factorization property exist for such structures? As explained above, this is a necessary ingredient to obtain distributed monitoring/diagnosis procedures. And finally, what are the relations between these *trellis processes* and unfoldings? Surprisingly, there exist simple answers to these questions, that we explore in this order. The construction of trellis processes intensively relies on a reasoning proposed in (Winskel 1985), that is used at several places in this paper. We recall it in the next section, and orient the reader to Appendix A for a quick overview of the necessary category theory material. We conclude the paper with an application of trellis processes to the diagnosis of concurrent systems, extending the unfolding based method of (Benveniste et al. 2003).

Notice that, independently, similar nets called “Merged Processes” were proposed in (Khomenko et al. 2005), for model-checking purposes. The latter are obtained indirectly by partly refolding a finite complete prefix of the unfolding. Since the objective is different in (Khomenko et al. 2005), the stress is rather put on interesting experimental results, comparing sizes of prefixes, than on algebraic properties of these objects. Merged Processes also slightly differ from the trellises developed in this paper (see Appendix D for a comparison).

Finally, let us mention that the present work has been reexpressed in the sequential semantics, instead of the partial order semantics (Fabre and Hadjicostis 2006).

2 Nets and unfoldings

Without losing too much generality, we consider concurrent systems expressed under the form of safe Petri nets. This section defines our notations for Petri nets (PN), occurrence nets (ON), etc. In Sections 3 and 4, we introduce trellis nets and study their properties. These sections rely on categorical constructions and results proved by Winskel in (Winskel 1985), that are briefly recalled here.

2.1 Category of nets

Net. A net is a 4-tuple $\mathcal{N} = (P, T, \rightarrow, P^0)$ where P, T are respectively the place and transition sets, and $\rightarrow \subseteq (P \times T) \cup (T \times P)$ is the flow relation.² In this paper, we consider only *safe* nets, i.e. nets for which places hold at most one token in any reachable marking. So markings identify with subsets of places, as the initial marking $P^0 \subseteq P$. As usually, $\bullet x$ and x^\bullet denotes pre- and post-sets of a given node $x \in P \cup T$. A run as a firable sequence $s = (t_1, t_2, \dots, t_n)$ of transitions, rooted at P^0 (for more details see Yoo and Lafortune 2002 or Reisig 1985). For technical reasons, specifically the use of recursive procedures in the sequel, we limit ourselves to safe nets where the number of marked places as well as the number of enabled transitions remain finite in any reachable marking.

At different places in the paper, mostly Sections 5 and 7, we use *labeled nets*, i.e. nets $\mathcal{N} = (P, T, \rightarrow, P^0, \lambda, \Lambda)$ provided with a label set Λ and a labeling function on transitions $\lambda : T \rightarrow \Lambda$.

²We assume each transition has at least one input place and one output place.

Morphism. To turn the collection of nets into a category (denoted $Nets$), we need the extra notion of morphism between nets. We use Winskel’s definition (Winskel 1984): A *morphism* ϕ from \mathcal{N}_1 to \mathcal{N}_2 , with $\mathcal{N}_i = (P_i, T_i, \rightarrow_i, P_i^0)$, is defined as a pair (ϕ_P, ϕ_T) where

1. ϕ_T is a partial function from T_1 to T_2 , and ϕ_P a relation between P_1 and P_2 ,
2. $P_2^0 = \phi_P(P_1^0)$ and $\forall p_2 \in P_2^0, \exists! p_1 \in P_1^0 : p_1 \phi_P p_2$,
3. if $p_1 \phi_P p_2$ then the restrictions $\phi_T : \bullet p_1 \rightarrow \bullet p_2$ and $\phi_T : p_1^\bullet \rightarrow p_2^\bullet$ are total functions,
4. if $t_2 = \phi_T(t_1)$ then the restrictions $\phi_P^{op} : \bullet t_2 \rightarrow \bullet t_1$ and $\phi_P^{op} : t_2^\bullet \rightarrow t_1^\bullet$ are total functions where ϕ_P^{op} denotes the opposite relation to ϕ_P .

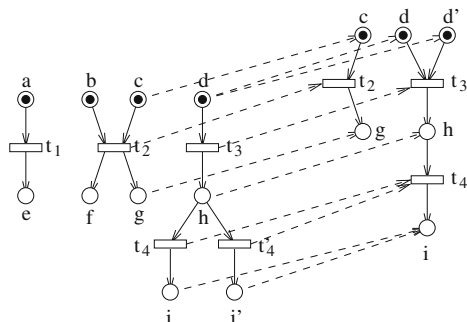
In the sequel, we will simply write ϕ for ϕ_P or ϕ_T . Thanks to 3–4, morphisms preserve the flow relation. This definition is actually designed to guarantee that a firable sequence of transitions in \mathcal{N}_1 is mapped to a firable sequence in \mathcal{N}_2 . Observe that morphisms are able to erase and to merge places and transitions, but they are also able to *duplicate* places. This last feature may look strange (see Fig. 2). It is actually motivated by the following fact: we need to define the product of concurrent systems (an extension of the usual synchronous product of automata). It is highly desirable that this product be the categorical product in $Nets$, which entails several nice algebraic properties. To this end, the family of morphisms must be able to duplicate places (see Appendices A.1 and B).

In the case of labeled nets, we must add the following requirements to the definition of a morphism $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$

5. ϕ_T preserves labels,
6. $Dom(\phi_T) = \lambda_1^{-1}(\Lambda_2)$, i.e. ϕ is defined on transitions carrying a shared label, and only on them,
7. $\Lambda_2 \subseteq \Lambda_1$.

Beyond point 5, quite natural, 6 is again necessary to obtain a categorical product on labeled nets (see below), and 7 ensures that the composition of morphisms remains a morphism, once 6 is introduced.

Fig. 2 A morphism between two nets (indicated by dashed arrows)



Product. Let $\mathcal{N}_1, \mathcal{N}_2$ be two nets, their product $\mathcal{N}_1 \times \mathcal{N}_2$ is defined as the triple $(\mathcal{N}, \psi_1, \psi_2)$ where $\mathcal{N} = (P, T, \rightarrow, P^0)$ is a net, $\psi_i : \mathcal{N} \rightarrow \mathcal{N}_i$ a morphism, and satisfying

1. $P = \{(p_1, \star) : p_1 \in P_1\} \cup \{(\star, p_2) : p_2 \in P_2\}$;
 $\psi_1(p_1, p_2) = p_1$ if $p_1 \neq \star$ and is undefined otherwise, and symmetrically for ψ_2 ,
2. $P^0 = \psi_1^{-1}(P_1^0) \cup \psi_2^{-1}(P_2^0)$,
3. $T = \{(t_1, \star) : t_1 \in T_1\} \cup \{(t_1, t_2) : t_1 \in T_1, t_2 \in T_2\} \cup \{(\star, t_2) : t_2 \in T_2\}$;
 $\psi_1(t_1, t_2) = t_1$ if $t_1 \neq \star$ and is undefined otherwise, and symmetrically for ψ_2 ,
4. \rightarrow is defined by $\star t = \star \psi_1(t) \cup \star \psi_2(t)$ and $t \star = \psi_1(t) \cup \psi_2(t) \star$, assuming $\star \psi_i(t) = \psi_i(t) \star = \emptyset$ if ψ_i is undefined on t .

In a product, each component preserves its places by the disjoint union in (1), and its transitions by (3), but synchronized transitions are also created, by merging transitions of \mathcal{N}_1 and \mathcal{N}_2 . This definition makes \times the categorical product of *Nets* (see A.1 for a definition, and Winskel 1997 for a proof; see also Vaandrager 1989).

For labeled nets, the product assumes the existence of a *synchronization algebra* (Winskel 1983). The latter specifies which pairs of transitions are legal, according to their labels. In this paper, we assume a simple rule for synchronizations: a transition t_1 of T_1 carrying a private label $\alpha_1 \in \Lambda_1 \setminus \Lambda_2$ doesn't synchronize, and yields (t_1, \star) in the product. Symmetrically for a private transition of T_2 . And any $t_1 \in T_1$ carrying a shared label $\alpha \in \Lambda_1 \cap \Lambda_2$ must synchronize with every $t_2 \in T_2$ carrying the same label. The product of labeled nets is thus the ordinary product of nets where pairs of transitions not satisfying synchronization rules are simply discarded.

2.2 Occurrence nets and unfoldings

Occurrence net. The net $\mathcal{O} = (C, E, \rightarrow, C^0)$ is an *occurrence net* (ON) iff it satisfies :

1. $C^0 = \{c \in C : \star c = \emptyset\}$,
2. the *causality* relation \rightarrow^* , irreflexive transitive closure of \rightarrow , is a partial order, and $\forall x \in C \cup E, [x] \triangleq \{x\} \cup \{y \in C \cup E : y \rightarrow^* x\}$ is finite,
3. $\forall c \in C, |\star c| \leq 1$,
4. the *conflict* relation $\#$ defined by the two properties below is irreflexive :
 - (a) $\forall e, e' \in E, [e \neq e', \star e \cap \star e' \neq \emptyset] \Rightarrow e \# e'$,
 - (b) $\forall x, x' \in C \cup E, [\exists e, e' \in E, e \# e', e \rightarrow^* x, e' \rightarrow^* x'] \Rightarrow x \# x'$.

The change in notations accounts for the usual terminology of *conditions*, instead of places, and *events*, instead of transitions. In an ON, “time” is unfolded, as indicated by (2). A condition can be marked by a unique event (3). By contrast, it may enable several events, which corresponds to a conflict situation. This creates a branching in the net, and the corresponding branches will never meet each other again (4). So conflicts are also unfolded.

ONs are generally introduced to represent runs of a net in the so-called *true concurrency semantics*. To do so, we need extra elements of terminology about ONs. Two nodes $x, x' \in C \cup E$ are said to be *concurrent*, denoted by $x \perp x'$, iff neither $x \# x'$ nor $x \rightarrow^* x'$ nor $x' \rightarrow^* x$ holds. A *co-set* is a set of pairwise concurrent conditions, and a *cut* is a maximal co-set for the inclusion. Finally, a *configuration* is a subset κ of $C \cup E$ which is conflict-free, causally closed (i.e. left-closed for \rightarrow^*), and such that

$\forall e \in E, e \in \kappa \Rightarrow e^\bullet \subseteq \kappa$. We denote by $\mathcal{K}_\mathcal{O}$ the set of configurations in \mathcal{O} . Observe that conditions form a co-set iff they appear as extremal nodes of a configuration.

Occurrence nets, equipped with morphisms of nets, form the subcategory Occ of $Nets$. Observe that in this category, morphisms can only erase (not create) causality or conflict relations between two nodes. Concurrency relations are preserved, and configurations are mapped to configurations.

Branching process. \mathcal{O} is said to be a *branching process* (BP) of net \mathcal{N} iff there exists a morphism $f : \mathcal{O} \rightarrow \mathcal{N}$ satisfying (Engelfriet 1991)

1. f is a total function on \mathcal{O} , also named a *folding* of \mathcal{O} ,
2. $\forall e, e' \in E, [*e = *e', f(e) = f(e')] \Rightarrow e = e'$.

Notice that being a total function, the restriction $f : C^0 \rightarrow P^0$ is a bijection, as well as $f : \bullet e \rightarrow \bullet f(e)$ and $f : e^\bullet \rightarrow f(e)^\bullet$ for every event $e \in E$. Formally, and following (Engelfriet 1991), a branching process is the pair (\mathcal{O}, f) . Here, with a slight abuse of notation, we often omit mentioning the folding f .

Consider a configuration κ in a branching process \mathcal{O} of \mathcal{N} (see Fig. 3). The events of κ are partially ordered by \rightarrow^* . Let us make a sequence e_1, e_2, \dots, e_N of these events by taking any linear extension of this partial order. Then $\phi(e_1), \phi(e_2), \dots, \phi(e_N)$ is a valid run of \mathcal{N} . Conversely, provided the BP \mathcal{O} is “large enough,” any run of \mathcal{N} can be recovered in that way from a configuration κ , and the latter is unique by the parsimony condition (2) above.

Unfolding. A *prefix* \mathcal{O}' of an occurrence net \mathcal{O} is defined as a sub-net of \mathcal{O} which is causally closed, contains the initial marking (or equivalently all minimal conditions), and such that $\forall e \in E, e \in \mathcal{O}'$ implies $e^\bullet \subseteq \mathcal{O}'$. So a configuration of \mathcal{O} is a conflict free prefix of \mathcal{O} . The prefix relation is denoted by $\mathcal{O}' \sqsubseteq \mathcal{O}$.

Given a net \mathcal{N} , there exists a maximal branching process of \mathcal{N} for the prefix relation. It is called the *unfolding* of \mathcal{N} . We denote it by $\mathcal{U}(\mathcal{N})$, or $\mathcal{U}_\mathcal{N}$ for short, and its corresponding folding by $f_\mathcal{N} : \mathcal{U}_\mathcal{N} \rightarrow \mathcal{N}$. Let \mathcal{O} be a branching process of \mathcal{N} , with folding $f : \mathcal{O} \rightarrow \mathcal{N}$, there exists a unique morphism $\psi : \mathcal{O} \rightarrow \mathcal{U}_\mathcal{N}$ such that

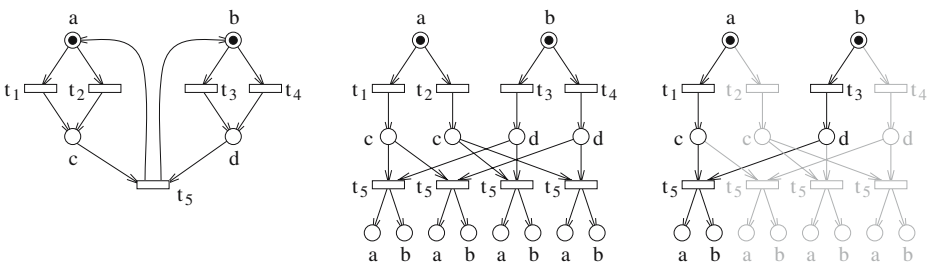


Fig. 3 A net \mathcal{N} (left), a branching process \mathcal{O} of that net (center) and a configuration κ in \mathcal{O} (right), corresponding to a run of \mathcal{N} . The folding of \mathcal{O} into \mathcal{N} is represented by transition and place names attached to events and conditions

$f = f_{\mathcal{N}} \circ \psi$. This obvious property on branching processes actually generalizes to any occurrence net \mathcal{O} :

$$\forall \mathcal{O} \in Occ, \forall \phi : \mathcal{O} \rightarrow \mathcal{N}, \exists ! \psi : \mathcal{O} \rightarrow \mathcal{U}_{\mathcal{N}}, \phi = f_{\mathcal{N}} \circ \psi \tag{1}$$

Eq. 1 is the *universal property* that characterizes the unfolding of \mathcal{N} , see (Winskel 1985).

For the net \mathcal{N} in Fig. 3, consider the central branching process \mathcal{O} . The unfolding of \mathcal{N} can be obtained by connecting a copy of \mathcal{O} to every pair of conditions (a, b) generated by the same event in \mathcal{O} , and so on repeatedly. This example illustrates that an unfolding generally grows in “width” (the conflict dimension) as one progresses in “length” (the time dimension).

There exists a simple and intuitive procedure to build (a prefix of) the unfolding of a net \mathcal{N} . We briefly mention it below,³ since it forms the basis of other constructions we use in the sequel.

Procedure 1

- Initialization :
 - Create $|P^0|$ conditions in C^0 , and define a bijection $f_{\mathcal{N}} : C^0 \rightarrow P^0$.
 - Set $C = C^0, E = \emptyset$ and $\rightarrow = \emptyset$.
- Recursion :
 - Let X be a co-set of C and $t \in T$ a transition of \mathcal{N} such that $\bullet t = f_{\mathcal{N}}(X)$.
 - If there doesn’t exist an event e in E with $\bullet e = X$ and $f_{\mathcal{N}}(e) = t$,
 - * create a new event e in E with $\bullet e = X$ and $f_{\mathcal{N}}(e) = t$,
 - * then create a subset Y of $|t^\bullet|$ new conditions in C , set $Y = e^\bullet$, and extend $f_{\mathcal{N}}$ to have the bijection $f_{\mathcal{N}} : Y \rightarrow t^\bullet$.

Product. A product can be defined directly in the category *Occ*, by the following recursive procedure. Let $\mathcal{O}_1, \mathcal{O}_2$ be two ONs, with $\mathcal{O}_i = (C_i, E_i, \rightarrow_i, C_i^0)$, their product $\mathcal{O} = (C, E, \rightarrow, C^0) = \mathcal{O}_1 \times \mathcal{O}_2$ and the associated morphisms $\psi_i : \mathcal{O} \rightarrow \mathcal{O}_i$ are given by:

Procedure 2

- Initialization :
 - Create $|C_1^0| + |C_2^0|$ conditions in C^0 , and define injective partial functions $\psi_i : C^0 \rightarrow C_i^0$ in such a way that they have disjoint domains.
 - Set $C = C^0, E = \emptyset$ and $\rightarrow = \emptyset$.
- Recursion :
 - Let X be a co-set of C , and $I \subseteq \{1, 2\}$ a non-empty index set; $\forall i \in I$, let e_i be an event of E_i such that $\psi_i(X) = \bullet e_i$.

³This recursive construction is explicit in (Winskel 1985) as well as in (Esparza and Römer 1999), where new events and places are named by a backward pointer technique. It also appears in the definition of “canonical branching processes” in (Engelfriet 1991). We avoid this heavy notation here, and use instead the less formal “create” primitive.

- If there doesn't exist an event $e \in E$ with $\bullet e = X$ and $\forall i \in I, \psi_i(e) = e_i$,
 - * create a new event e in E with $\bullet e = X$ and $\forall i \in I$, set $\psi_i(e) = e_i$,
 - * then create a subset Y of $\sum_{i \in I} |e_i^\bullet|$ new conditions in C , set $e^\bullet = Y$,
 - * extend the partial functions ψ_i to Y in order to have disjoint definition domains in Y and to satisfy $\psi_i : Y \rightarrow e_i^\bullet$ injective.⁴

The index set I takes value $\{1\}$ or $\{2\}$ to build in \mathcal{O} a “private event” of \mathcal{O}_1 or \mathcal{O}_2 , and takes value $\{1, 2\}$ to build a synchronized event. It can be proved directly that $\times_{\mathcal{O}}$ defined by Procedure 2 corresponds to the categorical product of Occ . But this result is derived in a more direct manner in the next sub-section.

In the particular case of labeled occurrence nets, the recursive construction of the product preserves the shape of Procedure 2, with the additional constraint that the event sets $\{e_i, i \in I\}$ selected for synchronization must satisfy the rules specified in a synchronization algebra. In the simple setting chosen in 2.1, only pairs (e_1, e_2) with $\lambda_1(e_1) = \lambda_2(e_2)$, or singletons e_i with $\lambda_i(e_i) \in \Lambda_i \setminus \Lambda_j$, can thus be selected in the first point of the recursion. Details are given in Appendix C (specialization of the product of trellis nets).

2.3 Relations between the two categories

Consider the inclusion functor $F = \subseteq$ of Occ into $Nets$. Following (Winskel 1985), we recall that $G = \mathcal{U}$, the unfolding operation on nets, forms the right adjoint of functor F . To do so, we use the construction of theorem 2-iv, in chap. IV-1 of (Mac Lane 1971), recalled in Appendix A.3, with notations $C=Occ, D=Nets, F = \subseteq$ and $G = \mathcal{U}$. The keystone of this construction is the universal property (1) of unfoldings.

Candidate co-unit. The starting point is to find a candidate co-unit $\epsilon : FG \rightarrow \mathbb{I}_D$ for this adjunction, where \mathbb{I}_D is the identity functor in $D=Nets$. The co-unit ϵ is defined by a morphism $\epsilon_{\mathcal{N}}$ for every $\mathcal{N} \in Nets, \epsilon_{\mathcal{N}} : \mathcal{U}_{\mathcal{N}} \rightarrow \mathcal{N}$. A straightforward choice is the folding $\epsilon_{\mathcal{N}} = f_{\mathcal{N}}$. For every \mathcal{N} in $Nets$, we must show that the pair $(G(\mathcal{N}), \epsilon_{\mathcal{N}}) = (\mathcal{U}_{\mathcal{N}}, f_{\mathcal{N}})$ is a universal arrow from F to \mathcal{N} (see Eq. 30 in Appendix A.3), which is exactly the universal property (1) of the unfolding $\mathcal{U}_{\mathcal{N}}$. So assumption (UP) holds.

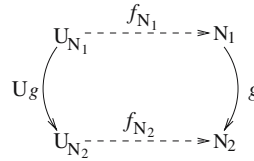
Unfolding as a functor. As a second step in this construction, we derive that $G = \mathcal{U}$, the unfolding operation on nets, can be turned into a functor from $Nets$ to Occ . To do so, we must explain how morphisms are transformed by \mathcal{U} . Let $g : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ be a morphism in $Nets, \mathcal{U}(g)$ (or \mathcal{U}_g) is defined as the unique morphism from $\mathcal{U}_{\mathcal{N}_1}$ to $\mathcal{U}_{\mathcal{N}_2}$ satisfying (Fig. 4)

$$g \circ f_{\mathcal{N}_1} = f_{\mathcal{N}_2} \circ \mathcal{U}(g) \tag{2}$$

Existence and unicity of $\mathcal{U}(g)$ are guaranteed by Eq. 1: take $\mathcal{O} = \mathcal{U}_{\mathcal{N}_1}, \mathcal{N} = \mathcal{N}_2$ and $\phi = g \circ f_{\mathcal{N}_1}$. It is then easy to show that \mathcal{U} commutes with the composition of morphisms, and maps identity to identity. Alternatively, it is instructive to build directly $\mathcal{U}(g)$, through a recursion that preserves Eq. 2 at each step:

⁴on its domain of definition

Fig. 4 Commutative diagram satisfied by the unfolding of a morphism



Procedure 3

- Initialization :
given bijections $f_{N_i} : C_i^0 \rightarrow P_i^0$, U_g is determined by Eq. 2 between C_1^0 and C_2^0 .
- Recursion :
 - let X_1 be a co-set of U_{N_1} where U_g is defined, so $g \circ f_{N_1}(X_1) = f_{N_2} \circ U_g(X_1)$, and let e_1 be an event of U_{N_1} such that $\bullet e_1 = X_1$,
 - then g is defined on $t_1 = f_{N_1}(e_1)$ (since it is defined on its pre-set), and $\bullet g(t_1) = g(\bullet t_1) = g \circ f_{N_1}(X_1) = f_{N_2} \circ U_g(X_1)$,
 - since $t_2 = g(t_1)$ is enabled by places of $f_{N_2} \circ U_g(X_1)$, there exists a unique event e_2 in U_{N_2} such that $f_{N_2}(e_2) = t_2$ and $\bullet e_2 = U_g(X_1)$, so one must define $U_g(e_1) = e_2$,
 - given bijections $f_{N_i} : e_i^\bullet \rightarrow t_i^\bullet$, extend U_g so that it coincides with g between e_1^\bullet and e_2^\bullet .

Adjunction. As a last step, we have to show that ϵ is a natural transformation of functor FG into \mathbb{I}_D . As FG performs an unfolding in *Nets*, this property coincides with Eq. 2 (Fig. 4). This is enough to prove the adjunction, i.e. to derive a one to one binatural correspondence between morphisms of $\text{Mor}(\text{Occ}, \text{Nets})$ and those of $\text{Mor}(\text{Occ}, \mathcal{U}(\text{Nets}))$, by Eq. 32 in Appendix A.3.

This particular form of adjunction, where the left adjoint is the inclusion functor, is called a *co-reflection* of the category of occurrence nets in the category of nets. As right adjoints preserve products, one has

$$\mathcal{U}(\mathcal{N}_1 \times_N \mathcal{N}_2) = \mathcal{U}(\mathcal{N}_1) \times_O \mathcal{U}(\mathcal{N}_2) \tag{3}$$

This would actually be sufficient to prove the existence of a product in *Occ* for occurrence nets that are unfoldings. But we can say more. Consider the *unit* of the adjunction, i.e. the natural transformation $\eta : \mathbb{I}_D \rightarrow GF$, given here by $\forall \mathcal{O} \in \text{Occ}, \eta_{\mathcal{O}} : \mathcal{O} \rightarrow \mathcal{U}\mathcal{O}$. η is obviously a natural equivalence since every ON is isomorphic to its own unfolding. So assumption (NE) of Appendix A.4 is satisfied, and the relation between \times_O and \times_N in Eq. 3 reaches all elements in *Occ*. This yields the following definition of \times_O

$$\mathcal{O}_1 \times_O \mathcal{O}_2 \cong \mathcal{U}(\mathcal{O}_1) \times_O \mathcal{U}(\mathcal{O}_2) = \mathcal{U}(\mathcal{O}_1 \times_N \mathcal{O}_2) \tag{4}$$

where \cong stands for “isomorphic to.” The fact that \times_O is a standard product in *Nets* followed by an unfolding gives exactly the recursive definition of the product mentioned at the end of Section 2.2 (observe that procedure 2 is indeed built on procedure 1, an unfolding algorithm).

2.4 Multi-clock nets

The notion of trellis that we develop in the sequel only applies to a (large) sub-class of safe Petri nets, that we define now. We show in particular that the categorical constructions developed in Section 2.3 adapt to this sub-class. The motivation for this light restriction is discussed later.

Multi-clock net. A multi-clock net (MCN) is a tuple $\mathcal{N}=(P, T, \rightarrow, P^0, \nu)$ satisfying;

1. (P, T, \rightarrow, P^0) is an ordinary safe net,
2. $\nu : P \rightarrow P^0$ defines a partition on places, and the restriction $\nu|_{P^0}$ is the identity; we denote by \bar{p} the equivalence class $\nu^{-1}(\nu(p))$ of a place p ,
3. $\forall t \in T, \nu$ is injective on $\bullet t$ and on t^\bullet , and $\nu(\bullet t) = \nu(t^\bullet)$,

This definition deserves some comments. Observe first that every transition satisfies $|\bullet t| = |t^\bullet|$. So the number of tokens remains constant in a MCN. Moreover, let $M \subseteq P$ be a reachable marking of \mathcal{N} , one has $\nu|_M$ is bijective. In other words, let $p \in P^0$, at any time there is exactly one place in $\nu^{-1}(p)$ holding a token. Secondly, consider the restriction of \mathcal{N} to places of $\bar{p}, p \in P$, that we denote by $\mathcal{N}_{|\bar{p}}$. Then $\mathcal{N}_{|\bar{p}}$ is an automaton, i.e. a Petri net where a single place holds a token at any time. Therefore, a multi-clock net can be regarded as a synchronous product of automata (we shall come back on this in Section 5, dedicated to labeled nets). By abuse of vocabulary, we will sometimes consider \bar{p} as the state variable of $\mathcal{N}_{|\bar{p}}$ (more rigorously, it corresponds to the value set of this state variable). Notice that $\mathcal{N}_{|\bar{p}}$, considered as a graph, generally has several connected components. Only one of them contains the place that is initially marked. Places and transitions belonging to the other connected components are unreachable, and can thus be discarded from \mathcal{N} without changing its dynamics.

Morphism of MCNs. Let $\mathcal{N}_1, \mathcal{N}_2$ be two MCNs, with $\mathcal{N}_i = (P_i, T_i, \rightarrow_i, P_i^0, \nu_i)$, we restrict ourselves to morphisms $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ that preserve the partitions ν_1, ν_2 , i.e. that satisfy

$$\forall p_1 \in P_1, \forall p_2 \in P_2, \quad p_1 \phi p_2 \Rightarrow \nu_1(p_1) \phi \nu_2(p_2) \tag{5}$$

The following lemma emphasizes that MCN morphisms actually erase or duplicate state variables as a whole.

Lemma 1 *Let $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ be a morphism of MCNs, with $\mathcal{N}_i = (P_i, T_i, \rightarrow_i, P_i^0, \nu_i)$, and assume that every sub-automaton $\mathcal{N}_{1|\bar{p}_1}$ of \mathcal{N}_1 has a single connected component.*

- a. *The inverse image by ϕ of a class of ν_2 is included in a class of ν_1 .*
- b. *Given a class of ν_1 , ϕ is either defined on all elements of this class, or on none of them.*
- c. *When a place $p_1 \in P_1$ is duplicated by ϕ , i.e. related to (elements of) several classes of ν_2 , each place in \bar{p}_1 is duplicated in the same way, i.e. related to the same classes.*

Proof

- (a) Assume $p_1 \phi p_2$ and $p'_1 \phi p'_2$ (where $p_i, p'_i \in P_i$), we must show that if $v_2(p_2) = v_2(p'_2)$ then $v_1(p_1) = v_1(p'_1)$. Since ϕ preserves partitions, we have $v_1(p_1) \phi v_2(p_2)$ and $v_1(p_1) \phi v_2(p'_2) = v_2(p_2)$ between P_1^0 and P_2^0 . But by definition of net morphisms, there is only one place in P_1^0 in relation with $v_2(p_2) \in P_2^0$. So $v_1(p_1) = v_1(p'_1)$. Observe that the converse result doesn't hold: One can have $p_1 \phi p_2, p'_1 \phi p'_2, v_1(p_1) = v_1(p'_1)$ and $v_2(p_2) \neq v_2(p'_2)$. This is the case in particular when ϕ duplicates some places.
- (b) Assume ϕ is defined on $p_1 \in P_1$, with $p_1 \phi p_2$. Let $t_1 \in p_1^\bullet$, by point 4 in the definition of net morphisms, ϕ is defined on t_1 and one has $\phi(t_1) = t_2 \in p_2^\bullet$. Let p'_2 be the unique place in t_2^\bullet such that $v_2(p'_2) = v_2(p_2)$. By point 5 in the definition of net morphisms, there exists a unique p'_1 in t_1^\bullet such that $p'_1 \phi p'_2$. Using (a), p'_1 is actually the only place in $t_1^\bullet \cap \bar{p}_1$. A similar argument shows that ϕ is also defined on $\bar{p}_1 \cap \bullet t_1$ for $t_1 \in \bullet p_1$. By recursion, and since $\mathcal{N}_{1|\bar{p}_1}$ has a single connected component, ϕ is defined on all places of \bar{p}_1 .
- (c) As a by-product of the proof for (b), if $p_1 \phi p_2$, and p'_1 satisfies $v_1(p'_1) = v_1(p_1)$, there exists p'_2 such that $v_2(p'_2) = v_2(p_2)$ and $p'_1 \phi p'_2$. □

We define \overline{Nets} as the category formed by multi-clock nets and their morphisms. Notice that it is always possible to turn a safe net \mathcal{N} into a multi-clock net with essentially the same behavior, by simply adding to each place of \mathcal{N} a complementary place. So \overline{Nets} “almost covers” all *Nets*.

Product. The product $\mathcal{N}_1 \times \mathcal{N}_2$ of two MCNs is defined as the standard product of nets where the resulting partition ν is simply the union of partitions ν_1 and ν_2 (recall that the product builds the disjoint union of places). It is straightforward to check that $\mathcal{N}_1 \times \mathcal{N}_2$ remains a MCN, and that morphisms $\psi_i : \mathcal{N}_1 \times \mathcal{N}_2 \rightarrow \mathcal{N}_i$ are morphisms of MCNs. To prove that \times actually defines the categorical product in \overline{Nets} , we must check that its universal property holds also in this sub-category. Let \mathcal{N} be a MCN, and let the $f_i : \mathcal{N} \rightarrow \mathcal{N}_i$ be morphisms of MCNs, $i = 1, 2$. There exists a unique arrow $\phi : \mathcal{N} \rightarrow \mathcal{N}_1 \times \mathcal{N}_2$ in \overline{Nets} such that $f_i = \psi_i \circ \phi$. From (Winskel 1997), we know that ϕ is given by $\phi(t) = (f_1(t), f_2(t))$, on transitions where at least one of the f_i is defined. On places, ϕ is given by

$$\begin{aligned}
 p \phi (p_1, \star) &\text{ iff } p f_1 p_1 \\
 p \phi (\star, p_2) &\text{ iff } p f_2 p_2
 \end{aligned}$$

Therefore ϕ is clearly an arrow of \overline{Nets} , and \times defines the categorical product of \overline{Nets} . This results extends trivially to labeled nets.

Multi-clock occurrence nets and unfoldings. A multi-clock occurrence net (MCON) is naturally a multi-clock net $\mathcal{O} = (C, E, \rightarrow, C^0, \nu)$ where (C, E, \rightarrow, C^0) is an occurrence net. They define the sub-category \overline{Occ} of \overline{Nets} . The unfolding of a MCN \mathcal{N} is clearly a MCON, and the associated folding $f_{\mathcal{N}} : \mathcal{U}(\mathcal{N}) \rightarrow \mathcal{N}$ is of course a morphism in \overline{Nets} . So we are in good shape to get a co-reflexion of \overline{Occ} into \overline{Nets} . This result can actually be derived exactly as before. One only has to check that the universal property (1) of MCN unfoldings involves MCN morphisms, from which (UP) holds. These simple verifications are left to the reader.

Terminology. In the remaining of the paper, we only deal with multi-clock nets. Therefore the term “multi-clock” will not appear systematically.

3 Trellis nets

This section contains the main contribution of this paper: we introduce trellis nets and study some of their properties. We have insisted in the previous section on known results relating nets to occurrence nets because we will now follow exactly the same track for trellis nets. The next section will then focus on relations between the three notions: nets, trellis nets, and occurrence nets.

3.1 Definition

Pre-trellis net. The (multi-clock) net $\mathcal{T} = (C, E, \rightarrow, C^0, \nu)$ is a *pre-trellis net* iff it satisfies:

1. $C^0 = \{c \in C : \bullet c = \emptyset\}$,
2. for every $c \in C^0$, the automaton $\mathcal{T}_{\bar{c}}$ has no circuit (i.e. its flow relation defines a partial order).

The definition of pre-trellis nets is much less restrictive than the definition of occurrence nets. Specifically, point 1 is preserved, point 2 is weakened since \rightarrow^* is not any more required to define a partial order, and we have abandoned points 3 and 4: conflicting branches are now allowed to merge on conditions.

As an oriented graph, and by contrast with occurrence nets, a pre-trellis net is not necessarily a partial order. Figure 5 gives a counter-example of a pre-trellis net containing a circuit. However, one has the following property:

Lemma 2 *No run of a pre-trellis net \mathcal{T} can have a loop, i.e. can fill twice the same place. As a consequence, the restriction $\mathcal{T}_{\bar{s}}$ of \mathcal{T} to (nodes involved in) any run s defines a partial order of nodes.*

Proof Assume place c of net \mathcal{T} is filled twice by some sequence s of transitions of \mathcal{T} . The canonical projection of s on transitions of $\mathcal{T}_{\bar{c}}$ is of course a valid run of this automaton. And this sequence fills twice place c , which contradicts the fact that $\mathcal{T}_{\bar{c}}$ has no circuit. □

Fig. 5 A pre-trellis net containing a circuit (thick arrows)

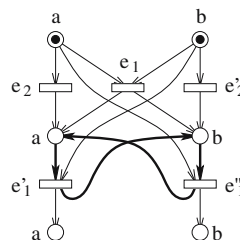
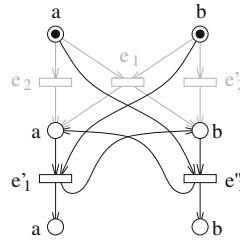


Fig. 6 In the net of Fig. 5, a subset of nodes satisfying the first four requirements of a configuration, but failing on the last one



Configuration, trellis net. In an occurrence net, every event belongs at least to one configuration, and so is reachable. This is not guaranteed anymore in a pre-trellis net (see T_1 in Fig. 7), so we must refine our definition. We define a *configuration* κ of a pre-trellis net $\mathcal{T} = (C, E, \rightarrow, C^0, \nu)$ as a sub-net of \mathcal{T} satisfying

1. $C^0 \subseteq \kappa$,
2. $\forall e \in E \cap \kappa, \bullet e \subseteq \kappa$ and $e \bullet \subseteq \kappa$: each event has all its causes and consequences,
3. $\forall c \in C \cap \kappa, |\bullet c \cap \kappa| = 1$ or $c \in C^0$: each condition is either minimal or has one of its possible causes,
4. $\forall c \in C \cap \kappa, |c \bullet \cap \kappa| \leq 1$: each condition triggers at most one event,
5. the restriction of \mathcal{T} to nodes of κ is a partial order.

This definition is close to the one introduced for ONs, apart from the fact that $|\bullet c| \leq 1$ is not automatic anymore in a pre-trellis net. So one must not only solve conflicts forward (point 4) but also backwards (point 3), to get a valid conflict-free ON. And the requirement that a configuration is “causally closed” is now spread on 2, 3 and 4. The last point is suggested by lemma 2, and is indeed necessary since points 1–alone do not guarantee this property (see a counter-example in Fig. 6). With the above definition, it is straightforward to check that a sequence s is a run of \mathcal{T} iff it corresponds to a linear extension of some configuration κ of \mathcal{T} . And so an event of a pre-trellis net is reachable iff it belongs to a configuration. We thus define a *trellis net* (TN) as a pre-trellis net where each event belongs at least to one finite configuration (see Fig. 7 for examples).

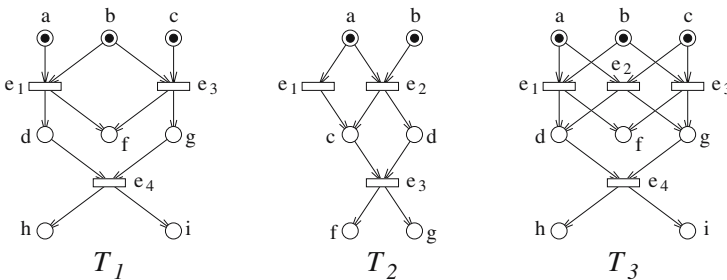


Fig. 7 T_1 is a pre-trellis net but not a trellis net: event e_4 is unreachable. The other nets are trellis nets: all events are reachable. In T_2 , e_1 and e_3 are not causally related... but in conflict! T_3 displays a non-binary conflict: $\{d, f\}$, $\{f, g\}$ and $\{d, g\}$ are all pairs of concurrent conditions, but the triple $\{d, f, g\}$ appears in no run. Removing e_2 in T_3 doesn't yield a valid prefix: we are back to T_1 which is not a trellis net

Concurrency and conflict. From the definitions above, one sees that both ONs and TNs are graphical structures encoding families of configurations in different ways, just like Fig. 1 represents the same sets of runs in different ways for an automaton. TNs offer the advantage of being more compact... at the expense of a more complex display of configurations. In particular, the familiar causality, conflict and concurrency relations on events do not have any more a simple graphical translation (see \mathcal{T}_2 in Fig. 7). This is due to the fact that, in a TP, an event (or a condition) generally appears on top of several histories. This phenomenon introduces a strong contrast with ONs, where a node belongs to a unique minimal configuration (its causal closure). As a consequence, concurrency and causality are now “context dependent”: two events may be concurrent in one configuration, and appear as causally related in another (Fig. 8).

It is important to define co-sets in a trellis net, i.e. to determine conditions that can be used at the same time to connect one more event to the structure. To define this extended notion of concurrency, we thus have to abstract the context. Let x_1, x_2, \dots, x_n be n nodes of \mathcal{T} , they are *concurrent* in \mathcal{T} , denoted by $\perp(x_1, x_2, \dots, x_n)$, iff there exists a configuration κ where they appear as concurrent nodes. On the example of Fig. 8 (left), e_3 and e_4 are thus declared concurrent for this extended notion. The notion of co-set (of conditions) derives from this definition. Observe that on a TN, concurrency can no longer be derived from pairwise relations, by contrast with ONs (see \mathcal{T}_3 in Fig. 7). In the same way, an extended notion of conflict can be defined as follows: x_1, x_2, \dots, x_n are in conflict, $\#(x_1, x_2, \dots, x_n)$, iff there is no configuration containing all of them (for example $\#(e_5, e_6)$ in Fig. 8). Again, $\#$ cannot be derived from pairwise relations, i.e. conflict is not binary in TNs.

Prefix. Prefixes are less easy to define graphically for TNs than for ONs. Let \mathcal{T} be a TN, \mathcal{T}' is a *prefix* of \mathcal{T} ($\mathcal{T}' \sqsubseteq \mathcal{T}$) iff

1. \mathcal{T}' is a sub-net of \mathcal{T} ,
2. $\{c \text{ condition of } \mathcal{T}, \bullet c = \emptyset\} = \{c' \text{ condition of } \mathcal{T}', \bullet c' = \emptyset\}$
3. $\forall e \text{ event of } \mathcal{T}, e \in \mathcal{T}' \Rightarrow [\bullet e \subseteq \mathcal{T}' \text{ and } e^\bullet \subseteq \mathcal{T}']$,
4. \mathcal{T}' is a trellis net.

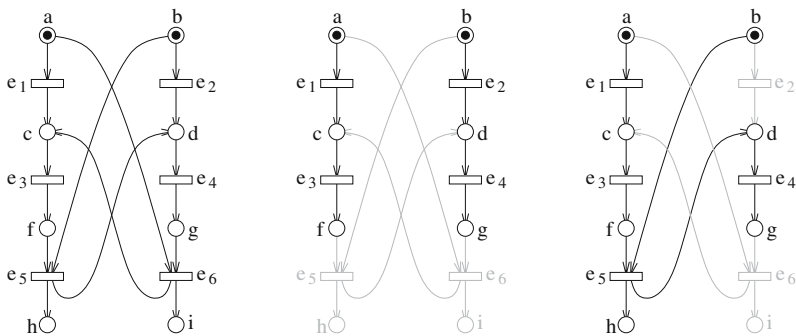


Fig. 8 On this trellis net (left), events e_3 and e_4 appear in several configurations. They can be concurrent in one of them (center) and causally related in another (right)

The last requirement imposes that every event in the sub-net \mathcal{T}' remains reachable. To illustrate its necessity, consider \mathcal{T}_3 in Fig. 7: if e_2 is removed, points 1–3 are satisfied, but e_4 becomes unreachable. Of course, \sqsubseteq on TNs extends the relation \sqsubseteq on ONs. Notice also that $\mathcal{T}' \sqsubseteq \mathcal{T}$ implies the existence of an injective morphism $\phi : \mathcal{T}' \rightarrow \mathcal{T}$ (which means here that ϕ is a total function).

Height function. The definition of trellis nets now allows to merge conflicting conditions produced by different configurations. However, this leaves a large amount of flexibility to represent a given set of configurations. If one wishes to get a universal object to represent this set, some kind of guideline is necessary to indicate where merges should be performed.

We define a *string* as a configuration $\sigma = (C, E, \rightarrow, C^0, \nu)$ in \overline{OCC} where $|C^0| = 1$. So σ has a single class and thus corresponds to a sequence alternating conditions and events. The *height* $H_\sigma(c)$ of condition c in σ is given by

$$H_\sigma(c) = |\{c' \in C, c' \rightarrow^* c\}| \tag{6}$$

In a general configuration κ , we set $H_\kappa(c) \triangleq H_\sigma(c)$ where $\sigma = \kappa|_{\bar{c}}$, so

$$H_\kappa(c) = |\{c' \in C, \nu(c') = \nu(c), c' \rightarrow^* c\}| \tag{7}$$

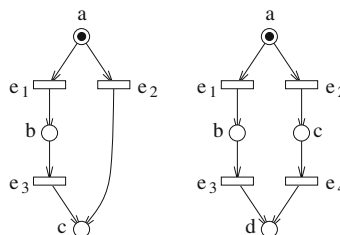
A trellis net \mathcal{T} *complies with* H , or is *correctly folded* for H , iff for every condition c of \mathcal{T} and every pair of strings σ, σ' containing c in $\mathcal{T}_{\bar{c}}$, one has $H_\sigma(c) = H_{\sigma'}(c)$. We denote this common value by $H_{\mathcal{T}}(c)$ or simply $H(c)$ when there is no ambiguity. Figure 9 illustrates this property.

Lemma 3 *The trellis net \mathcal{T} is correctly folded for H iff $\forall c \in C, \forall \kappa, \kappa'$ configurations of \mathcal{T} containing $c, H_\kappa(c) = H_{\kappa'}(c)$.*

Proof This condition is obviously necessary, so we only have to show it is sufficient. Assume \mathcal{T} is not well folded at c , i.e. there exist strings σ_1, σ_2 in $\mathcal{T}_{\bar{c}}$ such that $H_{\sigma_1}(c) \neq H_{\sigma_2}(c)$. WLOG we can assume that \mathcal{T} is correctly folded at all conditions below c in $\mathcal{T}_{\bar{c}}$. Let $c_1 \rightarrow e_1 \rightarrow c$ in σ_1 and $c_2 \rightarrow e_2 \rightarrow c$ in σ_2 . Since e_i is reachable in \mathcal{T} , there exists a configuration κ_i containing c_i, e_i and c in \mathcal{T} . One has $H_{\sigma_i}(c_i) = H_{\kappa_i}(c_i)$, because \mathcal{T} is correctly folded at c_i . Adding 1 to both sides of the equality, we get $H_{\sigma_i}(c) = H_{\kappa_i}(c)$. So $H_{\sigma_1}(c) \neq H_{\sigma_2}(c)$ entails $H_{\kappa_1}(c) \neq H_{\kappa_2}(c)$. \square

In the sequel, we only consider H -compliant TNs. The latter, associated to the usual notion of morphism (of MCNs), form the category \overline{Tr} . So we have three

Fig. 9 Two trellis nets; the left one is not H -compliant, the other one is



nested categories: $\overline{Occ} \subset \overline{Tr} \subset \overline{Nets}$. Notice that the notion of height function is generalized in Section 6.

3.2 Trellis processes and time unfoldings

Trellis process. Following ideas developed for occurrence nets, trellis nets can be used to represent runs of a given net. The trellis net $\mathcal{T} = (C, E, \rightarrow, C^0, \nu)$ is a *trellis process* (TP) of net \mathcal{N} iff there exists a morphism $\phi : \mathcal{T} \rightarrow \mathcal{N}$ satisfying

1. ϕ is a folding of \mathcal{T} (i.e. a total function on \mathcal{T}),
2. $\forall e, e' \in E, [\bullet e = \bullet e', \phi(e) = \phi(e')] \Rightarrow e = e'$,
3. $\forall c, c' \in C, [H(c) = H(c'), \phi(c) = \phi(c')] \Rightarrow c = c'$.

Notice that being a total function, the restriction $\phi : C^0 \rightarrow P^0$ is a bijection, as well as $\phi : \bullet e \rightarrow \bullet \phi(e)$ and $\phi : e^\bullet \rightarrow \phi(e)^\bullet$ for every event $e \in E$. By contrast with branching processes, this definition contains a double parsimony condition : by 2, redundant branchings are eliminated, and by 3, merges are imposed. Again, a trellis process is formally the pair (\mathcal{T}, ϕ) , but we will often omit mentioning ϕ .

By definition, ϕ is a folding of \mathcal{T} into \mathcal{N} , so every configuration κ of \mathcal{T} represents a run of \mathcal{N} in the true concurrency semantics, and has a counterpart in $\mathcal{U}_{\mathcal{N}}$. So a trellis process of \mathcal{N} corresponds to a collection of runs of \mathcal{N} . Conversely, a run of \mathcal{N} is represented by at most one configuration in \mathcal{T} : If κ_1 and κ_2 are isomorphic and folded to \mathcal{N} in the same way (up to this isomorphism), then they are identical. Indeed, one has $H_{\kappa_1} = H_{\kappa_2}$ which shows that conditions are identical (point 3), from which events are also identical (point 2).

Let us consider the restriction of a TP \mathcal{T} of \mathcal{N} to nodes with height lower than $h \in \mathbb{N}$. The two remarks above indicate that this restriction is a finite TN, since the restriction of $\mathcal{U}_{\mathcal{N}}$ to nodes lower than h is itself finite. This property opens the way to recursive reasonings on trellis processes.

Time unfolding of a net. As for branching processes, one can easily build trellis processes of a net $\mathcal{N} = (P, T, \rightarrow, P^0, \mu)$ with a simple recursion, yielding both $\mathcal{T} = (C, E, \rightarrow, C^0, \nu)$ and the folding $\phi : \mathcal{T} \rightarrow \mathcal{N}$.

Procedure 4

- Initialization :
 - Create $|P^0|$ conditions in C^0 , and define a bijection $\phi : C^0 \rightarrow P^0$.
 - Set $C = C^0, E = \emptyset, \rightarrow = \emptyset$ and $\nu = Id$.
- Recursion :
 - Let X be a co-set of C and $t \in T$ a transition of \mathcal{N} such that $\bullet t = \phi(X)$.
 - If there doesn't exist an event $e \in E$ with $\bullet e = X$ and $\phi(e) = t$,
 - * create a new event e in E with $\bullet e = X$ and $\phi(e) = t$,
 - * create a subset Y of $|t^\bullet|$ new conditions in C , with $Y = e^\bullet$,
 extend ϕ to have $\phi : Y \rightarrow t^\bullet$ bijective,
 and extend the partition ν to preserve $\nu = \phi^{-1} \circ \mu \circ \phi$,
 - * then, for every $c \in Y$,
 if $\exists c' \in C, \phi(c') = \phi(c)$ and $H(c') = H(c)$ then merge c and c' .

Procedure 4 is a variation of procedure 1 (that builds BPs of a net \mathcal{N}). It essentially differs by the last steps, where the newly created conditions are merged to existing ones as soon as they represent the same place and have the same height. Observe in particular that the partitioning defined by ν on conditions of \mathcal{T} is a direct image of the partitioning given by μ in \mathcal{N} (this choice is imposed by the necessity for ϕ to be a MCN morphism).

Procedure 4 generates a trellis process of \mathcal{N} , by construction, and conversely, it is easily checked that *any* TP of \mathcal{N} can be reached in that way. The proof relies on two facts: First, as soon as a subset of conditions X is declared as forming a co-set, this property remains true forever, whatever the next steps of the recursion are. And secondly, every event e in a TP is reachable, i.e. belongs to a configuration. So the connection of an event e to its co-set X of pre-conditions can't be prevented in the procedure, whatever the ordering of operations.

As for unfoldings, we would like to prove the existence of a unique stationary point of procedure 4. We start by defining the union of TPs. Let $\mathcal{T}_1, \mathcal{T}_2$ be TPs of \mathcal{N} , with respective foldings ϕ_i . The union $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, its folding ϕ and morphisms $\psi_i : \mathcal{T}_i \rightarrow \mathcal{T}$ are defined by the following

Procedure 5

- Initialization :
 - Create $|P^0|$ conditions in C^0 , define a bijection $\phi : C^0 \rightarrow P^0$, and bijections $\psi_i : C^0 \rightarrow C_i^0$ satisfying $\phi_i = \phi \circ \psi_i, i \in \{1, 2\}$.
 - Set $C = C^0, E = \emptyset, \rightarrow = \emptyset$ and $\nu = Id$.
- Recursion :
 - for $i \in \{1, 2\}$ and $e_i \in E_i$ such that ψ_i is defined on $\bullet e_i$ but not on e_i
 - if $\exists e \in E$ such that $\bullet e = \psi_i(\bullet e_i)$ and $\phi(e) = \phi_i(e_i)$, set $\psi_i(e_i) = e$, and define $\psi_i : e_i^\bullet \rightarrow e^\bullet$ in order to preserve $\phi_i = \phi \circ \psi_i$,
 - otherwise create a new $e \in E$ with $\bullet e = \psi_i(\bullet e_i), \phi(e) = \phi_i(e_i), \psi_i(e_i) = e$,
 - then, for every condition $c_i \in e_i^\bullet$
 - * if $\exists c \in C, H(c) = H(c_i)$ and $\phi(c) = \phi_i(c_i)$, then add e in $\bullet c$ and set $\psi_i(c_i) = c$,
 - * otherwise create a new $c \in C$ with $\bullet c = e, \phi(c) = \phi_i(c_i), \psi_i(c_i) = c$, and set $\nu(c) = \psi_i \circ \nu_i(c_i)$ (in order to preserve $\nu \circ \psi_i = \psi_i \circ \nu_i$).

Clearly, procedure 5 yields a TP of \mathcal{N} , a folding $\phi : \mathcal{T} \rightarrow \mathcal{N}$, and morphisms $\psi_i : \mathcal{T}_i \rightarrow \mathcal{T}$. The latter are injective total functions, which proves $\mathcal{T}_i \sqsubseteq \mathcal{T}$, and \mathcal{T} is the smallest TP of \mathcal{N} having $\mathcal{T}_1, \mathcal{T}_2$ as prefixes. Using this property, one has that a TP of \mathcal{N} is isomorphic to the union of its configurations, or conversely, that a set of configurations determines a unique TP. Notice also that there exist unique morphisms $\psi_i : \mathcal{T}_i \rightarrow \mathcal{T}_1 \cup \mathcal{T}_2$ satisfying $\phi_i = \phi \circ \psi_i$: precisely the ones obtained by procedure 5.

Procedure 5 generalizes without difficulty to the union of an arbitrary number of trellis processes⁵ of \mathcal{N} .

⁵For branching processes, Engelfriet (1991) proceeds by isolating a canonical representent for a class of isomorphic branching processes, i.e. BPs formed by the same occurrence net but different foldings,

The intersection $\mathcal{T}_1 \cap \mathcal{T}_2$ can be defined in a similar manner, by a simple modification of procedure 5, or by taking the union of configurations in \mathcal{K}

$$\mathcal{K} = \{\kappa_1 \in \mathcal{K}_{\mathcal{T}_1} : \exists \kappa_2 \in \mathcal{K}_{\mathcal{T}_2}, \exists \phi : \kappa_1 \rightarrow \kappa_2 \text{ isomorphism, } \phi_1 = \phi_2 \circ \phi\} \quad (8)$$

Theorem 1 *Let \mathcal{N} be a multi-clock net, there exists a unique maximal trellis process of \mathcal{N} for the prefix relation. We call it the trellis of \mathcal{N} or the time unfolding of \mathcal{N} , and denote it by $\mathcal{U}'_{\mathcal{N}}$, with corresponding folding $f'_{\mathcal{N}} : \mathcal{U}'_{\mathcal{N}} \rightarrow \mathcal{N}$.*

$\mathcal{U}(\mathcal{N})$, the unfolding of \mathcal{N} , and $\mathcal{U}^t(\mathcal{N})$, the time-unfolding of \mathcal{N} , are different encodings for the set of trajectories of \mathcal{N} . In particular, they have the same configuration set.

Proof We have proved that the collection of TPs of \mathcal{N} is stable by arbitrary union, and that $\mathcal{T} \sqsubseteq \mathcal{T} \cup \mathcal{T}'$. So there exists a unique maximal TP of \mathcal{N} for the prefix relation. Moreover, since every finite TP of \mathcal{N} is reachable by procedure 4, this procedure converges to $\mathcal{U}'_{\mathcal{N}}$, its unique stationary point.

Given a net \mathcal{N} , consider \mathcal{K} , the set of all configurations in $\mathcal{U}_{\mathcal{N}}$. Every configuration of \mathcal{K} is a TP of \mathcal{N} , and every configuration in a TP of \mathcal{N} is in \mathcal{K} . So $\mathcal{U}'_{\mathcal{N}}$, the trellis of \mathcal{N} , is the union in the sense of trellis processes of configurations of \mathcal{K} , just like the unfolding $\mathcal{U}_{\mathcal{N}}$ can be viewed as the union in the sense of branching processes of all configurations of \mathcal{K} . As a consequence, $\mathcal{U}_{\mathcal{N}}$ and $\mathcal{U}'_{\mathcal{N}}$ describe the same set of configurations. $\mathcal{U}'_{\mathcal{N}}$ can actually be recovered from $\mathcal{U}_{\mathcal{N}}$ by a folding operation: merge conditions c, c' such that $f_{\mathcal{N}}(c) = f_{\mathcal{N}}(c')$ and $H(c) = H(c')$, in order to ensure point 3 in the definition of a TP, then merge redundant events violating the parsimony condition (point 2). We shall express this relation more formally in Section 4. □

Theorem 2 (Universal property of $\mathcal{U}'_{\mathcal{N}}$) *Let $\mathcal{N} \in \overline{\text{Nets}}$, for every trellis net \mathcal{T} in $\overline{\text{Tr}}$ and morphism $\phi : \mathcal{T} \rightarrow \mathcal{N}$, there exists a unique morphism $\psi : \mathcal{T} \rightarrow \mathcal{U}'_{\mathcal{N}}$ such that $\phi = f'_{\mathcal{N}} \circ \psi$.*

Proof We proceed in several steps. If ψ exists, ϕ and ψ necessarily have identical domains of definition. So let $\mathcal{T}' = \mathcal{T}_{\text{dom}(\phi)}$, and let π be the canonical projection from \mathcal{T} to \mathcal{T}' . By lemma 1, \mathcal{T}' is the restriction of \mathcal{T} to a subset of its state variables, so by definition of H , \mathcal{T}' remains a correctly folded trellis net. There exists a unique $\phi' : \mathcal{T}' \rightarrow \mathcal{N}$ such that $\phi = \phi' \circ \pi$, and if ψ exists, there exists as well a unique $\psi' : \mathcal{T}' \rightarrow \mathcal{U}'_{\mathcal{N}}$ such that $\psi = \psi' \circ \pi$. The relation $\phi = f'_{\mathcal{N}} \circ \psi$ entails $\phi' \circ \pi = f'_{\mathcal{N}} \circ \psi' \circ \pi$, and since π is obviously an epi-morphism, we get $\phi' = f'_{\mathcal{N}} \circ \psi'$. So we can simplify the problem and assume that ϕ is defined everywhere on \mathcal{T} .

If \mathcal{T} is a trellis process of \mathcal{N} , the existence and uniqueness of ψ derives directly from procedure 5 applied to $\mathcal{T}_1 = \mathcal{T}$ and $\mathcal{T}_2 = \mathcal{U}'_{\mathcal{N}}$. So let us examine the remaining case: \mathcal{T} is a general trellis net, folded by ϕ onto \mathcal{N} , but \mathcal{T} may not be maximally folded, nor satisfy the parsimony criterion (points 2 and 3 in the definition of a TP). We nevertheless proceed as for a trellis process and build the morphism $\psi : \mathcal{T} \rightarrow$

while still representing the same configurations. The union is then defined on these canonical BPs. Here, we circumvent this construction by handling equivalence classes “as a whole.” Adding an equivalent TP in the union doesn’t change the result.

$\mathcal{U}_{\mathcal{N}}^t$ recursively on events of \mathcal{T} (recall that events of \mathcal{T} are all reachable). We adopt the notation $\mathcal{U}_{\mathcal{N}}^t = (C', E', \rightarrow', C'^0, v')$.

As a start point, we define ψ between C^0 and C'^0 . Since the restriction $f_{\mathcal{N}}^t : C'^0 \rightarrow P^0$ is bijective, we must take

$$\forall c \in C^0, \forall c' \in C'^0, \quad c \psi c' \Leftrightarrow c \phi f_{\mathcal{N}}^t(c') \tag{9}$$

We design the following steps in order to ensure that ψ , *restricted to its current domain of definition*, remains a morphism (of MCNs) and satisfies $\phi = f_{\mathcal{N}}^t \circ \psi$. By Eq. 9, this is obviously true at the start point. We show that this property can be progressively extended to cover all \mathcal{T} , and thus satisfy point 3 in the definition of a morphism.

Consider an event e of \mathcal{T} , and assume ψ is defined on the co-set $X = \bullet e$ but not at e . ψ is a morphism (on its domain of definition) so $\psi(X)^6$ is a co-set of $\mathcal{U}_{\mathcal{N}}^t$. Since $\bullet\phi(e) = \phi(X)$ in \mathcal{N} , there exists a unique event e' in $\mathcal{U}_{\mathcal{N}}^t$ such that $\bullet e' = \psi(X)$ and $f_{\mathcal{N}}^t(e') = \phi(e)$. To preserve $\phi = f_{\mathcal{N}}^t \circ \psi$, we must extend ψ by $\psi(e) = e'$, and since $f_{\mathcal{N}}^t : e \bullet \rightarrow \phi(e) \bullet$ is bijective, the extension of ψ between $e \bullet$ and $e' \bullet$ is also imposed, as in Eq. 9. By construction, this extended ψ satisfies points 1, 2 and 4 of the definition of net morphisms. And point 3 is obtained when we restrict ψ to its domain of definition. Finally, as a relation on conditions and relying on Eq. 9, ψ clearly preserves partitions v and v' .

Since all events of \mathcal{T} are reachable, ψ can be extended to finally cover all \mathcal{T} . This proves both existence and uniqueness of ψ . □

3.3 Co-reflection of \overline{Tr} into \overline{Nets}

To match again notations of Appendix A, define categories $C = \overline{Tr}$ and $D = \overline{Nets}$, and take for functor $F : C \rightarrow D$ the inclusion functor. We prove that the time unfolding operation on nets can be turned into a functor $G = \mathcal{U}^t : D \rightarrow C$, i.e. $\mathcal{U}^t : \overline{Nets} \rightarrow \overline{Tr}$, which is the right adjoint of F . We proceed as in Section 2.3: the derivation of the adjunction is based on the universal property stated in theorem 2.

Candidate co-unit. As for unfoldings, we look first for a candidate co-unit $\epsilon : FG \rightarrow \mathbb{I}_D$ for this adjunction. The co-unit ϵ is defined as a collection of morphisms $\epsilon_{\mathcal{N}} : \mathcal{U}_{\mathcal{N}}^t \rightarrow \mathcal{N}$, for every $\mathcal{N} \in \overline{Nets}$. An obvious choice is the folding $\epsilon_{\mathcal{N}} = f_{\mathcal{N}}^t$. For every $\mathcal{N} \in \overline{Nets}$, we must show that the pair $(G(\mathcal{N}), \epsilon_{\mathcal{N}}) = (\mathcal{U}_{\mathcal{N}}^t, f_{\mathcal{N}}^t)$ is a universal arrow from F to \mathcal{N} (see Eq. 30 in Appendix A.3), which is exactly the universal property of the time unfolding $\mathcal{U}_{\mathcal{N}}^t$ expressed in theorem 2. So assumption (UP) holds.

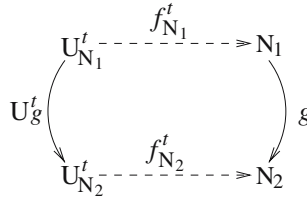
Time unfolding as a functor. Let $g : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ be a morphism in \overline{Nets} , $\mathcal{U}^t(g)$ can be defined as the unique morphism in \overline{Tr} satisfying (see Fig. 10):

$$g \circ f_{\mathcal{N}_1}^t = f_{\mathcal{N}_2}^t \circ \mathcal{U}^t(g) \tag{10}$$

Existence and unicity of $\mathcal{U}^t(g)$ are guaranteed by theorem 2, and Eq. 10 is sufficient to prove that \mathcal{U}^t is indeed a functor. Moreover, in practice, it is possible to define

⁶By abuse of notation, we write $\psi(X)$ the set $\{c' \in C' : \exists c \in X, c \psi c'\}$.

Fig. 10 Commutative diagram satisfied by the time unfolding of a morphism



$U^t(g)$ recursively with procedure 3, where U_g is replaced by U_g^t , U_{N_i} by $U_{N_i}^t$, the folding f_{N_i} by $f_{N_i}^t$, and where the invariant (2) is replaced by Eq. 10.

Co-reflection. Finally, with a now classical argument, Eq. 10 reveals that ϵ is actually a natural transformation of functor FG into \mathbb{I}_D , which allows to derive the one to one binatural correspondence between morphisms of $\text{Mor}(\overline{Tr}, \overline{Nets})$ and those of $\text{Mor}(\overline{Tr}, U^t(\overline{Nets}))$, by Eq. 32 in Appendix A.3. This evidences the co-reflection of \overline{Tr} into \overline{Nets} .

Product in \overline{Tr} . An important property we expect from trellis processes concerns their factorization. Indeed, this property forms the heart of modular/distributed algorithms, that we have based on unfoldings up to now (Fabre et al. 2005). The co-reflection of the category of trellis nets into the category of nets yields directly this factorization property.

As right adjoints preserve products, one has

$$U^t(\mathcal{N}_1 \times_{\overline{N}} \mathcal{N}_2) = U^t(\mathcal{N}_1) \times_{\overline{T}} U^t(\mathcal{N}_2) \tag{11}$$

which also proves the existence of a product in \overline{Tr} for trellis nets that are time unfoldings of a net. But the unit η of the adjunction, i.e. the natural transformation $\eta : \mathbb{I}_D \rightarrow GF$ defined here by $\eta_{\mathcal{T}} : \mathcal{T} \rightarrow U_{\mathcal{T}}^t$ for every \mathcal{T} in \overline{Tr} , is obviously a natural equivalence ($\eta_{\mathcal{T}}$ is the identity). In other words, assumption (NE) of Appendix A.4 is satisfied, so the product $\times_{\overline{N}}$ reaches all elements in \overline{Tr} . We can thus define the product $\times_{\overline{T}}$ in \overline{Tr} by

$$\mathcal{T}_1 \times_{\overline{T}} \mathcal{T}_2 \cong U^t(\mathcal{T}_1) \times_{\overline{T}} U^t(\mathcal{T}_2) = U^t(\mathcal{T}_1 \times_{\overline{N}} \mathcal{T}_2) \tag{12}$$

As for $\times_{\overline{O}}$, the fact that $\times_{\overline{T}}$ is a standard product in \overline{Nets} followed by a time unfolding provides a recursive definition of the product in \overline{Tr} , based on procedure 4. Let $\mathcal{T}_1, \mathcal{T}_2$ be two TNs, with $\mathcal{T}_i = (C_i, E_i, \rightarrow_i, C_i^0, \nu_i)$, their product $\mathcal{T} = (C, E, \rightarrow, C^0, \nu) = \mathcal{T}_1 \times_{\overline{T}} \mathcal{T}_2$ and the associated morphisms $\psi_i : \mathcal{T} \rightarrow \mathcal{T}_i$ are given by:

Procedure 6

- Initialization :
 - Create $|C_1^0| + |C_2^0|$ conditions in C^0 , and define injective partial functions $\psi_i : C^0 \rightarrow C_i^0$ in such a way that they have disjoint domains.
 - Set $C = C^0, E = \emptyset, \rightarrow = \emptyset$ and $\nu = Id$.

- Recursion :
 - Let X be a co-set of C , and $I \subseteq \{1, 2\}$ a non-empty index set ;
 $\forall i \in I$, let e_i be an event of E_i such that $\psi_i(X) = \bullet e_i$.
 - If there doesn't exist an event $e \in E$ with $\bullet e = X$ and $\forall i \in I, \psi_i(e) = e_i$,
 - * create a new event $e \in E$ with $\bullet e = X$, and $\forall i \in I$ set $\psi_i(e) = e_i$,
 - * create a subset Y of $\sum_{i \in I} |e_i^*| = |X|$ new conditions in C , set $e^\bullet = Y$,
 - * extend the partial functions $\psi_i, i \in I$, to Y in order to have disjoint definition domains in Y and to satisfy $\psi_i : Y \rightarrow e_i^*$ injective,⁷
 - * $\forall c \in Y$ define $\nu(c)$ as $\psi_i^{-1} \circ \nu_i \circ \psi_i(c)$ for the unique ψ_i defined at c ,
 - * $\forall i \in I, \forall c \in Y$, if $\exists c' \in C, \psi_i(c') = \psi_i(c)$ and $H(c') = H(c)$ then merge conditions c and c' .

4 Relations between nets, trellises and unfoldings

4.1 Co-reflection of \overline{Occ} into \overline{Tr}

At this point, we have three nested categories $\overline{Occ} \subset \overline{Tr} \subset \overline{Nets}$. By restricting \overline{Nets} to \overline{Tr} in the co-reflection of \overline{Occ} into \overline{Nets} , we can derive another adjunction between \overline{Occ} and \overline{Tr} (actually another co-reflection). Specifically, take categories $C = \overline{Occ}$ and $D = \overline{Tr}$, with the inclusion functor for $F : \overline{Occ} \rightarrow \overline{Tr}$ and the unfolding functor for $G : \overline{Tr} \rightarrow \overline{Occ}$. Notice that applying \mathcal{U} to a trellis net \mathcal{T} performs an unfolding in the “conflict dimension,” since time is already unfolded (each $\mathcal{T}_{\bar{c}}$ is a partial order of nodes). We thus denote by $G = \mathcal{U}^c$ the restriction of \mathcal{U} to \overline{Tr} .

In this adjunction, the universal property of “conflict unfoldings,” corresponding to assumption (UP), yields

$$\forall \mathcal{T} \in \overline{Tr}, \forall \mathcal{O} \in \overline{Occ}, \forall \phi : \mathcal{O} \rightarrow \mathcal{T}, \exists! \psi : \mathcal{O} \rightarrow \mathcal{U}_{\mathcal{T}}^c, \phi = f_{\mathcal{T}}^c \circ \psi \tag{13}$$

where $f_{\mathcal{T}}^c : \mathcal{U}_{\mathcal{T}}^c \rightarrow \mathcal{T}$ is the folding of $\mathcal{U}^c(\mathcal{T})$ into \mathcal{T} . Let $\mathcal{T}_1, \mathcal{T}_2$ be two TNs, the limit preservation theorem on right adjoints gives :

$$\mathcal{U}^c(\mathcal{T}_1 \times_{\overline{\mathcal{T}}} \mathcal{T}_2) = \mathcal{U}^c(\mathcal{T}_1) \times_{\overline{\mathcal{O}}} \mathcal{U}^c(\mathcal{T}_2) \tag{14}$$

And finally, using the fact that the unit of the adjunction, $\eta_{\mathcal{O}} : \mathcal{O} \rightarrow \mathcal{U}^c(\mathcal{O})$, defines a natural equivalence in \overline{Occ} (assumption (NE)), we can actually use this relation to (re)define the product $\times_{\overline{\mathcal{O}}}$ by

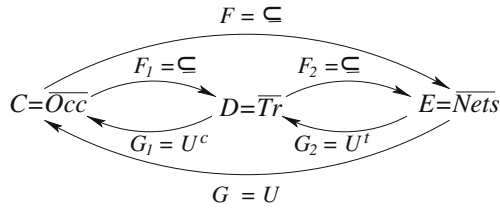
$$\mathcal{O}_1 \times_{\overline{\mathcal{O}}} \mathcal{O}_2 \cong \mathcal{U}^c(\mathcal{O}_1) \times_{\overline{\mathcal{O}}} \mathcal{U}^c(\mathcal{O}_2) = \mathcal{U}^c(\mathcal{O}_1 \times_{\overline{\mathcal{T}}} \mathcal{O}_2) \tag{15}$$

4.2 Composition of adjunctions

Gathering results obtained so far, we have three adjunctions relating categories $C = \overline{Occ}, D = \overline{Tr}$ and $E = \overline{Nets}$, as displayed by Fig. 11. It is a well known fact that adjunctions can be composed (Mac Lane 1971, chap. IV-8, thm 1), so $F_2 \circ F_1 :$

⁷on its domain of definition

Fig. 11 Adjunctions relating categories \overline{Occ} , \overline{Tr} and \overline{Nets}



$\overline{Occ} \rightarrow \overline{Nets}$ and $G_1 \circ G_2 : \overline{Nets} \rightarrow \overline{Occ}$ defines an adjunction. Since $F_2 \circ F_1 = F$ is the inclusion functor, we thus have that $G = G_1 \circ G_2$, up to a natural equivalence.⁸ This translates into

$$\forall \mathcal{N} \in \overline{Nets}, \quad \mathcal{U}(\mathcal{N}) \cong \mathcal{U}^c \circ \mathcal{U}^t(\mathcal{N}) \tag{16}$$

and naturally the corresponding foldings can be composed: $f_{\mathcal{N}} = f_{\mathcal{N}}^t \circ f_{\mathcal{U}_{\mathcal{N}}}^c$, up to the isomorphism in Eq. 16.

Notice that Eq. 16 expresses that the time-unfolding $\mathcal{U}^t(\mathcal{N})$ of a net can be recovered by “refolding” conflicts on the full unfolding $\mathcal{U}_{\mathcal{N}}$. Specifically, $f_{\mathcal{U}_{\mathcal{N}}}^c : \mathcal{U}(\mathcal{N}) \rightarrow \mathcal{U}^t(\mathcal{N})$ merges conditions with the same height and representing the same place of \mathcal{N} , then merges (or removes) redundant events representing the same transition connected to a given co-set. This is illustrated in Fig. 12 that compares the unfolding and the trellis of a net.

In terms of product preservation, the composition of adjoints yields, for any pair $\mathcal{N}_1, \mathcal{N}_2$ in \overline{Nets}

$$\mathcal{U}(\mathcal{N}_1 \times_{\overline{N}} \mathcal{N}_2) = \mathcal{U}(\mathcal{N}_1) \times_{\overline{O}} \mathcal{U}(\mathcal{N}_2) \tag{17}$$

$$\cong \mathcal{U}^c \circ \mathcal{U}^t(\mathcal{N}_1 \times_{\overline{N}} \mathcal{N}_2) \tag{18}$$

$$= \mathcal{U}^c(\mathcal{U}^t(\mathcal{N}_1) \times_{\overline{T}} \mathcal{U}^t(\mathcal{N}_2)) \tag{19}$$

$$= \mathcal{U}^c(\mathcal{U}^t(\mathcal{N}_1)) \times_{\overline{O}} \mathcal{U}^c(\mathcal{U}^t(\mathcal{N}_2)) \tag{20}$$

5 Application to labeled nets

As mentioned in Section 2.1, the standard product of nets is generally not used in its basic form, but is rather constrained by a synchronization algebra. The latter specifies where synchronizations must take place, and what transitions can be considered as “private” to a component. We use this formalism to illustrate an important property of trellis processes.

A labeled multi-clock net (LMCN) $\mathcal{N} = (P, T, \rightarrow, P^0, \mu, \lambda, \Lambda)$ is a (multi-clock) net extended with a labeling function λ on T , taking values in the alphabet Λ . A morphism $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ of LMCN satisfies both the requirements of labeled net morphisms, and the component preservation property (5). This means in particular that the components preserved by ϕ exactly match transitions of \mathcal{N}_1 carrying a label

⁸The adjoint of a functor is unique up to a natural equivalence, see (Mac Lane 1971), chap. IV-1, cor. 1.

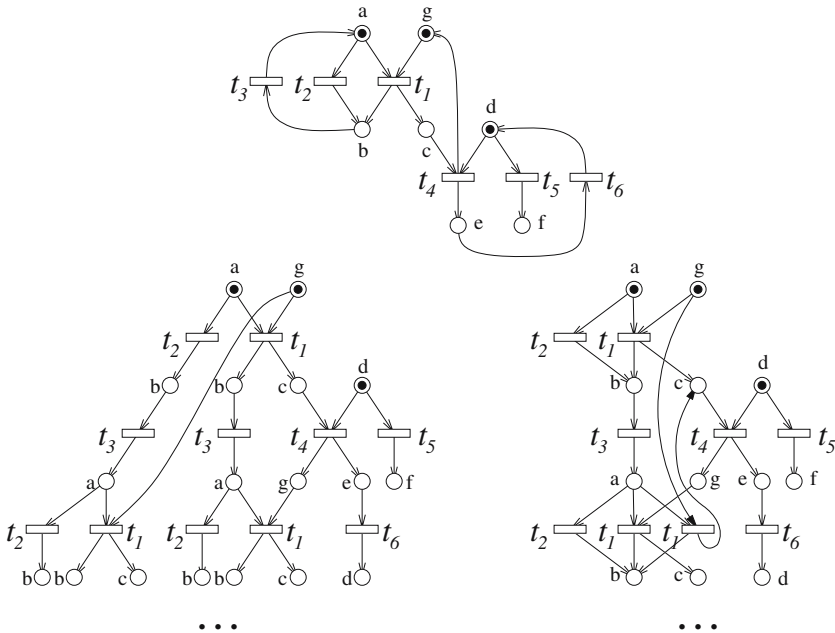


Fig. 12 A net \mathcal{N} (top), with three sequential components, defined by $\{a, b\}$, $\{g, c\}$ and $\{d, e, f\}$. Its unfolding $\mathcal{U}_{\mathcal{N}}$ (bottom left), and its trellis $\mathcal{U}'_{\mathcal{N}}$ (bottom right). For clarity, conditions/events are labeled by the place/transition they represent in \mathcal{N} , instead of having distinct names

of Λ_2 . Labeled nets, trellis nets and occurrence nets form the nested categories $\lambda\overline{Nets} \supset \lambda\overline{Tr} \supset \lambda\overline{Occ}$.

With the synchronization algebra defined in Section 2, the product $\mathcal{N}_1 \times_{\lambda\overline{N}} \mathcal{N}_2$ of two labeled nets $\mathcal{N}_i = (P_i, T_i, \rightarrow_i, P_i^0, \mu_i, \lambda_i, \Lambda_i)$ is obtained by first computing the ordinary product in \overline{Nets} , and then preserving only transition pairs (t_1, t_2) for which $\lambda_1(t_1) = \lambda_2(t_2)$, as well local transitions (t_1, \star) (resp. (\star, t_2)) for which $\lambda_1(t_1) \in \Lambda_1 \setminus \Lambda_2$ (resp. $\lambda_2(t_2) \in \Lambda_2 \setminus \Lambda_1$). In the same way, the product $\times_{\lambda\overline{Tr}}$ (resp. $\times_{\lambda\overline{Occ}}$) of labeled trellis nets (resp. occurrence nets) can be obtained by 1/ taking the standard product of non-labeled trellis nets (resp. occurrence nets), and 2/ removing transitions not matching the rules of the synchronization algebra.

It is also straightforward to check that the three categories above are related exactly as \overline{Nets} , \overline{Tr} and \overline{Occ} . So the products $\times_{\lambda\overline{Tr}}$ and $\times_{\lambda\overline{Occ}}$ can be derived from $\times_{\lambda\overline{N}}$ by Eq. 4 and Eq. 12. This yields recursive definitions for them, detailed in Appendix C.

5.1 Factorization in elementary trellises

With this simple synchronization algebra, every multi-clock net $\mathcal{N} = (P, T, \rightarrow, P^0, \mu)$ can be viewed as a synchronous product of sequential machines (Fig. 13). Add to \mathcal{N} the label set $\Lambda = T$ and take the identity as labeling function λ . Then consider each restriction $\mathcal{N}_{|_{\bar{p}}}$ for $p \in P^0$, with $T_{|_{\bar{p}}} = \{t \in T : \bullet t \cap \bar{p} \neq \emptyset\}$ as transition set, and $\Lambda = T_{|_{\bar{p}}}$ as label set. By definition of multi-clock nets, $\mathcal{N}_{|_{\bar{p}}}$ is a sequential machine,

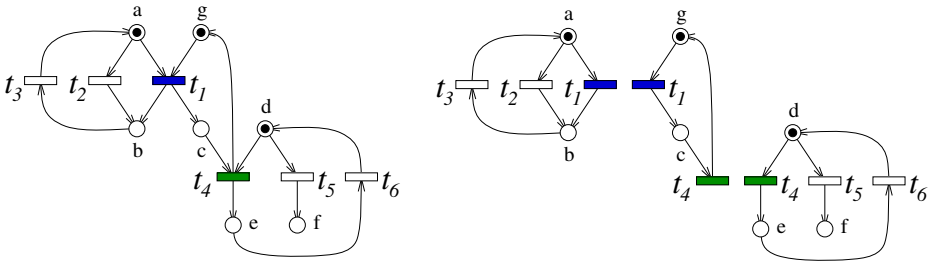


Fig. 13 A multi-clock net \mathcal{N} (left) and its decomposition as a product of labeled automata $\mathcal{N}_{|a}, \mathcal{N}_{|g}, \mathcal{N}_{|d}$ (right). The labels are transition names

and clearly $\mathcal{N} = \times_{\lambda \bar{N}, p \in P^0} \mathcal{N}_{|p}$. This decomposition justifies *a posteriori* the name “multi-clock net”: each automaton $\mathcal{N}_{|p}$ has a natural notion of time, that we use to compute the height function.

Applying Eq. 11 to such a decomposition reveals that the time-unfolding of a net \mathcal{N} is the product (in $\lambda \overline{Tr}$) of the time unfoldings of its components $\mathcal{N}_{|p}$. And the latter are nothing more than ordinary automata trellises, as they are usually understood by several communities (Fig. 14). This nice property adds substance to the claim that trellis nets are the correct generalization to concurrent systems of the ordinary notion of trellis.

5.2 Other properties of trellises

We take advantage of the previous discussion to illustrate other properties of trellises. By construction, the trellis of a net remains bounded on conditions: the number of conditions having the same height is bounded by a constant. However, this does not hold on events, as one can see in Fig. 13: the number of events labeled t_1 in $\mathcal{U}'_{\mathcal{N}}$ increases with height. This phenomenon is in favor of factorized forms of

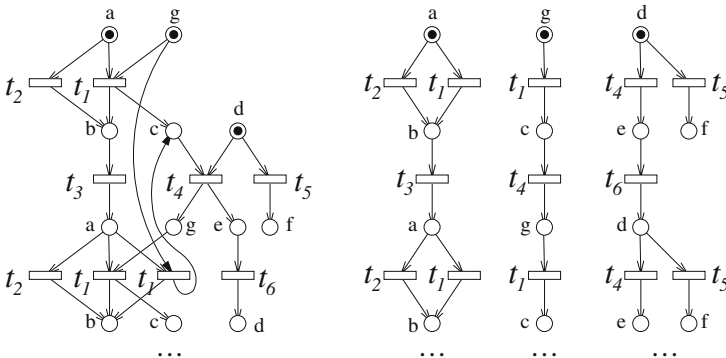


Fig. 14 Left: Trellis of the net \mathcal{N} depicted in Fig. 13. Right: Trellises of its components $\mathcal{N}_{|a}, \mathcal{N}_{|g}, \mathcal{N}_{|d}$. The trellis on the LHS is the product of the three other trellises, in the sense of $\lambda \overline{Tr}$: $\mathcal{U}'_{\mathcal{N}} = \mathcal{U}'_{\mathcal{N}_{|a}} \times_{\lambda \overline{Tr}} \mathcal{U}'_{\mathcal{N}_{|g}} \times_{\lambda \overline{Tr}} \mathcal{U}'_{\mathcal{N}_{|d}}$

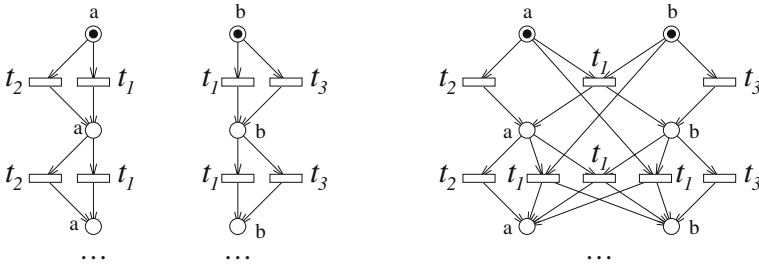


Fig. 15 Two labeled TPs (*left*), sharing label t_1 , and their product (*right*). This example shows that the factorization of TP has little impact on the number of conditions, but significantly reduces the number of events, and the complexity of the TP. The factors on the *left* have constant width at a given height, but the product explodes

trellises: although it keeps the number of conditions under control, the product of trellises augments the number of events, as in the case of unfoldings (see for example Fig. 15).

We have mentioned in Section 2.4 that the restriction to multi-clock nets was harmless: one can always add complementary places to a safe net and make it multi-clock, without changing its behavior. This operation has little impact on the construction of unfoldings: let \mathcal{N} be a safe net and $\tilde{\mathcal{N}}$ its complemented version, then $\mathcal{U}_{\mathcal{N}}$ can be recovered by erasing conditions pointing to complementary places in $\mathcal{U}_{\tilde{\mathcal{N}}}$. Things are different with trellises, as shown by the example in Fig. 16. This net has a single sequential component. By adding complementary places, one artificially creates three sequential components, and thus three clocks instead of one. Although configurations of \mathcal{N} and $\tilde{\mathcal{N}}$ are in a simple one to one correspondence, trellis shapes are strongly different, due to the different ways of computing heights.

6 Variations around the height function

Referring to Eq. 16, trellis processes are obtained as a maximal (conflict) folding of a branching process, guided by a height function H . To ensure all properties of the previous sections, the latter must essentially satisfy two conditions:

1. H must be a causal function, i.e. must depend only on the past of a condition in a given string, in order to allow recursive constructions of trellis processes;
2. and H must be a monotonic function, in order to prevent the creation of circuits in each $\mathcal{T}_{\tilde{c}}$.

This leaves a fair amount of flexibility that we explore now.

6.1 Height measure by a causal monotonic function

Let $(\mathcal{E}, <)$ be a partially ordered set. A *height function* H taking values in \mathcal{E} is a collection of functions H_σ , for $\sigma = (C_\sigma, E_\sigma, \rightarrow, C_\sigma^0, \nu)$ a string, with $H_\sigma : C_\sigma \rightarrow \mathcal{E}$, and such that H is invariant by string isomorphism. H is said to be *causal* iff it satisfies

$$\forall \sigma, \forall c \in C_\sigma, \quad H_\sigma(c) = H_{[c]}(c) \tag{21}$$

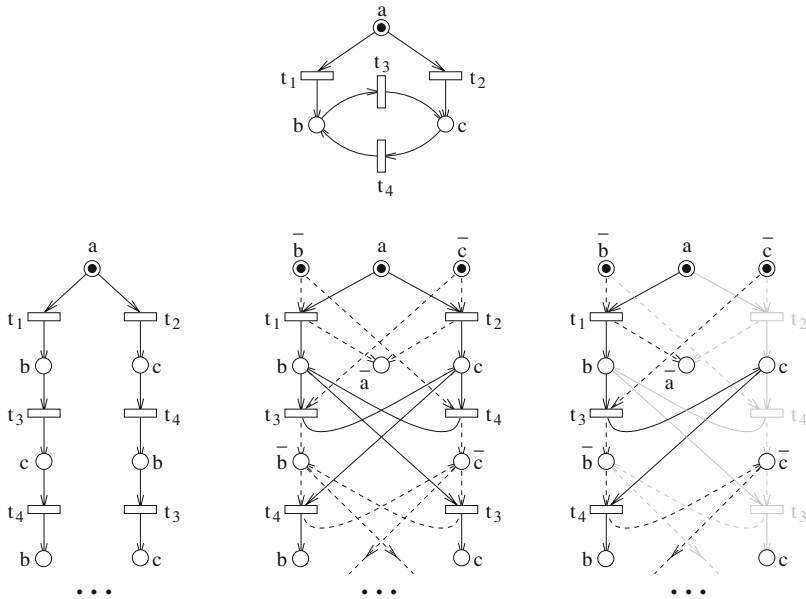


Fig. 16 A net \mathcal{N} (top), its trellis (bottom left), and the trellis of its complemented version $\tilde{\mathcal{N}}$ (center). Dashed lines underline the effect of complementary places. One of the two infinite configurations of $\mathcal{U}_{\tilde{\mathcal{N}}}^t$ is enlightened in the rightmost net

where $[c]$ denotes the minimal sub-string of σ containing c . H is monotonic iff

$$\forall \sigma, \forall c, c' \in C_\sigma, \quad c \rightarrow^* c' \Rightarrow H_\sigma(c) < H_\sigma(c') \tag{22}$$

The definition of a causal monotonic H given above introduces little flexibility since two strings are isomorphic as soon as they have the same length. So we basically rephrased Eq. 6... Things change when one considers richer structures. For instance, let us define *weighted nets* (associated to weight-preserving morphisms) as $\mathcal{N} = (P, T, \rightarrow, P^0, v, w)$, with $w : P \cup T \rightarrow \mathbb{R}^+$. On a weighted string σ , one can define

$$H_\sigma(c) = \sum_{e \in E_\sigma, e \rightarrow^* c} w(e) \quad \text{or} \tag{23}$$

$$H_\sigma(c) = \sum_{x \in C_\sigma \cup E_\sigma, x \rightarrow^* c} w(x) \tag{24}$$

As another example, let us consider labeled nets $\mathcal{N} = (P, T, \rightarrow, P^0, v, \lambda, \Lambda)$, where all label sets Λ are included in \mathcal{L} . Take $\mathcal{E} = \mathcal{L}^*$, and let $<$ be the prefix relation in \mathcal{L}^* . In a labeled string $\sigma = (C_\sigma, E_\sigma, \rightarrow, C_\sigma^0, v, \lambda, \Lambda)$ corresponding to the sequence $c_0 \rightarrow e_1 \rightarrow c_1 \rightarrow e_2 \rightarrow \dots \rightarrow e_n \rightarrow c_n (\rightarrow \dots)$ one can define

$$H_\sigma(c_n) = \lambda(e_1) | \lambda(e_2) | \dots | \lambda(e_n) \tag{25}$$

where $|$ denotes the concatenation operator.

Lemma 4 *Given the height function H defined by Eq. 25, consider the labeled net $\mathcal{N} = (P, T, \rightarrow, P^0, \nu, \lambda, \Lambda)$. If λ is injective, the trellis $\mathcal{U}_{\mathcal{N}}^t$ is isomorphic to the unfolding $\mathcal{U}_{\mathcal{N}}$.*

Proof Let (\mathcal{O}, ϕ) be a branching process of \mathcal{N} , and consider its restriction $\mathcal{O}_{|c}$ for $c \in C^0$. If λ is injective in \mathcal{N} , it is also injective in any of its sequential components, and in particular in $\mathcal{N}_{|p}$ for $p = \phi(c) \in P^0$. With this remark, two conditions of $\mathcal{O}_{|c}$ have the same height iff they have been generated by the same sequence of transitions of \mathcal{N} , and thus are identical. Therefore \mathcal{O} is correctly and maximally folded for H , which defines a trellis process of \mathcal{N} . □

It is straightforward to check that, given a causal monotonic height function, correctly folded trellis nets enjoy the same properties as before. Time unfoldings can still be defined and satisfy theorem 2.

6.2 Different merge rules in the sequential components

A height function operates on runs of a sequential component, but with the definitions above, H is the same for all sequential components of all nets. We describe here a mechanism that allows a fine tuning of the height function, according to the component to which it applies. We start by attaching a “type” to each sequential component of a MCN. Let \mathcal{A} be a set of possible types, a *typed net* $\mathcal{N} = (P, T, \rightarrow, P^0, \nu, \tau)$ is a MCN enriched with a function $\tau : P^0 \rightarrow \mathcal{A}$. The type function τ extends to all places $p \in P$ by $\tau(p) \triangleq \tau(\nu(p))$. We naturally limit ourselves to type preserving morphisms.

It is now possible to define different height functions, according to the component type: a *typed height function* is a collection of height functions $H_{\cdot, \alpha}$ operating on strings of type $\alpha \in \mathcal{A}$. The height of condition c in a typed configuration κ is naturally given by $H_{\cdot, \tau(c)}$. With the extra requirement that each $H_{\cdot, \alpha}$ is causal and monotonic, one recovers all properties of trellis nets.

Combining this mechanism with the ideas of Section 6.1 allows a fine tuning of the “refolding degree” performed by the trellis of a net. For example, it is possible to decide that in $\mathcal{U}_{\mathcal{N}}$, runs of a given sequential component will remain completely unfolded, whereas for another component, the usual trellis structure will be chosen. Consider the running example of Fig. 12, where \mathcal{N} has three sequential components $\mathcal{N}_{|a}$, $\mathcal{N}_{|g}$ and $\mathcal{N}_{|d}$. Let H be chosen as in Eq. 6 for all components excepted $\mathcal{N}_{|g}$, for which H imposes no merge at all. One gets the LHS trellis in Fig. 17. Conversely, if runs of $\mathcal{N}_{|a}$ aren’t merged, one gets the RHS trellis. Both satisfy the universal property (for the corresponding choice of height function), and thus enjoy the factorization property (Eq. 11).

To summarize things schematically, by playing with the height function one can place the central category $D = \overline{Tr}$ in Fig. 11 at different positions between $C = \overline{Occ}$ and $E = \overline{Nets}$. \overline{Tr} can be very “close” to \overline{Occ} , or even coincide with it.

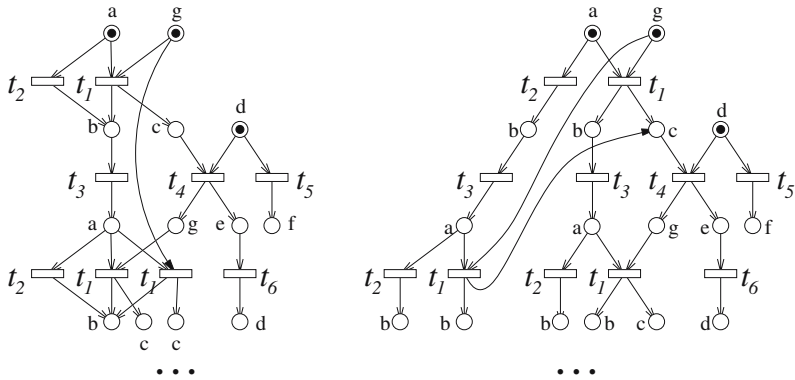


Fig. 17 Two trellises of the net \mathcal{N} of Fig. 13. *On the left*, the height function doesn't refold runs of the sequential component $\mathcal{N}_{|g}$. Observe the two occurrences of c on the *bottom line*. Conversely, runs of $\mathcal{N}_{|a}$ are not refolded on the *RHS trellis*

7 Trellis based diagnosis

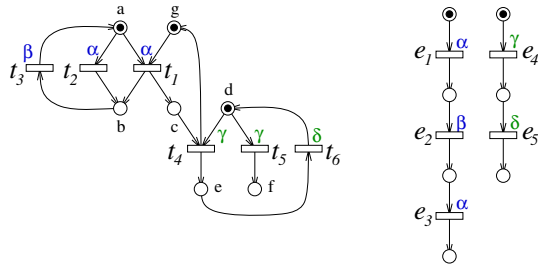
Up to here we have intensively studied trellis nets and their factorization properties. We consider now the application of trellis nets to the diagnosis of concurrent systems. In the case of an automaton, the diagnosis problem consists in recovering all possible runs of the system that explain a given sequence of observations, from which one can easily infer whether an undesirable event certainly occurred, may have occurred or didn't occur for sure in the actual run that produced the observations.

Here, we complexify the setting on two points.

First, we consider a concurrent system, under the form of a labeled MCN \mathcal{N} , and assume a true concurrency semantics on its runs. So runs of \mathcal{N} are actually configurations κ in $\mathcal{U}_{\mathcal{N}}$ or in $\mathcal{U}'_{\mathcal{N}}$. The labels carried by transitions of \mathcal{N} correspond to “visible events:” when t fires, the label $\lambda(t)$ is observed. Notice that transitions may be “silent,” i.e. may not produce any visible event, which we denote by $\lambda(t) = \epsilon$. Ambiguities go further: several transitions may have the same label. Systems where transitions could produce several possible signatures can also be modeled, by duplicating these transitions and associating one of the possible labels to each copy.

The second difference concerns the observation process. For an automaton, labels are produced in sequence, and collected as such. With a concurrent system, and no notion of global time, the picture is more complex. As a first model, we can assume labels are collected by a sensor into an observation sequence, satisfying the *causal observation assumption (COA)*: if two events e_1 and e_2 are causally related in the hidden run κ_{true} , the corresponding labels appear in this order in the observed sequence. Otherwise they may appear in any order. In other words, one observes the labels of a linear extension of κ_{true} . In large concurrent systems however, another situation generally prevails: there generally exist several sensors, collecting observations produced by different parts of the system. For example in the case of a telecommunication network, several operators are in charge of monitoring different parts of the network, and thus only see part of the “alarms” produced by the net. In this situation, observations are rather a tuple of sequences, and the ordering relations between the different sequences are lost. This can be captured by the following

Fig. 18 A labeled net \mathcal{N} (left), and an observed partial order of labels produced by a hidden run of \mathcal{N} , represented as a labeled configuration \mathcal{A} (right)



general setting : observations are a partial order \mathcal{A} of events, with a labeling function $\lambda_{\mathcal{A}}$. \mathcal{A} is obtained by selecting visible events in the partial order defined by κ_{true} , adding some extra causality links, and then removing causality relations in the result. This construction clearly preserves the COA (see Fig. 18).

With the above assumptions, a run κ of \mathcal{N} is a possible candidate for κ_{true} iff it satisfies

$$\psi_A(\kappa \times_{\lambda \overline{O}} \mathcal{A}) = \mathcal{A} \tag{26}$$

$$\psi_N(\kappa \times_{\lambda \overline{O}} \mathcal{A}) = \kappa \tag{27}$$

where ψ_N, ψ_A are the morphisms relating the product $\kappa \times_{\lambda \overline{O}} \mathcal{A}$ to factors κ and \mathcal{A} respectively. In effect, the product synchronizes events of κ with observations, through the labels, and at the same time guarantees that causality relations on both sides are satisfied. Silent events in κ are naturally regarded as private : they don't have to synchronize with an observed event. Equation 26 ensures that all observations are explained, and Eq. 27 ensures that the minimal explanation is selected : recall that in general $\psi_N(\kappa \times_{\lambda \overline{O}} \mathcal{A}) \sqsubseteq \kappa$.

It is possible to compute all such κ 's at the same time, either by considering $\mathcal{U}_{\mathcal{N}} \times_{\lambda \overline{O}} \mathcal{A}$, as it was done in (Benveniste et al. 2003), or by considering $\mathcal{U}'_{\mathcal{N}} \times_{\lambda \overline{T}} \mathcal{A}$, solution that we illustrate now (Fig. 19). The runs of \mathcal{N} satisfying Eq. 27 are the configurations of $\mathcal{T} = \psi_N(\mathcal{U}'_{\mathcal{N}} \times_{\lambda \overline{T}} \mathcal{A})$. \mathcal{T} can be very easily computed with the recursive procedure 7 given in Appendix C, and it obviously satisfies $\mathcal{T} = \psi_N(\mathcal{T} \times_{\lambda \overline{T}} \mathcal{A})$. The solutions to the diagnosis problem are then the configurations of $\mathcal{T} \times_{\lambda \overline{T}} \mathcal{A}$ satisfying Eq. 26, which are easy to extract (they are characterized by the fact that they contain the maximal events of \mathcal{A}). They are also maximal configurations in \mathcal{T} (up to silent events). On the example of Fig. 19, there exist three such solutions. All of them contain the events pointing to t_4 and t_6 of \mathcal{N} , and none of them contains the event pointing to t_5 (since label δ wouldn't be explained).

The diagnosis procedure described above only relies on the product of trellis nets. But it is possible to make a deeper use of the previous sections : the factorization properties of trellises allow to solve the diagnosis problem by parts, with a distributed algorithm, as it was done in (Fabre et al. 2005) with prime event structures. This approach requires the introduction of extra material, in particular a notion of projection on trellis processes, and the satisfaction of joint properties by the product and projection operators. A simple version of modular computations based on trellis processes is presented in (Fabre 2005); the general treatment will be detailed in a forthcoming paper.

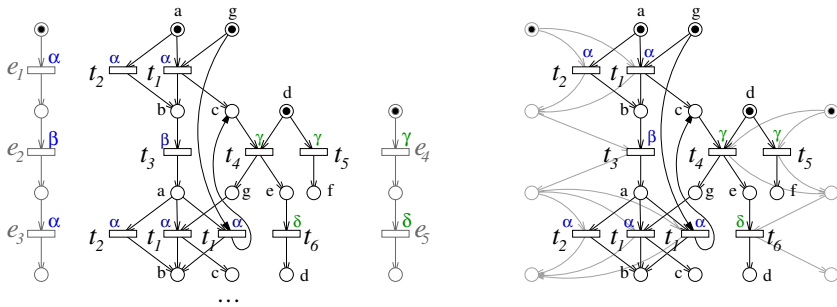


Fig. 19 Product of U_N^t with observations \mathcal{A} : before (left) and after (right)

8 Conclusion

Unfoldings were introduced as an efficient way of representing all possible runs of a concurrent system, in the so-called true concurrency semantics. We have proposed in this paper an alternate and more compact data structure, the trellis, to represent the same set of runs. The trellis of a net can be obtained either directly, or by a partial refolding of the unfolding of that net, guided by a simple height function. The price to pay for having a more compact structure is a more complex procedure to extract configurations, i.e. runs of the system. However, by an appropriate choice of the height function, one has at least a means to adjust this classical trade-off between memory and computation complexity. The height function is indeed the element that pilots the refolding degree of conflicts. Beyond the compactness, trellis processes appear to be the right generalization to concurrent systems of the ordinary notion of trellis of an automaton, which forms the core of on-line algorithms in many applications.

Like the unfolding, the trellis of a concurrent system enjoys factorization properties: When a system can be expressed as a product of components, its trellis itself is the product (in an appropriate sense) of the trellises of these components. The factorized form of a trellis is of course even more compact, and the factors are much simpler than their product. This suggests a first strategy to simplify the extraction of configurations, by operating “by parts.” More generally, factorized forms of unfoldings allow distributed/modular processings on complex systems (see Fabre et al. (2005) for an application in distributed diagnosis). To work with factorized forms of an unfolding, one needs a product but also a projection operator, that both satisfy a small set of axioms (Fabre 2003b). The derivation of this framework for trellis nets will be detailed in a forthcoming paper.

A standard problem about unfoldings is the derivation of a finite complete prefix (FCP), which was the key to make unfolding techniques a practical tool. What does this notion become with trellises? As a first easy answer, one can always claim that a FCP of the unfolding yields a FCP of the trellis, by simply merging conditions with same height and same image place in the original net. Conversely, a FCP of the trellis yields a FCP of the unfolding, by the reverse unfolding operation. Indeed, like branching processes, trellis processes are only a compact way of representing a given set of configurations. So as far as these configurations are selected regardless of the data structure that supports them, one gets the above correspondence. Is there

an efficient way, however, to compute directly a finite complete prefix of the trellis? Can we define a minimal FCP in a trellis, or a canonical one? We leave these open questions to a future work.

Acknowledgement The author is grateful to Philippe Darondeau and Glynn Winskel for fruitful and patient exchanges in the elaboration of early versions of this work.

Appendix A: Some keypoints about adjunctions

This section briefly recalls some elements of category theory. Its objective is to show how a few key results trigger standard constructions, from which we derive most properties mentioned without proof in the paper. The paper will thus concentrate on these key results. We assume the reader is familiar with the basic notions of category, functor and natural transformation of a functor into another (Mac Lane 1971).

A.1 Product

Let o_1 and o_2 be two objects in a category C , and consider triples (o, f_1, f_2) where o is an object in C , and the $f_i : o \rightarrow o_i$ are morphisms, $i = 1, 2$. The *product* of o_1 and o_2 in C , if it exists, is defined as such a triple $o_1 \times_C o_2 \triangleq (p, \psi_1, \psi_2)$ which is also required to be extremal. Specifically,

$$\forall (o, f_1, f_2), \exists ! \phi : o \rightarrow p, [f_1 = \psi_1 \circ \phi \text{ and } f_2 = \psi_2 \circ \phi] \tag{28}$$

This condition is referred to as the universal property of the product (Fig. 20). The product, when it exists, is unique up to isomorphism.

A.2 Adjunction

This is probably the main tool in this paper: it forms the basis of most results we state. An *adjunction* between two categories C and D is defined as a triple (F, G, ϕ) . $F : C \rightarrow D$ and $G : D \rightarrow C$ are two functors relating C and D , and working in opposite directions. F can be understood as an embedding of C into D (it reshapes objects), and G as a projection (or abstraction) of objects of D onto C . F and G are respectively called the left and right adjoints. ϕ is a bijective correspondence between morphisms of the two categories. It is used to express how object relations and constructions in one category are mapped to the other one. Specifically, for any two

Fig. 20 Universal property of the product, expressed as a commutative diagram

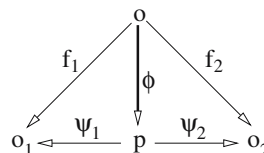
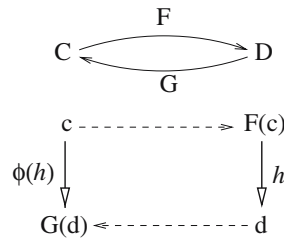


Fig. 21 An adjunction between two categories



objects $c \in C$ and $d \in D$, $\phi_{c,d} : \text{Mor}_D(F(c), d) \rightarrow \text{Mor}_C(c, G(d))$ is bijective,⁹ where $\text{Mor}_X(u, v)$ represents the set of morphisms from u to v in category X (see Fig. 21).

The mapping ϕ is also required to be “natural in c and in d ,” which means the following. Let $f : c' \rightarrow c$ and $g : d \rightarrow d'$ be two morphisms in C and D respectively, and $h \in \text{Mor}_D(F(c), d)$, then (see Fig. 22)

$$\phi(g \circ h \circ F(f)) = G(g) \circ \phi(h) \circ f \tag{29}$$

This property can obviously be checked separately in f and in g .

A.3 Construction

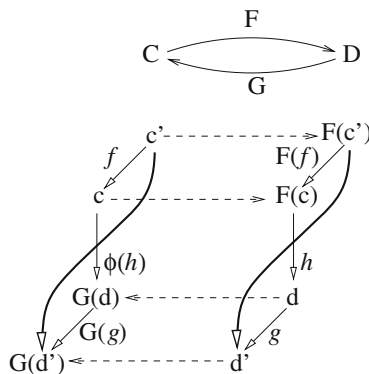
In this paper, functor F is generally given, and G is defined on objects of C . So the construction of an adjunction amounts to proving that G is indeed a functor (i.e. is also defined on morphisms), and to obtaining the correspondence ϕ . The keystone of the construction is the following assumption :

(UP) for every object $d \in D$, there exists a morphism $\epsilon_d : FG(d) \rightarrow d$ such that the pair $(G(d), \epsilon_d)$ forms a universal morphism from (images of) functor F to d :

$$\forall c \in C, \forall g : F(c) \rightarrow d, \exists ! f : c \rightarrow G(d) \quad g = \epsilon_d \circ F(f) \tag{30}$$

In words, all morphisms from $F(c)$ to d factorize through $FG(d)$.

Fig. 22 Naturality of the mapping ϕ in an adjunction



⁹Indexes c, d are often omitted in $\phi_{c,d}$.

From this assumption, the adjunction is derived in the following way (Mac Lane 1971, chap. IV-1, thm 2-iv).

One first proves that G can be extended to morphisms of D , and form a functor $G : D \rightarrow C$. Let $g : d \rightarrow d'$ be an arrow of D , and consider morphism $g \circ \epsilon_d : FG(d) \rightarrow d'$. By (UP), there exists a unique $f : G(d) \rightarrow G(d')$ in C such that $\epsilon_{d'} \circ F(f) = g \circ \epsilon_d$. We define $G(g) \triangleq f$, so

$$g \circ \epsilon_d = \epsilon_{d'} \circ FG(g) \tag{31}$$

It is then easy to show that G is a functor: $G(1) = 1$ and $G(g_2 \circ g_1) = G(g_2) \circ G(g_1)$.

Property (19) expresses that $\epsilon : FG \rightarrow \mathbb{I}_D$ is a natural transformation of functor FG to the identity functor \mathbb{I}_D in D . The correspondence ϕ is then derived in the following way :

$$\forall h : c \rightarrow G(d), \quad \phi^{-1}(h) \triangleq \epsilon_d \circ F(h) \tag{32}$$

which implies in particular $\epsilon_d = \phi^{-1}(1_{G(d)})$, see Fig. 23. By (UP), one checks that $(\phi^{-1})^{-1}$ is well defined, i.e. that ϕ is bijective. By definition of ϕ^{-1} , one has $\phi^{-1}(h \circ f) = \phi^{-1}(h) \circ F(f)$ which proves the naturality of ϕ in c . The naturality of ϕ in d means $\phi^{-1}(G(g) \circ h) = g \circ \phi^{-1}(h)$, which is a direct consequence of Eq. 31, i.e. the commutative square on the right-hand side of Fig. 23.

A.4 Properties

Preservation of limits. This is the main result we use on adjunctions : a functor which is a right adjoint preserves limits, and in particular products, which are a special kind of limits (Mac Lane 1971, chap. V-5, thm 1). Specifically, let d_1, d_2 be objects in D and let $d_1 \times_D d_2 = (d, \psi_1, \psi_2)$, then $(G(d), G(\psi_1), G(\psi_2))$ satisfy the universal property of the product for $G(d_1)$ and $G(d_2)$ in C , which allows to define \times_C on the objects of C lying in the image of G .

Fig. 23 Derivation of an adjunction, by extending G into a functor, and deriving G from the co-unit ϵ

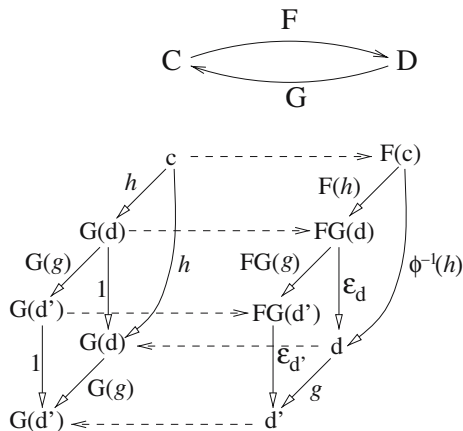
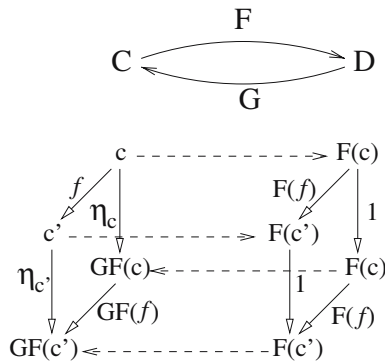


Fig. 24 Unit of an adjunction, from which ϕ can also be reconstructed



Unit. The natural transformation $\epsilon : FG \rightarrow \mathbb{I}_D$ is called the *co-unit* of the adjunction. By symmetry, the *unit* η can be defined as the natural transformation $\eta : \mathbb{I}_D \rightarrow GF$ in category C (Fig. 24), and ϕ can be recovered from η if the counterpart of (UP) is satisfied.

A special case deserves some interest. Assume

(NE) The unit η is a natural equivalence,¹⁰ i.e. has an inverse η^{-1} (which is then also a natural transformation).

Then every object $c \in C$ is isomorphic to $GF(c)$. So if a product \times_D exists in D , this product can be mapped by G into a product on objects of $G(D)$ in C , and finally into a product for all pairs of objects in C , since the product is defined up to isomorphism. One thus has

$$\forall c_1, c_2 \in C, \quad c_1 \times_C c_2 \cong GF(c_1) \times_C GF(c_2) = G(F(c_1) \times_D F(c_2)) \quad (33)$$

Composition of adjunctions. When (F, G, ϕ) is an adjunction between C and D , and (F', G', ϕ') an adjunction between D and E , one easily checks that $(\bar{F}, \bar{G}, \bar{\phi}) \triangleq (F' \circ F, G \circ G', \phi \circ \phi')$ defines an adjunction between C and E (Mac Lane 1971, chap. IV-8, thm 1). Its co-unit $\bar{\epsilon}$ and its unit $\bar{\eta}$ are given by

$$\forall c \in C, \quad \bar{\epsilon}_c = G(\epsilon'_{F(c)}) \circ \epsilon_c \quad (34)$$

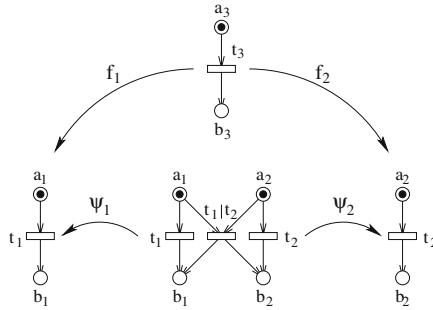
$$\forall e \in E, \quad \bar{\eta}_e = \eta'_e \circ F'(\eta_{G'(e)}) \quad (35)$$

Appendix B: On the choice of net morphisms

This section illustrates the fact that, if one does not allow morphisms that duplicate places of a net, the resulting category *Nets* doesn't have a categorical product, and

¹⁰The synonym "natural isomorphism" is also frequent.

Fig. 25 The universal property of the product requires morphisms with the ability to duplicate places



so is necessarily incomplete.¹¹ We provide a simple counter-example : two nets for which the standard construction of product violates the universal property.

Consider the three isomorphic nets \mathcal{N}_i , $i \in \{1, 2, 3\}$, each composed of a single transition t_i , with one input place a_i (initially marked) and one output place b_i (empty). By construction, there exist unique morphisms ψ_i from (what should be) the product $\mathcal{N}_1 \times_{\mathcal{N}} \mathcal{N}_2$ to factors \mathcal{N}_i , $i \in \{1, 2\}$ (see Fig. 25). In the same way, there exist isomorphisms $f_i : \mathcal{N}_3 \rightarrow \mathcal{N}_i$. Finally, the unique morphism $\phi : \mathcal{N}_3 \rightarrow \mathcal{N}_1 \times_{\mathcal{N}} \mathcal{N}_2$ that makes the diagram commutative, i.e. that satisfies $f_i = \psi_i \circ \phi$, is such that it duplicates places of \mathcal{N}_3 : a_3 is mapped to a_1 and a_2 in the product, b_3 to b_1 and b_2 , and t_3 is mapped to the synchronized transition (t_1, t_2) . So if morphisms duplicating places are not allowed, the commutative diagram cannot be constructed, and the universal property of the candidate product $\mathcal{N}_1 \times_{\mathcal{N}} \mathcal{N}_2$ is lost.

Appendix C: Product of labeled trellis nets

Let $\mathcal{N}_1, \mathcal{N}_2$ be two labeled nets, $\mathcal{N}_i = (P_i, T_i, \rightarrow_i, P_i^0, \mu_i, \lambda_i, \Lambda_i)$, their product $\mathcal{N}_1 \times_{\lambda, \bar{N}} \mathcal{N}_2$ is defined as the net $\mathcal{N} = (P, T, \rightarrow, P^0, \mu, \lambda, \Lambda)$ (and associated morphisms $\psi_i : \mathcal{N} \rightarrow \mathcal{N}_i$) satisfying

1. $P = \{(p_1, \star) : p_1 \in P_1\} \cup \{(\star, p_2) : p_2 \in P_2\}$, where \star means “empty,” $\psi_1(p_1, p_2) = p_1$ if $p_1 \neq \star$ and is undefined otherwise (symmetrically for ψ_2),
2. $P^0 = \psi_1^{-1}(P_1^0) \cup \psi_2^{-1}(P_2^0)$,
3. $T = \{(t_1, \star) : t_1 \in T_1, \lambda_1(t_1) \in \Lambda_1 \setminus \Lambda_2\} \cup \{(t_1, t_2) : t_1 \in T_1, t_2 \in T_2, \lambda_1(t_1) = \lambda_2(t_2) \in \Lambda_1 \cap \Lambda_2\} \cup \{(\star, t_2) : t_2 \in T_2, \lambda_2(t_2) \in \Lambda_2 \setminus \Lambda_1\}$, $\psi_1(t_1, t_2) = t_1$ if $t_1 \neq \star$ and is undefined otherwise (symmetrically for ψ_2),
4. \rightarrow is defined by $\star t = \star \psi_1(t) \cup \star \psi_2(t)$ and $t \star = \psi_1(t) \cup \psi_2(t) \star$, assuming $\star \psi_i(t) = \psi_i(t) \star = \emptyset$ if ψ_i is undefined on t ,
5. $\Lambda = \Lambda_1 \cup \Lambda_2$; λ is the obvious and unique labeling that makes ψ_1, ψ_2 label preserving morphisms; μ is the disjoint union of partitions μ_1, μ_2 .

Let $\mathcal{T}_i = (C_i, E_i, \rightarrow_i, C_i^0, \nu_i, \lambda_i, \Lambda_i)$, $i = 1, 2$, be two labeled trellis nets. Their product $\mathcal{T} = (C, E, \rightarrow, C^0, \nu, \lambda, \Lambda = \Lambda_1 \cup \Lambda_2)$ is defined by $\mathcal{T} = \mathcal{T}_1 \times_{\lambda, \bar{T}} \mathcal{T}_2 \cong \mathcal{U}^t(\mathcal{T}_1 \times_{\lambda, \bar{N}} \mathcal{T}_2)$

¹¹Notice that, even with morphisms able to duplicate places, the category *Nets* remains incomplete, for a different reason that we do not detail here. But at least *Nets* has a product.

\mathcal{T}_2). By merging the definition of $\times_{\lambda\bar{\mathcal{N}}}$ with the recursive construction of $U^t(\mathcal{N})$ for a net \mathcal{N} , one gets the following recursive form for the product $\times_{\lambda\bar{\mathcal{T}}}$:

Procedure 7

- Initialization :
 - Create $|C_1^0| + |C_2^0|$ conditions in C^0 , and define injective partial functions $\psi_i : C^0 \rightarrow C_i^0$ in such a way that they have disjoint domains.
 - Set $C = C^0, E = \emptyset, \rightarrow = \emptyset, \nu = Id$ and $\Lambda = \Lambda_1 \cup \Lambda_2$.
- Recursion :
 - Let X be a co-set of $C, \alpha \in \Lambda$ and $I = \{i : \alpha \in \Lambda_i\}$;
 $\forall i \in I$, let e_i be an event of E_i such that $\lambda_i(e_i) = \alpha$ and $\bullet e_i = \psi_i(X)$.
 - If there doesn't exist an event $e \in E$ with $\bullet e = X, \lambda(e) = \alpha$ and $\forall i \in I, \psi_i(e) = e_i$,
 - * create a new event $e \in E$ with $\bullet e = X, \lambda(e) = \alpha$ and $\forall i \in I$ set $\psi_i(e) = e_i$,
 - * create a subset Y of $\sum_{i \in I} |e_i^\bullet| = |X|$ new conditions in C , set $e^\bullet = Y$,
 - * extend the partial functions $\psi_i, i \in I$, to Y in order to have disjoint definition domains in Y and to satisfy $\psi_i : Y \rightarrow e_i^\bullet$ injective,¹²
 - * define ν on Y by $\nu(c) = \psi_i^{-1} \circ \nu_i \circ \psi_i(c)$ for the unique $i \in I$ where $\psi_i(c)$ is defined,
 - * $\forall i \in I, \forall c \in Y$, if $\exists c' \in C, \psi_i(c') = \psi_i(c)$ and $H(c') = H(c)$ then merge conditions c and c' .

The choice of creating new conditions that may disappear afterward in a merge operation is somehow inelegant. However, this formulation has a nice advantage: procedure 7 without the final merge operation actually computes the product of labeled *occurrence nets*.

Appendix D: Comparison with “merged processes”

Some authors have independently proposed the notion of *merged processes* (Khomenko et al. 2005), which shares many similarities with trellis processes. In the same way, the time unfolding of a net would be the *unraveling* of (Khomenko et al. 2005). We briefly compare these notions here, and stress some differences.

Merged processes (MP) are obtained by a partial refolding of the branching processes of a net (see the comment after Eq. 16). They are defined for general nets: places can contain several tokens, but transitions consume/produce at most one token in a given place. The start-point of the construction is a BP in the sense of Engelfriet’s definition (Engelfriet 1991). Since the latter assumes places containing at most one token, the link is established by the standard duplication trick: a place p holding k tokens in \mathcal{N} is split into k (concurrent) conditions in the unfolding, all

¹²on its domain of definition

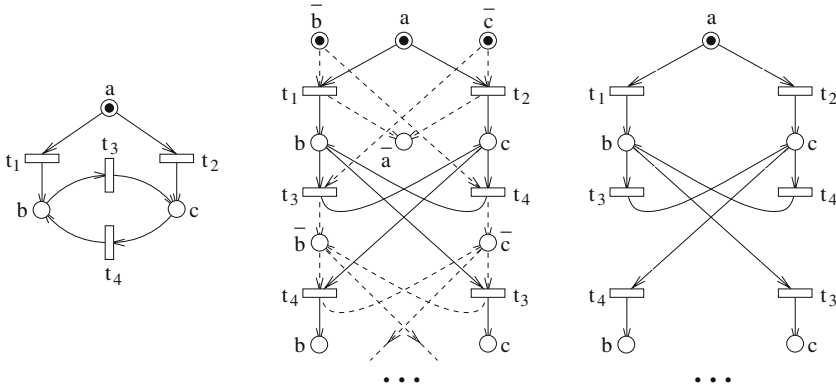


Fig. 26 A sequential machine \mathcal{N} (left). The trellis of $\tilde{\mathcal{N}}$, i.e. \mathcal{N} enriched with complementary places (center). Dashed lines are associated to the treatment of complementary conditions. The removal of complementary conditions yields the “unraveling” of \mathcal{N} (right), which contains executable cycles (e.g. circuit t_3, t_4)

labeled by p . The refolding criterion of a BP into a merged process is based on a counterpart of the height function that is called the *occurrence-depth* in (Khomenko et al. 2005). Specifically, let us limit ourselves to safe nets, for a comparison. Let $\mathcal{N} = (P, T, \rightarrow, P^0)$ be a general safe net, and consider the safe net $\tilde{\mathcal{N}} = (\tilde{P}, \tilde{T}, \rightarrow', \tilde{P}^0, \nu)$ obtained by adding a complementary place to each place of \mathcal{N} ($\tilde{\mathcal{N}}$ and \mathcal{N} have the same behavior). $\tilde{\mathcal{N}}$ becomes a multi-clock net when each place is associated to its complement, so the n places of P give rise to n sequential components in $\tilde{\mathcal{N}}$. By computing the time-unfolding of $\tilde{\mathcal{N}}$, then erasing all conditions of $\mathcal{U}'(\tilde{\mathcal{N}})$ pointing to complementary places, one obtains the unraveling of \mathcal{N} .

Despite this very tight link between the two notions, the last operation (the removal of complementary conditions) has an important effect on the resulting structure. First of all, because executable cycles are introduced. Consider for example the sequential machine \mathcal{N} of Fig. 26 (left). The trellis of $\tilde{\mathcal{N}}$ (Fig. 26, center) contains cycles, but the latter can't be executed, precisely because of the presence of complementary places. So $\mathcal{U}'(\tilde{\mathcal{N}})$ essentially contains two infinite configurations (see one of them in Fig. 16). However, when complementary conditions are removed, it becomes possible to execute these cycles (Fig. 26, right). One may object that this has little importance, since every run of the unraveling can still be mapped to a run of the original net. But the major drawback is elsewhere: the universal property of the time-unfolding is lost with the unraveling. Specifically, there exists several morphisms mapping a given configuration into the unraveling, for example the sequence (t_1, t_3) appears in two different ways in Fig. 26 (right). As a consequence, all categorical properties vanish, and in particular factorization properties.

Nevertheless, the contribution of (Khomenko et al. 2005) is of great interest, in particular for its important experimental work comparing the size of a finite complete prefix in the unfolding, to the size of its image in the unraveling. The compression factor is important on some benchmark examples, which is very promising for model-checking applications.

References

- Baroni P, Lamperti G, Pogliano P, Zanella, M (1999) Diagnosis of large active systems. *Artificial Intelligence* 110:135–183
- Benveniste A, Fabre E, Haar S, Jard C (2003) Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Transactions on Automatic Control* 48(5):714–727
- Boel RK, Jiroveanu G (2004) Distributed contextual diagnosis for very large systems. In: Proc. of WODES'04, pp 343–348
- Boel RK, van Schuppen JH (2002) Decentralized failure diagnosis for discrete event systems with costly communication between diagnosers. In: Proc. 6th Int. Workshop on Discrete Event Systems, WODES'02, pp 175–181
- Contant O, Lafortune S (2004) Diagnosis of modular discrete event systems. In: Proc. of WODES'04, pp 337–342
- Couvreur J-M, Grivet S, Poitrenaud D (2001) Unfolding of products of symmetrical petri nets. 22nd International Conference on Applications and Theory of Petri Nets (ICATPN 2001), pp 121–143. LNCS 2075
- Debouk R, Lafortune S, Teneketzis D (2000) Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Journal of Discrete Event Dynamical Systems: Theory and Application* 10(1/2):33–86
- Engelfriet J (1991) Branching processes of petri nets. *Acta Informatica*, 28:575–591
- Esparza J (1994) Model checking using net unfoldings. *Science of Computer Programming* 23:151–195
- Esparza J, Römer S, Vogler W (1996) An improvement of McMillan's unfolding algorithm. In: Proc. of TACAS'96, LNCS 1055, pp 87–106
- Esparza J, Römer S, Vogler W (2002) An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design* 20(3):285–310 (Extended version of Esparza et al (1996))
- Esparza J, Römer S (1999) An unfolding algorithm for synchronous products of transition systems. In: Proc. of CONCUR'99, LNCS 1664, Springer, Berlin Heidelberg New York
- Esparza J, Schröter C (2000) Reachability analysis using net unfoldings. Workshop of Concurrency, Specification and Programming, Volume II of Informatik-Bericht 140, pp 255–270 (Humboldt-Universität zu Berlin)
- Fabre E (2003a) Factorization of unfoldings for distributed tile systems, part 1 : limited interaction case. Inria Research Report no. 4829
- Fabre E (2003b) Convergence of the turbo algorithm for systems defined by local constraints. Inria Research Report no. PI 1510
- Fabre E (2004) Factorization of unfoldings for distributed tile systems, part 2 : general case. Inria Research Report no. 5186
- Fabre E (2005) Distributed diagnosis based on trellis processes. In: 44th Conf. on Decision and Control (CDC), Seville, Spain, pp 6329–6334
- Fabre E, Hadjicostis C (2006) A trellis notion for distributed system diagnosis with sequential semantics. In: Proc. 8th Int. Workshop on Discrete Events Systems, WODES'06, pp 294–300
- Fabre E, Benveniste A, Haar S, Jard C (2005) Distributed monitoring of concurrent and asynchronous systems. *Journal of Discrete Event Systems, special issue* 15(1):33–84
- Genc S, Lafortune S (2003) Distributed diagnosis of discrete-event systems using petri nets. In: Proc. 24th Int. Conf. on Applications and Theory of Petri Nets, LNCS 2679, pp 316–336
- Giua A, Xie X (2004) Control of safe ordinary Petri nets with marking specifications using unfolding. In: Proc. of WODES'04: 7th Workshop on Discrete Event Systems, Reims, France
- Giua A, Xie X (2005) Control of safe ordinary Petri nets using unfoldings. *Journal of Discrete Event Dynamic Systems* 15(4):349–375
- Khomenko V, Koutny M, Vogler W (2003) Canonical prefixes of petri net unfoldings. *Acta Informatica* 40:95–118
- Khomenko V, Kondratyev A, Koutny M, Vogler W (2005) Merged processes—a new condensed representation of Petri net behavior. Tech. Rep. Series CS-TR-884, Univ. of Newcastle upon Tyne
- Mac Lane S (1971) *Categories for the working mathematician*. Springer, Berlin Heidelberg New York
- McMillan KL (1992) Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: Proc. 4th Workshop of Computer Aided Verification, Montreal, pp 164–174

- McMillan KL (1993) Symbolic model checking: an approach to the state explosion problem, Ph.D. Thesis. Kluwer, Boston, MA
- Melzer S, Römer S (1997) Deadlock checking using net unfoldings. *CAV'97*, LNCS 1254, pp 352–363
- Nielsen M, Plotkin G, Winskel G (1981) Petri nets, event structures and domains. *Theoretical Computer Science* 13(1):85–108
- Reisig W (1985) *Petri Nets*. Springer, Berlin Heidelberg New York
- Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis D (1995) Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control* 40(9):1555–1575
- Sampath M, Sengupta R, Sinnamohideen K, Lafortune S, Teneketzis D (1996) Failure diagnosis using discrete event models. *IEEE Transactions on Systems Technology* 4(2):105–124
- Su R (2004) Distributed diagnosis for discrete-event systems, Ph.D. Thesis. Dept. of Elec. and Comp. Eng., Univ. of Toronto
- Su R, Wonham WM, Kurien J, Koutsoukos X (2002) Distributed diagnosis for qualitative systems. In: *Proc. 6th Int. Workshop on Discrete Event Systems, WODES'02* pp 169–174
- Vaandrager FW (1989) A simple definition for parallel composition of prime events structures. Report CS-R8903, CWI, Amsterdam
- Winskel G (1983) Event structure semantics of CCS and related languages. LNCS 140, 1982, also as report PB-159, Aarhus Univ., Denmark
- Winskel G (1984) A new definition of morphism on petri nets, LNCS 166 pp 140–149
- Winskel G (1985) Categories of models for concurrency. Seminar on Concurrency, Carnegie-Mellon Univ., pp 246–267, LNCS 197
- Winskel G (1997) Petri nets, algebras, morphisms, and compositionality. *Information and Computation*, 72:197–238
- Yoo T, Lafortune S (2002) A general architecture for decentralized supervisory control of discrete-event systems. *Journal of Discrete Event Dynamical Systems: Theory and Application*, 12(3): 335–377



Eric Fabre graduated from Ecole Nationale Supérieure des Télécommunications (ENST, Paris) in 1990, with a major orientation in Signal Processing. He obtained a Masters in Mathematics in 1993, and a PhD in Telecommunications and Signal Processing in 1994, both from the University of Rennes 1. After a postdoctoral year with Ladseb (Padua, Italy), he joined INRIA-Rennes (Irisa) in 1996.

His original interests were in multiresolution signal processing and graphical models of interactions. He then adapted the formalism of Bayesian networks to networks of dynamic systems. His current research interests follow these two related tracks: Iterative (turbo) algorithms for digital communications, and distributed monitoring algorithms for large concurrent systems. Target applications are in the field of network management, in particular distributed algorithms for autonomic communications.