Toward a Uniform Approach to the Unfolding of Nets

Eric Fabre
INRIA Rennes - Bretagne Atlantique, France
eric.fabre@inria.fr

G. Michele Pinna
Università degli Studi di Cagliari, Italy
gmpinna@unica.it

In this paper we introduce the notion of *spread* net. Spread nets are (safe) Petri nets equipped with vector clocks on places and with ticking functions on transitions, and are such that vector clocks are consistent with the ticking of transitions. Such nets generalize previous families of nets like unfoldings, merged processes and trellis processes, and can thus be used to represent runs of a net in a true concurrency semantics through an operation called the spreading of a net. By contrast with previous constructions, which may identify conflicts, spread nets allow loops in time.

1 Introduction

One of the most popular motto in Petri nets is that "*the semantics of a net is a net*" ([14]). Along this line of thought *non sequential* processes have been proposed ([8]), where the causal dependencies among the transitions of a net are faithfully represented. To model all the possible non sequential executions of nets, the notion of *unfolding* of a net has been proposed in [13] and further investigated in [16] and [6]. The idea is to represent conflicts as branching alternatives (whence the name of *Branching Processes*, that are essentially unfoldings).

Unfoldings were introduced to represent the non sequential behaviors of (safe) Petri nets, but their main application originated in the fact that they offered new techniques for the verification of concurrent systems: rather than exploring the sequential behaviors of nets the use of partial orders allows one to have more compact representations of these behaviors. Still, the data structure is in general infinite or too large. One of the first attempts to overcome this problem was to turn non sequential processes into an algebra, with a parallel composition and a suitable notion of *concatenation* ([4] and further investigated in [5]). An orthogonal approach has been the one pursued in [12] where the unfolding is *cut* in a way that still allows one to infer all the information needed to represent all possible computations of a safe net. Another way to address this problem is to define an equivalence on some behaviors of (safe) Petri nets, which implies that the data structure adopted cannot be any longer the one devised for unfoldings or prefixes. The notion of *unravel* net introduced in [3] and [2] goes in this direction requiring that each execution is a partial order, but the overall structure does not need to be a partial order. Overcoming the request that (at least locally) the behavior should be represented using a partial order has led to the introduction of a *reveal* relation playing the role of causality [1, 9]. There it is shown how to relate occurrence nets and reveal relations. Still the more compact data structure has its origin in the partial ordering representing the dependencies in the net.

In this paper we face the problem from another point of view. Rather than focussing on the properties the whole net representing the behavior of another net has to enjoy, we enrich the net with informations that will play a role analogous to those played by the properties the data structure has to fulfil.

We focus on systems (nets) that are composed of simpler subsystems: basically finite state automata. These automata *synchronize* on common transitions. The resulting system gives us the basic ingredients we want to elaborate on: causality, that coincides with *time* in each subsystem, and conflicts, which are local to a component as well. Each synchronization among finite state automata determines the expansion

M. Bartoletti and S. Knight (Eds.):
11th Interaction and Concurrency Experience (ICE 2018)
EPTCS 279, 2018, pp. 21–36, doi:10.4204/EPTCS.279.5

of conflicts and causalities to all the components of the system. When *unfolding* a net, one has to unfold completely both *time* and *conflicts*, and this yields a data structure that is generally infinite in time, and infinite also in conflicts (branchings) when choices are repeated.

These difficulties have been addressed by limiting first the time dimension: the unfolding is restricted to a finite prefix ([11]) that is sufficient (or "complete") to check the properties at stake, for example the reachability of some marking. Still the resulting data structure may be unnecessarily big, and in the last decade, merged processes ([10]) and the closely related trellis processes ([7]) were introduced to limit the expansion of the structure due to conflicts. The idea consists in merging runs that result from different choices but produce *identical* resources, where identical may mean that the same resource is produced by several alternative activities at the same time (trellis processes) or the i-th occurrence of the same resource is produced by again alternative activities (merged processes). These two approaches combined are quite successful to represent in a compact manner a sufficient set of runs of a concurrent system. However, they rely on distinct treatments for time and for conflicts.

Spread nets are nets where each place is annotated, and the annotation depends on the transitions putting a token in that place. In this way it is possible to keep track of the way that place is *reached*.

Based on this notion, in the present paper we propose the notion of spreading of nets as a unified approach to Petri net unfolding. While trellises and merged processes had abandoned the requirement that nodes should not be in self-conflict in the unfolding, the main move here is to abandon also the requirement that the unfolding should be a directed acyclic graph. In other words, we consider structures that partially unfold time, and then loop back to previously met resources. This parametric approach is flexible enough to partially or totally expand both conflicts and time, thus capturing previous constructions in a unified setting. It also assigns an equal treatment to time and conflicts. We consider structures that are just ordinary nets where places are annotated with *vector-clocks*, and these annotations gather all the information about time and conflicts.

The capability of folding time resembles the notion of concatenation introduced on non-sequential behaviors of nets, whereas the capability of folding conflicts can be considered similar to the so called *collective tokens* interpretations in Petri nets. According to this interpretation, the way a token is produced does not influence the subsequent use of it. With the capability of folding both time and conflict we allow to have that certain components of the net are executed according the individual token philosophy, whereas other parts may have different interpretations.

Structure of the paper: The paper is organized as follows. In the next section we recall the basic definitions about nets and we introduce *multi-clock* nets. In Section 3 we define the domains of information on which our spreading strategy is based. Spread nets over a suitable domain of information are then presented in Section 4. In Section 5 we first introduce the spreading operation of an multi-clock net, and then we show that the spreading of a net enjoys some nice algebraic properties similar to the ones of unfoldings and trellis processes. We also show that indeed our spreading strategy covers the ones of trellis and branching processes.

2 Nets

Notation: With $\mathbb{N}$ we denote the set of natural numbers. Let X be a set, with $|X|$ we denote the cardinality of the set. Let A be a set, a *multiset* of A is a function $f : A \to \mathbb{N}$. The usual operations on multisets, like multiset union $+$ or multiset difference $-$, are defined in the standard way. We write

$f \leq f'$ if $f(a) \leq f'(a)$ for all $a \in A$. If a multiset f is a set, *i.e.* for all $a \in A$. $f(a) \leq 1$, we confuse the multiset with the set and write $a \in f$ to indicate that $f(a) = 1$.

Given an alphabet Σ, with Σ^* we denote as usual the set of words on Σ, and with ε the empty word. The length of a word is defined as usual and, with abuse of notation, it is denoted with $|\cdot|$.

Given two mapping $f\colon A \to B$ and $g\colon B \to C$, with $f \circ g$ we denote the composition of the two mappings defined as $f \circ g(a) = g(f(a))$.

(Safe) Nets: We first review the notions of (safe) labeled Petri net and of the token game. Consider a set Σ of names.

Definition 1. *A labeled* Petri net *over Σ is a 5-tuple $N = \langle P, T, F, m, \ell, \Sigma \rangle$, where*

- *P is a set of* places *and T is a set of* transitions *(with $P \cap T = \emptyset$),*
- *$F \subseteq (P \times T) \cup (T \times P)$ is the* flow *relation,*
- *$m\colon P \to \mathbb{N}$ is called the* initial marking*, and*
- *$\ell\colon T \cup P \to \Sigma$ is a labeling mapping.*

With respect to the usual definition we have already added the labeling mapping, which is defined both on places and transitions. This will be handy when defining spread nets. Clearly ordinary Petri nets are those where the labeling is the identity (thus the transition names are the transitions themselves, and place names are the places themselves). Subscripts or superscripts on the net name carry over to the names of the net components. Given $x \in T \cup P$, ${}^\bullet x = \{y \mid (y,x) \in F\}$ and and $x^\bullet = \{y \mid (x,y) \in F\}$. ${}^\bullet x$ and $x^\bullet$ are called the *preset* and *postset* respectively of x. Observe that, given a $t \in T$, ${}^\bullet t$ and $t^\bullet$ can be seen ad multisets over P, as well as a marking m. A net $\langle P, T, F, m, \ell, \Sigma \rangle$ is as usual graphically represented as a bipartite directed graph where the nodes are the places and the transitions, and where an arc connects a place p to a transition t iff $(p,t) \in F$ and an arc connects a transition t to a place p iff $(t,p) \in F$. We assume that all nets we consider are such that $\forall t \in T$ ${}^\bullet t$ and $t^\bullet$ are not empty.

A transition t is enabled at a marking m, if m *contains* the preset of t, where contain here means that $m(p) \geq 1$ for all $p \in {}^\bullet t$, or equivalently ${}^\bullet t \leq m$. If a transition t is enabled at a marking m it may *fire* yelding a new marking defined as $m'(p) = m(p) - |{}^\bullet t \cap \{p\}| + |t^\bullet \cap \{p\}|$, or equivalently $m' = m - {}^\bullet t + t^\bullet$. The firing of t at m giving m' is denoted as $m\,[t\rangle\, m'$. The set of reachable markings of a net N is denoted with $\mathcal{M}_N$. A net N is said to be *safe* whenever its places hold at most one token in all possible reachable marking, namely $\forall m \in \mathcal{M}_N$ it holds that m can be seen as a set (the only possible values are 0 and 1). As markings may be considered as the characteristic function of a set, we will often confuse markings with subsets of places.

Net morphisms: We recall now the notion of morphism between safe nets [16].

Definition 2. *Let $N = \langle P, T, F, m, \ell, \Sigma \rangle$ and $N' = \langle P', T', F', m', \ell', \Sigma' \rangle$ be safe nets over Σ and Σ' respectively. A* morphism *$\phi : N \to N'$ is the triple $\langle \phi_T, \phi_P, \phi_\ell \rangle$, where*

- *$\phi_T : T \to T'$ is a partial function and $\phi_P \subseteq P \times P'$ is a relation such that*
  - *for each $p' \in m'$ there exists a unique $p \in m$ and $p\ \phi_P\ p'$,*
  - *if $p\ \phi_P\ p'$ then the restriction $\phi_T : {}^\bullet p \to {}^\bullet p'$ and $\phi_T : p^\bullet \to p'^\bullet$ are total functions, and*
  - *if $t' = \phi_T(t)$ then $\phi_P^{op}\colon {}^\bullet t' \to {}^\bullet t$ and $\phi_P^{op}\colon t'^\bullet \to t^\bullet$ are total functions, where ϕ_P^{op} is the opposite relation to ϕ_P, and*

- $\phi_\ell : \Sigma \to \Sigma'$ *is such that if* $\phi_T(t)$ *is defined then* $\ell'(\phi_T(t)) = \phi_\ell(\ell(t))$ *and if* $p\ \phi_P\ p'$ *then* $\ell'(p') = \phi_\ell(\ell(p))$.

The definition is the usual one, beside the last requirement which states that the labeling of the nets is preserved. We will omit the subscript when it will clear from the context, hence the triple $\langle \phi_T, \phi_P, \phi_\ell \rangle$ will be often indicated as ϕ.

Morphisms among safe nets preserve reachable markings. Consider the morphism $\phi : N \to N'$, then for each $m, m' \in \mathcal{M}_N$ and transition $t \in T$, if $m\,[t\rangle\, m'$ then $\phi_P(m)\,[\phi_T(t)\rangle\,\phi_P(m')$ provided that $\phi_T(t)$ is defined, where $\phi_P(m) = \{p' \in P' \mid \exists p \in m \text{ and } p\ \phi_P\ p'\}$.

Clearly morphisms compose and then safe nets and morphisms form a category called **Safe**.

Multi-clock nets: Safe nets can be seen as formed by various *sequential* components (automata) synchronizing on common transitions. Though this is not the usual way to consider safe nets, it is easy to see that if we add to a safe nets the so called *complemenary* places, except in the case of self-loops, we obtain a number of automata synchronizing on common transitions. A net automaton is a net in which the preset and the postset of each transition has exactly one element.

The intuition that a safe net can be viewed as a net formed by various components, each of them being a net automaton, is formalized in the notion of *multi-clock* nets, introduced by Fabre in [7].

Definition 3. *A* multi-clock *net (*mc*-net)* N *is a pair* (N, ν) *where* $N = \langle P, T, F, m, \ell, \Sigma \rangle$ *is a safe net and* $\nu : P \to m$ *is a mapping such that*

- *for all* $p, p' \in m$*, it holds that* $p \neq p'$ *implies* $\nu^{-1}(p) \cap \nu^{-1}(p') = \emptyset$*,*
- $\bigcup_{p \in m} \nu^{-1}(p) = P$*,*
- ν *is the identity when restricted to* m*, and*
- *for all* $t \in T$*.* ν *is injective on* ${}^\bullet t$ *and on* $t^\bullet$*, and* $\nu({}^\bullet t) = \nu(t^\bullet)$*.*

The dimension *of a* mc*-net* N*, denoted with* $\upsilon(\mathsf{N})$*, is the cardinality of* m*.*

The mapping ν is used to identify the various components of a *mc*-net. Given $p \in P$, with $\overline{p}$ we denote the subset of places defined by $\nu^{-1}(\nu(p))$. The consequences of three requirements, namely (a) $\nu(m) = m$, (b) ν is injective on the preset (postset) of each transition and (c) that $\nu({}^\bullet t) = \nu(t^\bullet)$, is that, for each $p \in m$, the net $\langle \overline{p}, T_{\overline{p}}, F_{\overline{p}}, \{p\}, \ell_{\overline{p}}, \Sigma \rangle$ is a net automaton, where $T_{\overline{p}}$ are the transitions of N such that $\forall t \in T_{\overline{p}}\ {}^\bullet t \cap \overline{p} \neq \emptyset$ and $t^\bullet \cap \overline{p} \neq \emptyset$, and $F_{\overline{p}}$ is the restriction of F to $\overline{p}$ and $T_{\overline{p}}$. Each place p in the initial marking can be identified with an index in $\{1, \ldots, \upsilon(\mathsf{N})\}$, hence we denote N_i as the net $\langle \overline{p}, T_{\overline{p}}, F_{\overline{p}}, \{p\} \ell_{\overline{p}} \rangle$ where i is the *index* of p. Thus the cardinality of the initial marking of a *mc*-net is the number of components forming the net, and it is the dimension of the net.

Example 1. *Consider the* mc*-net* N *in Figure 1. The* ν *in the* mc*-net* N *gives* $\nu(a) = \nu(b) = \nu(c) = \{a\}$ *and* $\nu(d) = \nu(e) = \{d\}$*. The two net automata are* N_1 *and* N_2*. The composition of the two automata (identifying the transitions with the same name, namely u and z) gives precisely* N*.*

We consider morphisms that preserve the partitions of multi-clock nets.

Definition 4. *Let* $N = (\langle P, T, F, m, \ell, \Sigma \rangle, \nu)$ *and* $N' = (\langle P', T', F', m', \ell', \Sigma' \rangle, \nu')$ *be two multi-clock nets. A* morphism $\phi : N \to N'$ *is a mcn-morphism iff* $\forall p \in P$*,* $\forall p' \in P'$*,* $p\ \phi_P\ p'$ *implies that* $\nu(p)\ \phi_P\ \nu'(p')$*.*

Multi-clock nets and mcn-morphisms form a category called **MCN**, which is a subcategory of **Safe**.

Example 2. *Consider the* mc*-nets* N *and* N_1 *in Figure 1. A mcn-morphism is the one relating places in* N *to places with the same name in* N_1*. Places d and e in* N *are not related with any place in* N_1*. The mapping on the transitions is the identity on* s, t, u, v *and z and it is undefined for w.*

In this paper, for the sake of simplicity, we will *spread* mc*-nets that are *injectively* labeled.

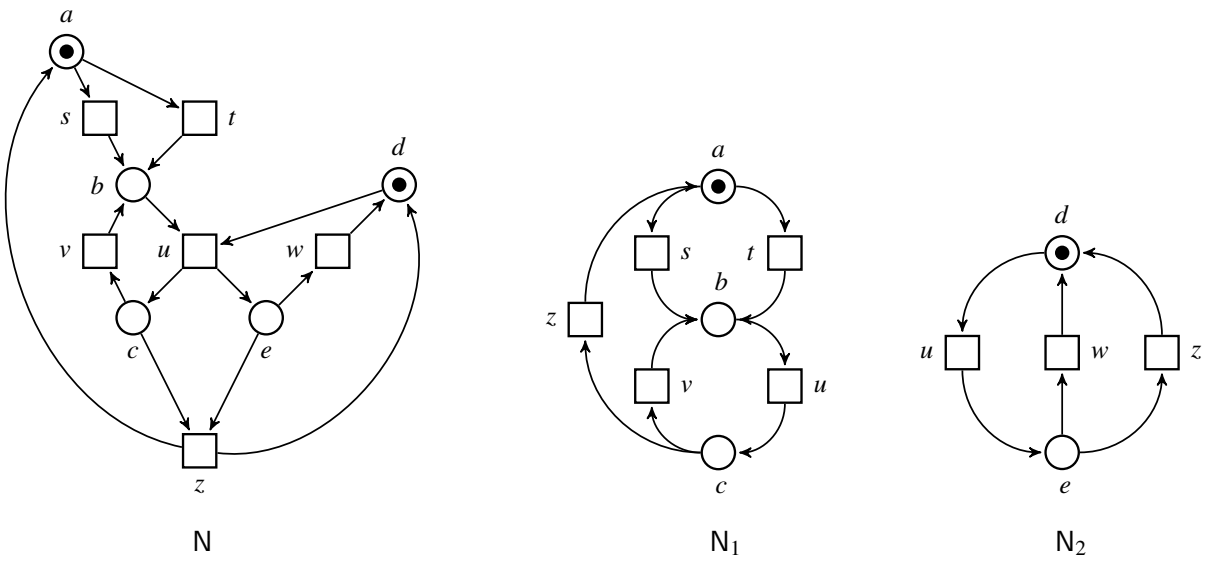

Figure 1: A *mc*-net and its components

3 Ticking domains

We introduce the *annotations* for places of the spread nets, which we will define in the next section. Annotations will be vector-clocks, where each entry of the vector will be an equivalence class of words representing the local runs of each component. This simple annotation will turn out to be powerful enough to represent most of the situations we are interested in.

Consider an alphabet A, the idea is that the elements of a ticking domain are equivalence classes of words on that alphabet. Let Eq be a set of equalities of the form $u_i = u_i'$, with $u_i, u_i' \in A^*$. We denote by $\sim_{Eq}$ the equivalence relation in A^* generated by relations in Eq and that is stable by suffix extension, i.e. $u \sim_{Eq} u' \Rightarrow uv \sim_{Eq} u'v$ for $u, u', v \in A^*$. Relation $\sim_{Eq}$ will simply be denoted $\sim$ when the generating set is clear from the context, and equivalence classes are denoted as $(|w|)_\sim$ or simply $(|w|)$. For the sake of light notations, in the sequel we will often confuse w with its class $(|w|)$. Observe that one has $(|\,(|u|)v\,|) = (|uv|)$.

Definition 5. *Let A be an alphabet and let Eq be a set of equalities of the form $\alpha = \beta$, with $\alpha, \beta \in A^*$. Then a* ticking domain *over A is the set of equivalence classes of words in A^* with respect to the suffix stable equivalence relation $\sim_{Eq}$ generated by Eq, and it is denoted $\mathcal{A}_{Eq} = A^*/\sim_{Eq}$. With $\mathsf{alph}(\mathcal{A}_{Eq})$ we denote the alphabet A.*

Example 3. *Consider the* mc-net N_1 *in Figure 1. The alphabet can be considered the name of the transitions (hence $A = \{s,t,u,v,z\}$) and we may imagine the following equations: $s = t$, $suvs = s$ and $suzs = s$. These equations induce, among others, the following equivalence classes on the word representing some executions of the* mc-net N_1*: $(|\varepsilon|)_{\sim_1}$, $(|s|)_{\sim_1}$, $(|su|)_{\sim_1}$, $(|suv|)_{\sim_1}$ and $(|suz|)_{\sim_1}$ (the other equivalence classes may be ignored, as it will become clear in the following). These equivalence classes form a ticking domain for the state-machine net* N_1.

Another set of equations over $A = \{s,t,u,v,z\}^*$ can be the following: for all $w, w' \in \{s,t,u,v,z\}$, $w = w'$ iff $|w| = |w'|$. In this case two words are in the same equivalence class iff they have the same length. Requiring that the equivalence of words is also a congruence, the same set of equivalence classes

could be obtained from the set of equations $u = v$ with $u, v \in \{s, t, u, v, z\}$.

If the set of equations is empty then each word $w \in A^*$ is the unique member of the equivalence class $(|w|)$. Being the equivalence relation stable with respect to suffix, the same can be obtained using as the set of equations $u = u$ with $u \in \{s, t, u, v, z\}$.

Given two ticking domains $\mathcal{A}_{Eq}$ and $\mathcal{A}'_{Eq'}$, $\delta\colon \mathcal{A}_{Eq} \to \mathcal{A}'_{Eq'}$ is a ticking domain mapping iff given any two words $w, w' \in A^*$ in the same equivalence class in $\mathcal{A}_{Eq}$, then $\delta(w), \delta(w')$ are in the same equivalence class in $\mathcal{A}'_{Eq'}$.

We are now ready to introduce the notion of *vector-clock*.

Definition 6. *Given a set of index I and a set of ticking domains $\mathcal{A}_i$, with $i \in I$, a* vector-clock $\overrightarrow{\alpha}$ *is an element of $\times_{i\in I}\mathcal{A}_i$, and $\mathcal{A} = \times_{i\in I}\mathcal{A}_i$ is called the* vector-clock domain *(*VCD *for short). The* dimension *of the vector-clock domain $\mathcal{A}$, denoted with $\iota(\mathcal{A})$, is given by $|I|$.*

The $\times$ on clock domains is associative and can be easily extended to an operation $\times$ on vector-clock domains as a component-wise operation.

Vector clock elements can be *mixed* to obtain a new vector clock element. The intuition is that the new element is obtained from the previous one selecting entries from each of them. This is formally stated in the next definition.

Definition 7. *Let J be a set of index and let $\Gamma = \{\alpha_j \mid j \in J \wedge \alpha_j \in \mathcal{A}\}$ be a set of vector clock in $\mathcal{A}$. Then $op^k_J\colon \mathcal{A}^{|J|} \to \mathcal{A}$, with $k \in J$, is an operation defined as follows: the i-th entry of $op^k_J(\Gamma)$ is the i-th entry of the vector clock $\alpha_i \in \Gamma$ if $i \in J$ and of $\alpha_k \in \Gamma$ otherwise.*

We briefly discuss the intuition behind this definition. The various components of a vector-clock represent the pieces of information each component has on its behavior and on the other components behaviors as well. The various pieces of information have to be combined together to form a new vector clock, which will be the argument of a function of a spread net. The way of combining the information should take into account mainly the information associated to a certain set of indexes, as it will be again clear when we will introduce the notion of spread net.

Clearly these pieces of information have to be consistent, and the operations defined above are responsible in assuring this.

Example 4. *Take $\Gamma = \{(w_1^{j_1}, w_2^{j_1}, w_3^{j_1}), (w_1^{j_2}, w_2^{j_2}, w_2^{j_2})\}$, and $op^{j_1}_J$, where $J = \{j_1, j_2\}$. Then $op^{j_1}_J(\Gamma)$ is $(w_1^{j_1}, w_2^{j_2}, w_3^{j_1})$, whereas $op^{j_2}_J(\Gamma)$ is $(w_1^{j_1}, w_2^{j_2}, w_3^{j_2})$.*

4 Spread Nets

We enrich *mc*-nets with vector clocks. The idea is that each place of a *mc*-net N of dimension $\upsilon(\mathsf{N})$ has associated a vector-clock belonging to a vector clock domain $\mathcal{A}$ such that $\iota(\mathcal{A}) = \upsilon(\mathsf{N})$. Thus in general the annotation of a place of a *mc*-net carries information on the component the place belongs to (the proper entry in the vector clock), but it may also convey information about the other components (the other entries of the vector clock).

We start by illustrating this idea with a little example.

Example 5. *Consider the* VCD *$\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$, where $\mathcal{A}_1 = \{(|\varepsilon|)_{\sim_1}, (|v|)_{\sim_1}, (|u|)_{\sim_1}, (|us|)_{\sim_1}\}$, with $\sim_1$ obtained by the equations $uu = u$, $usv = us$, $usu = us$, $uss = us$, $vs = v$, $vu = v$, $vv = v$ and $s = \varepsilon$, $\mathcal{A}_2 = \{(|\varepsilon|)_{\sim_2}, (|u|)_{\sim_2}, (|us|)_{\sim_2}\}$, with $\sim_2$ induced by the set of equations $Eq_2 = \{u = w, us = ws, s = \varepsilon, su = \varepsilon, sw = \varepsilon, uu = u, uw = u\}$, and $\mathcal{A}_3 = \{(|\varepsilon|)_{\sim_3}\}$, with $\sim_3$ induced from the set $Eq_3 = \{\varepsilon = w\}$.*

Consider now the mc*-net in Figure 2.*

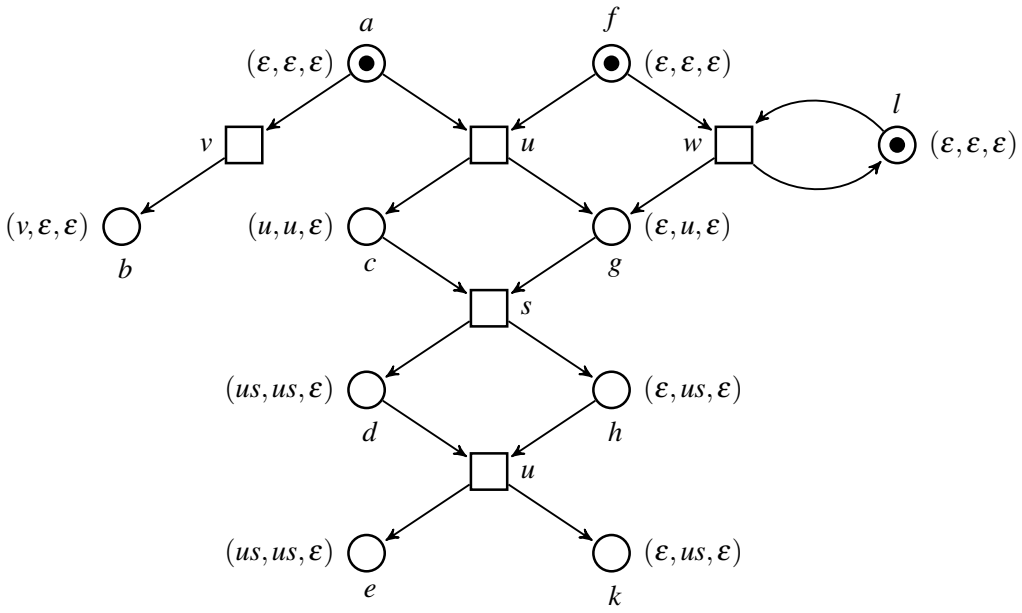

Figure 2: A net over an information domain

The components of this mc*-net are identified by the partition mapping defined as follows:* $\nu^{-1}(a) = \{a,b,c,d,e\}$, $\nu^{-1}(f) = \{f,g,h,k\}$ *and* $\nu^{-1}(l) = \{l\}$. *For the ticking domains* $\mathcal{A}_1, \mathcal{A}_2$ *and* $\mathcal{A}_3$ *the alphabets on which they are based is the set of transitions of each component, thus* $\mathsf{alph}(\mathcal{A}_1) = \{v,u,s\}$, $\mathsf{alph}(\mathcal{A}_2) = \{u,s,w\}$ *and* $\mathsf{alph}(\mathcal{A}_2) = \{w\}$. *The vector clocks associated to the places of this* mc*-net are the ones shown in the figure: to* a, f *and* l *the vector clock associated is* $(\varepsilon,\varepsilon,\varepsilon)$, *to place* b *the vector clock* $(v,\varepsilon,\varepsilon)$, *to* c *the vector clock* (u,u,ε), *to* f *the vector-clock* (u,u,ε), *to* d *and* e *the* (us,us,ε) *and finally to* h *and* k *the vector-clock* $(\varepsilon,us,\varepsilon)$.

***mc*-net over $\mathcal{A}$:** We first introduce the notion of *mc*-net over a domain and then we will formalize the one of spread net.

Definition 8. *The pair* $\mathcal{N} = ((\langle P,T,F,m,\ell,\Sigma\rangle,\nu),h\colon P \to \mathcal{A})$, *where*

- $\mathsf{N} = (\langle P,T,F,m,\ell,\Sigma\rangle,\nu)$ *is a* mc*-net,*
- $\mathcal{A}$ *is a vector clock domain such that* $\mathcal{A} = \times_{i=1}^{\upsilon(\mathsf{N})}\mathcal{A}_i$, *where* $\mathsf{alph}(\mathcal{A}_i) = \ell(T_i)$ *with* $1 \le i \le \upsilon(\mathsf{N})$, *and*
- $h\colon P \to \mathcal{A}$ *is a total mapping and it is called the* information *mapping.*

is called a mc*-net over* $\mathcal{A}$. *With* $\mathsf{td}(\mathcal{N})$ *we denote the vector clock domain* $\mathcal{A}$.

Spread nets: We assume, for each $\mathcal{A}_i$, that the set of equations Eq_i on words over the alphabet $\mathsf{alph}(\mathcal{A}_i)^*$ such that $\mathcal{A}_i = \mathsf{alph}(\mathcal{A}_i)^*/\sim_{Eq_i}$, is well understood. A *mc*-net over a VCD is then an annotated net, where the annotations are on places. These annotations are of a specific kind, namely they are vector-clocks where each component of the vector is an equivalence class of words on given alphabets. Based on this notion we can introduce the notion of *spread* net, where the annotations of places are *calculated*.

Definition 9. *Let* N *be a* mc*-net and* $\mathcal{A}$ *be a vector clock domain such that* $\iota(\mathcal{A}) = \upsilon(\mathsf{N})$*. Let* $\mathcal{S} = (\mathsf{N}, h\colon P \to \mathcal{A})$ *be a* mc*-net over* $\mathcal{A}$*. Let* $\vec{\tau} = \{\tau_i \mid 1 \leq i \leq \upsilon(\mathsf{N})\}$ *be a set of* ticking *mapping with* $\tau_i : \mathcal{A} \times T_i \to \mathcal{A}$ *Then* $\mathcal{S}$ *is a* spread *net with respect to* $\vec{\tau}$ *and* $\mathcal{A}$ *iff*

- $\forall p.\ m(p) = 1$ *it holds that* $h(p) = (\varepsilon, \ldots, \varepsilon)$*,*
- $\forall p, p' \in P.\ \ell(p) = \ell(p')$ *and* $h(p) = h(p')$ *implies* $p = p'$*, and*
- $\forall t \in T.\ \forall p \in t^\bullet\ h(p) = \tau_i(op^{\nu(p)}_{\nu(\bullet t)}(\{h(p') \mid p' \in {}^\bullet t\}), t)$*.*

With $\mathsf{support}(\mathcal{S})$ *we denote the* mc*-net* N*.*

A spread net is a *mc*-net over a specific domain where the annotations on the places in the postset of a transition t are related to annotations of the places in the preset of this transition and the transition t itself. The annotations are a mean to keep track on how a place can be *reached*. Thus we require that the annotations of the places in the initial marking is $(\varepsilon, \ldots, \varepsilon)$, and the annotations on the places in the postset of a transition are calculated on the basis of the annotations in the preset of this transition (combined using the operations according to Definition 7): the annotations of the places in the component i are *calculated* using a function τ_i which is *local* to the component itself, though the $op^{\nu(p)}_{\nu(\bullet t)}$ may not be local at all. The requirement that $\forall p, p' \in P.\ \ell(p) = \ell(p')$ and $h(p) = h(p')$ implies $p = p'$ implies that if two equally labeled places have the same information, then they are indeed the same place. Indeed two equally labeled places represent the same activity and if the annotation is the same then, despite the various possible alternatives that may have produced them, they should not be distinguished and hence are the same. This is a *succinctness* principle that avoid that the same information is associated to different places representing the same resource.

Example 6. *Consider again the net in Figure 2. Assume that the alphabets for the various components are the transition labels (in this case, as the net is injectively labeled, these coincide with the transitions themselves).*

Take τ_1 *as the mapping that concatenatee the transition t to each entry i of the vector clock such that* $t \in \mathsf{alph}(\mathcal{A}_i)$ *and leave the other entries untouched,* τ_2 *adds the transition t just to the second component and finally* τ_3 *is the constant mapping giving* $(\varepsilon, \varepsilon, \varepsilon)$*.*

When executing each transition of the net we assume that the vector clocks associated to the place in the postset of a transition are calculated *from the vector clocks associated to the places in the preset of this transition, and that this is done* locally*. Thus, when executing v, the vector clock* $(\varepsilon, \varepsilon, \varepsilon)$ *of a is used to obtain* $(v, \varepsilon, \varepsilon)$ *on b, or when executing u the vectors clocks* $(\varepsilon, \varepsilon, \varepsilon)$ *associated to a and* $(\varepsilon, \varepsilon, \varepsilon)$ *associated to e are used to obtain* (u, u, ε) *for c and* $(\varepsilon, w, \varepsilon)$ *for f, recalling that* $u \sim_2 w$*. The two vector clocks in* ${}^\bullet u$ *are* merged *together, by selecting the proper components. The vector-clocks associated to g and d are obtained first calculating a vector clock from the one in* ${}^\bullet s$*, and* (u, u, ε) *and* $(\varepsilon, u, \varepsilon)$ *are merged obtaining* (u, u, ε) *and then d gets* (us, us, ε) *whereas g gets the vector clock* $(\varepsilon, us, \varepsilon)$*.*

Morphisms: We specialize to this new setting the notion of morphism:

Definition 10. *Let* $\mathcal{S} = (\mathsf{N}, h\colon P \to \mathcal{A})$ *and* $\mathcal{S}' = (\mathsf{N}', h'\colon P' \to \mathcal{A}')$ *be two spread nets,* $\mathcal{S}$ *over* $\vec{\tau}$ *and* $\mathcal{A}$ *and* $\mathcal{S}'$ *over* $\vec{\tau}'$ *and* $\mathcal{A}'$ *respectively. A* spread*-morphism* $f : \mathcal{S} \to \mathcal{S}$ *is a pair* $f = (\phi, \delta)$ *where*

- $\phi\colon \mathsf{N} \to \mathsf{N}'$ *is a mcn-morphism,*
- $\delta : \mathcal{A} \to \mathcal{A}'$ *is a mapping such that for each* $\tau_i \in \vec{\tau}$ *and* $\tau'_i \in \vec{\tau}'$ *it holds that* $\tau'_i(\delta(\alpha), \phi_T(t)) = \delta(\tau_i(\alpha, t))$ *whenever* $\phi_T(t)$ *is defined and* $t \in T_i$*, and*
- $\delta(h(p)) = h'(p')$ *whenever* $p\ \phi_P\ p'$*.*

We show that spread-morphisms compose. Let $f = (\phi, \delta) : \mathcal{S} \to \mathcal{S}'$ and $g = (\phi', \delta') : \mathcal{S}' \to \mathcal{S}''$ two ND-morphisms. $f \circ g : \mathcal{S} \to \mathcal{S}''$ defined as $(\phi \circ \phi', \delta \circ \delta')$ is a well defined ND-morphism. The only condition to check is that $\delta \circ \delta'(\tau_i(\alpha,t))$ is defined whenever also $\phi_T \circ \phi'_T(t)$ is defined, and it is equal to $\tau''_i(\delta \circ \delta'(\alpha), \phi_T \circ \phi'_T(t)(t))$. Now $\delta \circ \delta'(\tau_i(\alpha,t)) = \delta'(\delta(\tau_i(\alpha,t)))$ and this is equal to $\delta'(\tau'_i(\delta(\alpha), \phi_T(t)))$ and finally also to $\tau''_i(\delta'(\delta(\alpha)), \phi'_T(\phi_T(t)))$ which is $\tau_i(\delta \circ \delta'(\alpha), \phi_T \circ \phi'_T(t))$ as required. Clearly we have that $\delta \circ \delta'(h(p)) = \delta'(\delta(h(p))) = \delta'(h(p')) = h'(p'')$ with $p\ \phi_P\ p'$ and $p'\ \phi_P\ p''$. Thus spread nets and spread-morphisms form a category, that we call **Spread**.

This category is related to the one of *mc*-nets via two obvious functors. One takes an object $\mathcal{S}$ in **Spread** and returns the *mc*-net $\mathsf{support}(\mathcal{S})$, and we call it $\mathfrak{F}$, the other takes a *mc*-net N where the labeling ℓ is injective and associate the spread net $\mathfrak{G}(\mathsf{N})$ over $\vec{\tau} = \{\tau_i \mid \tau_i$ is the constant mapping returning $(\varepsilon, \ldots, \varepsilon)\}$ and $\mathcal{A}_\perp$ defined as $(\mathsf{N}, h\colon P \to \{(\varepsilon, \ldots, \varepsilon)\})$ (thus all the places are annotated with the vector-clock $(\varepsilon, \ldots, \varepsilon)$), and each mcn-morphism ϕ gives $\mathfrak{G}(\phi) = (\phi, id)$.

When it will be clear from the context, we will omit to mention both the $\vec{\tau}$ and $\mathcal{A}$ on which a spread net is based on.

Configuration: We end this section by defining what a *configuration* of a spread net is. This notion will be used when spreading a net, and it is the usual one adapted to the context of spread nets.

Definition 11. *Let $\mathcal{S} = (\mathsf{N}, h\colon P \to \mathcal{A})$ be a spread net, and $m\,[t_1\rangle\, m_1 \ldots n_{n-1}\,[t_n\rangle\, m_n$ be a firing sequence in N. Then a configuration is the multiset $\sum_{i=1}^{n}\{t_i\}$.*

Configurations are ranged over with C and $m_n = mark(C)$ is marking reached executing the firing sequence associated to the configuration. The set of configuration of a spread net $\mathcal{S}$ is denoted with $\mathcal{C}onf\,(\mathcal{S})$.

5 Spreading nets

In this section we describe how to spread *mc*-nets. We assume that the labeling mapping of the net that should be spread is the identity.

First we recall what a *folding*-morphism is. Let N and N' be *mc*-nets. $\phi\colon \mathsf{N} \to \mathsf{N}'$ is a folding morphism iff

- ϕ is total,
- $\forall t, t' \in T.\ ({}^\bullet t = {}^\bullet t' \wedge \phi_T(t) = \phi_T(t')) \Rightarrow t = t'$.

These requirements are standard for folding morphisms. A folding what it does is to fold entirely a *mc*-net onto another (the requirement of totality of the mapping) and it does in an economical way, as transitions that are not distinguishable in the target net should be the same transition.

The algorithm will construct a spread net and also a morphism that will turn out to be a folding morphism.

Spreading algorithm: We spread a *mc*-net with respect to a certain domain $\mathcal{A}$ of information inferred by the net itself and a set of $\vec{\tau}$ of ticking mappings that obey to a schema (which basically states how conflicts are spread through the various components). In fact, as it will become clear in the following, the schema for the ticking mappings can be seen as a *parameter* of the spreading, and it simply state how time is counted in each component, also in relation with the other components.

Input: A *mc*-net $\mathsf{N} = (\langle P,T,F,m,\ell,\Sigma\rangle,\nu)$ of dimension $\upsilon(\mathsf{N})$, a VCD $\mathcal{A}$ of dimension $\imath(\mathcal{A}) = \upsilon(\mathsf{N})$ such that for each $i \in \upsilon(\mathsf{N})$. $\mathsf{alph}(\mathcal{A}_i) = \ell(T_i)$, a set $\vec{\tau}$ of ticking mapping and a set of operations op_J^k satisfying the requirements of Definition 7

Output: At each step a spread net $\mathcal{O}$ and a folding mapping ϕ onto N

Initialization step: Create $|m|$ places for O and define a bijection $\phi_P\colon m_O \to m$. Define, for each $p \in m_O$, $h_O(p) = (\varepsilon,\ldots,\varepsilon)$, and set $O = \langle m_O,\emptyset,\emptyset,m_O,\ell_O,P\cup T\rangle$ with $\ell_O(p) = \phi_P(p)$, obtaining the and *mc*-net $\mathsf{O} = (O,\nu_O)$ where $\nu_O(p) = \nu(\phi_P(p))$. Finally set $\mathcal{O} = (\mathsf{O},h_O)$. The ϕ mapping has just the component on places. Output $(\mathcal{O},\phi)$.

Recursion: Consider the spread net constructed so far $\mathcal{O} = (\mathsf{O},h_O)$ and the mapping ϕ.
Let C be a configuration of $\mathsf{O} = (O,\nu_O)$, with $O = \langle P_O,T_O,F_O,m_O,\ell_O,P\cup T\rangle$, and consider $\hat{m} = \phi_P(mark(C))$. Let $t \in T$ be a transition such that ${}^\bullet t \subseteq \hat{m}$. Check if T_0 contains a transition t' such that ${}^\bullet t' \subseteq mark(C)$ and $\phi_T(t') = t$. If yes consider another configuration, if not then

- add t' to T_O and set $\phi'_T(t') = t$ and $\phi'_T(t'') = \phi_T(t'')$ for all $t'' \in T_O$,
- add to F_O the set $F'_O = \{(p',t') \mid p' \in mark(C) \wedge \phi_P(p') \in {}^\bullet t\}$,
- for each $p \in t^\bullet$, check if there is a place $p' \in P_O$ such that
  - $\phi'_P(p') = p$ and
  - $h'(p') = \tau_{\nu_O(p')}(op_J^{\nu_O(p')}(\{h(p'') \mid p'' \in mark(C) \wedge \phi_P(p'') \in {}^\bullet t\}),\ell'(t))$, where $J = \{\nu_O(p'') \mid p'' \in mark(C) \wedge \phi_P(p'') \in {}^\bullet t\}$.

  If yes, then simply add (t',p') to F'_O. If not then create a place p', set $h'(p') = \tau_{\nu_O(p')}(op_J^{\nu_O(p')}(\{h(p'') \mid p'' \in mark(C) \wedge \phi_P(p'') \in {}^\bullet t\}),\ell'(t))$, where $J = \{\nu_O(p'') \mid p'' \in mark(C) \wedge \phi_P(p'') \in {}^\bullet t\}$ add it to P_O. Add (t',p') to F'_O as well,
- extend ϕ_P by setting $\phi'_P(p') = p$, and
- set $\nu_O(p') = \phi_P^{-1}(\nu_O(p))$

Let P' the set of the new added places, let $O' = \langle P_O \cup P', T \cup \{t'\}, F_0 \cup F'_O, m_O, \ell'_O, P \cup T\rangle$ with $\ell'_O(x) = \ell_O(x)$ for $x \in P_O \cup T_O$, $\ell'_O(t') = t$ and $\ell'_O(p) = \phi_P(p)$ for each $p \in P'$, and $\nu'_O(p) = \nu(\phi_P(p))$ for $p \in P'$ and $\nu'_O(p) = \nu_O(p)$ for $p \in P_O$.
Output $\mathcal{O} = ((O',\nu'_O),h')$ and ϕ'.

Figure 3: The spreading algorithm

Proposition 1. *Let* $\mathsf{N} = (\langle P,T,F,m,\ell,\Sigma\rangle,\nu)$ *be a mc-net of dimension* $\upsilon(\mathsf{N})$ *such that* $\ell\colon T \to \Sigma$ *is total and injective. For each* $i \in \{1,\ldots,\upsilon(\mathsf{N})\}$ *let* Eq_i *be a set of equations on* $\ell(T_i)^*$*, where* T_i *are the transitions of the i-th component of* N *and* $\mathcal{A}_i = \ell(T_i)^*/\sim_{Eq_i}$*. Let* $\mathcal{A} = \times_{i=1}^{\upsilon(\mathsf{N})}\mathcal{A}_i$ *and let* $\vec{\tau} = \{\tau_i \mid 1 \le i \le \upsilon(\mathsf{N})\}$ *be a set of* ticking *mapping with* $\tau_i : \mathcal{A}\times T_i \to \mathcal{A}$*. Then the algorithm in Figure 3 produces a spreading net* $\mathfrak{S}^{\mathcal{A}}_{\vec{\tau}}(\mathsf{N}) = (\mathsf{O},h)$ *and a folding morphism* $\phi\colon \mathsf{support}(\mathfrak{S}(\mathsf{N})) \to \mathsf{N}$.

It is quite obvious that the algorithm define a spread net and a folding morphism as well. Observe

that in the algorithm in Figure 3 we could have done the recursion step a bit differently, namely for each t' added such that $\phi_T(t') = t$, we could have added $|t^\bullet|$ places, and then some of them could be glued with some others in the spread net constructed so far provided that they are related to the same place in N and have the same vector-clock annotation. This alternative guarantees the fact that the morphism constructed is a folding one.

We want to stress that, depending on the vector-clock domain, the algorithm produce a *finite* data structure (a finite spread net). Indeed, if the elements of the vector-clock domain are finite the spread net constructed is finite as well, due to the way the labeling ℓ is defined when constructing the spread net.

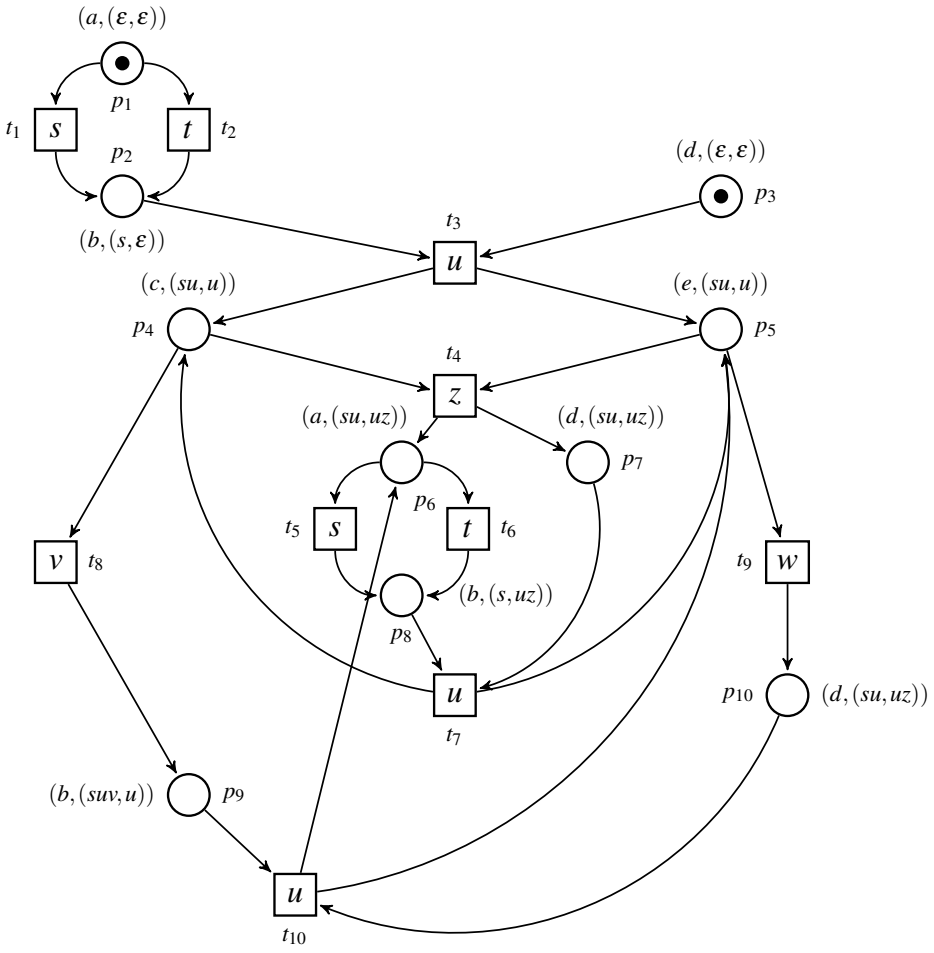

Figure 4: A spread net

Example 7. *Consider the net in Figure 4. This is the spreading of the* mc*-net in Figure 1 according to the* τ*s ticking mappings and to the vector-clock domain as described in the following.*

The ticking domain for the first component (the one on the left) is the with alphabet $\mathsf{alph}(\mathcal{A}_1) = \{s,t,u,v,z\}$ *and the equations are* $\varepsilon = \varepsilon$*,* $s = t$*,* $su = tu$*,* $suz = su$*,* $tuz = su$*,* $sus = s$*,* $tus = s$*,* $sut = s$*,* $tut = s$*,* $suvu = su$*,* $tuvu = su$*,* $u = \varepsilon$*,* $v = \varepsilon$*,* $ss = s$*,* $ts = s$ *and* $tt = t$ *and* $x = \varepsilon$ *for each* $x \in \mathsf{alph}(\mathcal{A}_1)^*$ $|x| \geq 4$ *and* $x \neq suvu$ *and* $tuvu$*. The equivalence classes obtained are* $(|\varepsilon|)_{\sim_1}$*,* $(|s|)_{\sim_1}$*,* $(|su|)_{\sim_1}$*,* $(|suv|)_{\sim_1}$ *and* $(|suz|)_{\sim_1}$*, and these form the ticking domain* $\mathcal{A}_1$*. Concerning the ticking domain for the second component, the*

alphabet is $\mathsf{alph}(\mathcal{A}_2) = \{u, w, z\}$ *the equivalence relation is based on the following equations:* $uwu = uz$, $\varepsilon = \varepsilon$ *and for each other word* x *in* $\mathsf{alph}(\mathcal{A}_2)^*$ *beside the ones involved in these equations, we have* $x = \varepsilon$. *The equivalence classes we obtain are* $(|\varepsilon|)_{\sim_2}, (|u|)_{\sim_2}$ *and* $(|uz|)_{\sim_2}$, *which are the elements of* $\mathcal{A}_2$. *Equivalence classes are identified with their representative.*

The operations τ_i *(with* $i \in \{1,2\}$*) take a vector-clock and concatenate each word with label* $\ell(t)$, *provided that* $\ell(t)$ *appears in the alphabet, thus* $\tau_1((su,u),t_8) = (suv,u)$ *as* $\ell(t_8) = v$ *is in the alphabet of the ticking domain* $\mathcal{A}_1$ *but not in the alphabet of* $\mathcal{A}_2$, *and* $\tau_1((su,u),t_8) = (suz,uz)$ *as* $\ell(t_{10}) = z$ *belongs to both alphabets.*

The operations op (we omit the indexes as it is clear what they do) if applied to just one vector return the same vector, otherwise the first component of the resulting vector comes from the first one and the second component from the second one.

In the figure places are annotated with the pair (p, α) *where* p *is the name of the place in the net* N *in Figure 1 and* $\alpha \in \mathcal{A}_1 \times \mathcal{A}_2$. *The first component of the pair is the* ℓ *mapping and the second is the* h *mapping* $h\colon \{p_1, \ldots, p_{10}\} \to \mathcal{A}_1 \times \mathcal{A}_2$ *defined as follows:* $h(p_1) = (\varepsilon,\varepsilon) = h(p_3), h(p_2) = (s,\varepsilon), h(p_4) = h(p_5) = (su,u)$, $h(p_6) = (suz,uz) = h(p_7), h(p_8) = (s,uz), h(p_9) = (suv,u)$ *and* $h(p_{10}) = (su,uw)$.

Observe that the spread net is finite as the VCD is finite and ℓ maps places and transitions of the spread net onto a finite set. Another spread net over the same domain with the same ticking mappings does not need to be finite, provided that the ℓ mapping has an infinite codomain.

The nice property that the spreading of a net enjoys is that it is indeed a universal construction.

Theorem 1. *Let* N *be an mc-net, then for each* $\mathcal{S}$ *spread net with respect to* $\vec{\tau}$ *and* $\mathcal{A}$ *and morphism* $g\colon \mathsf{support}(\mathcal{S}) \to \mathsf{N}$, *there exists a unique morphism* $l\colon \mathcal{S} \to \mathfrak{S}^{\mathcal{A}}_{\vec{\tau}}(\mathsf{N})$ *such that* $g = \mathfrak{F}(l) \circ \phi$.

The theorem implies that the spreading of *mc*-net with respect to a given $\vec{\tau}$ and $\mathcal{A}$, is somehow the *best* construction with these characteristic we can aim at. The fact that it is the best construction depends on the way the spreading is performed not only for the annotation of places but also for the labeling of them. It is then quite obvious that any other spread net which is mapped onto the one to be spread, should have a different labeling has it cannot have a different annotation.

We substantiate our claim showing that this new notion covers various notion of unfoldings. We will consider here just branching processes and trellis processes (for the proper definitions we refer to [6, 16] for branching processes and [7] for trellis processes).

Branching Processes: The ticking domain to be considered in this case is, for each component, the one induced by the set of equations containing just $\varepsilon = \varepsilon$, and the alphabet of each ticking domain are the transitions of the component. The result is that each equivalence class contains just a word. We call the resulting vector-clock domain $\mathcal{A}_{BP}$. The τ_i add the transition to the words in the entries of the vector-clock that are involved in the synchronization, and we call these τ_i as $\vec{\tau}_{BP}$. The operations *op* take the words belonging to the components synchronizing, shuffle them taking into account the synchronization transitions, and produces a new vector-clock where each entry is the word obtained projecting on the proper alphabet the word obtained as described above. This convey the intuition that from a given transition, there is a unique path to the places in the initial marking, which is the one of a causal net ([16]).

We can state the following result, where, $\mathcal{U}_{BP}(\mathsf{N})$ is the branching process obtained by the *mc*-net N.

Proposition 2. *Let* N *be a mc-net, and let* $\mathfrak{S}^{\mathcal{A}_{BP}}_{\vec{\tau}_{BP}}(\mathsf{N})$ *be its spreading with respect to* $\vec{\tau}_{BP}$ *and* $\mathcal{A}_{BP}$.

Then $\mathsf{support}(\mathfrak{S}^{\mathcal{A}_{BP}}_{\vec{\tau}_{BP}}(\mathsf{N}))$ *is isomorphic to* $\mathcal{U}_{BP}(\mathsf{N})$.

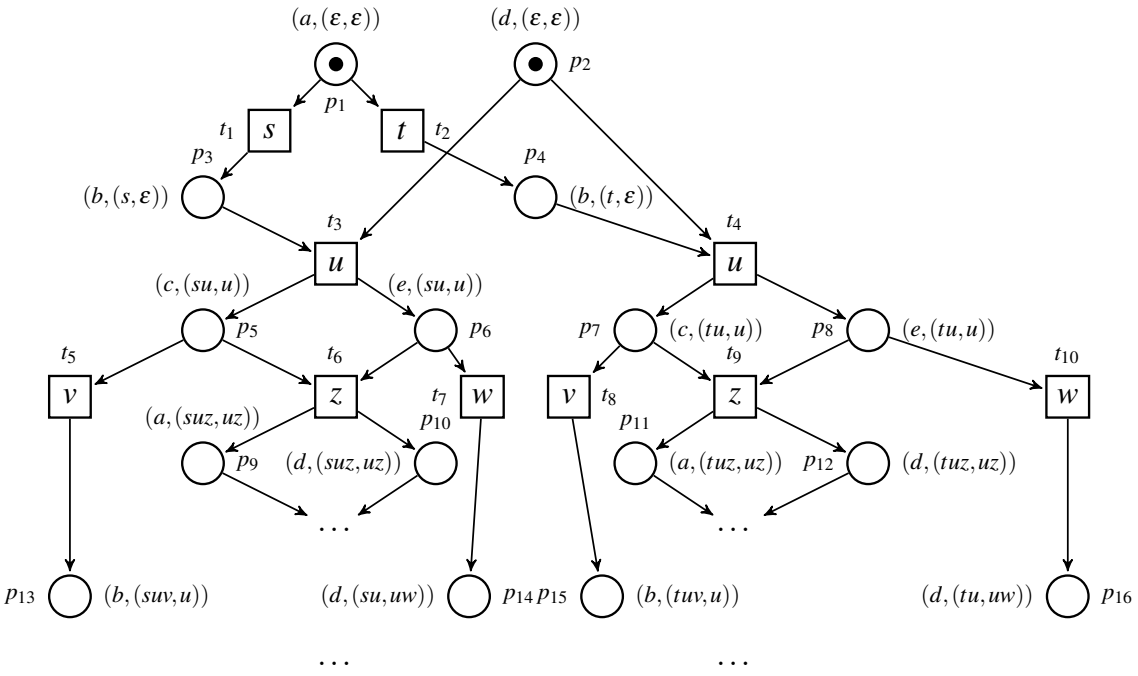

Figure 5: The initial part of the Branching process of the *mc*-net $\mathcal{N}$ in Figure 1

Example 8. *Figure 5 shows just the first part of the spreading of* N *according to the* $\mathcal{A}_{BP}$ *domain. The object constructed in this way is, as expected, infinite. Conflicts are inherited along the causality paths (executions in each automata) and the* quantity *of information associated to each place in this spreading increases. The annotation of a place contains, for each components, the trace in this component leading to that place. For instance, consider the place* p_9*. The annotation of* p_9 *is* (suz,uz)*. In fact the two components of the net synchronize first on u and then on z, in the first component the first transition executed is s whereas the second component should synchronize. Hence the annotation regarding the first component is suz and the one regarding the second component is uz.*

Trellises: Here the ticking domain for each component is the one induced by the following set of equations: $u = v$ for all u, v words with the same length on the alphabet such that they correspond to a firing sequence in the component ending in the same place (as the length counts there is a difference with the words ending in the same place). Thus two words w and w' are equivalent iff they have the same length and if they put a token in the same place. We call this domain $\mathcal{A}_{Tr}$.

The τ_i work as follows: each of them receives in input a vector-clock and a transition and return a vector-clock where the transition is concatenated to the word in the proper entry, and all the others are set to ε. The set of these τ_i is called $\vec{\tau}_{Tr}$. The operations *op* work like the ones devised for the branching processes.

With $\mathcal{U}_{Tr}(\mathsf{N})$ we denote the trellis obtained by the *mc*-net N we have the following result:

Proposition 3. *Let* N *be a mc-net, and let* $\mathfrak{S}^{\mathcal{A}_{Tr}}_{\vec{\tau}_{Tr}}(\mathsf{N})$ *be its spreading with respect to* $\vec{\tau}_{Tr}$ *and* $\mathcal{A}_{Tr}$*.*

Then $\mathsf{support}(\mathfrak{S}^{\mathcal{A}_{Tr}}_{\vec{\tau}_{Tr}}(\mathsf{N}))$ *is isomorphic to* $\mathcal{U}_{Tr}(\mathsf{N})$*.*

Example 9. *Figure 6 shows the first part of the spreading of* N *according to the* $\mathcal{A}_{Tr}$ *domain. The object constructed in this way is again infinite. Conflicts are folded in each automata according to the length of*

the local executions.

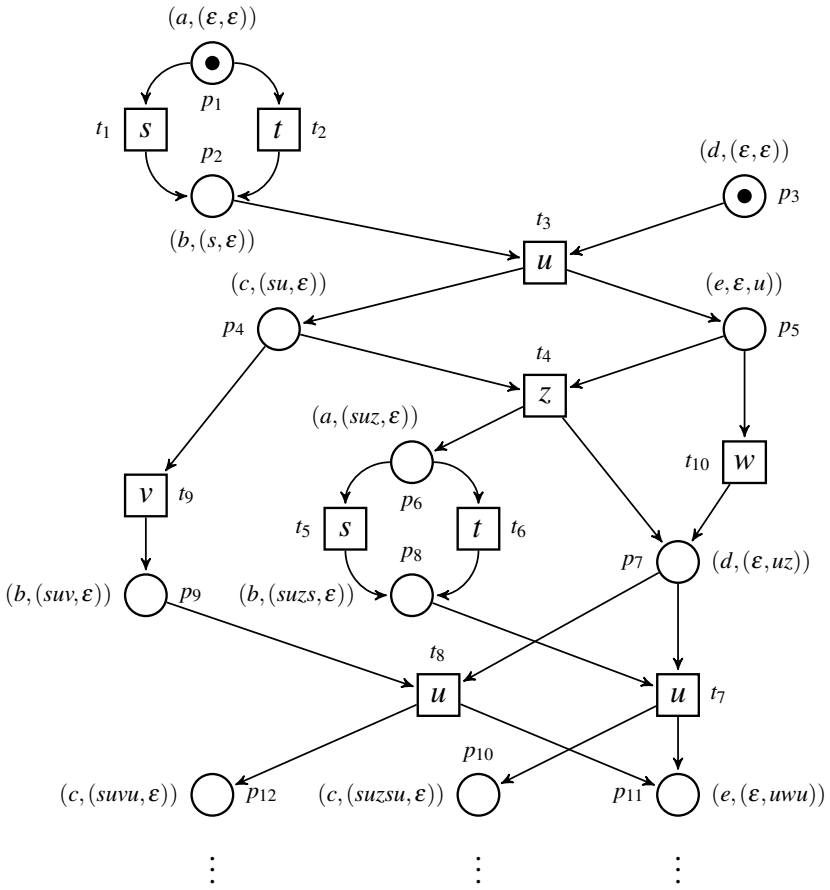

Figure 6: The initial part of the trellis of the *mc*-net $\mathcal{N}$ in Figure 1

Each place is annotated with the language of all the traces leading to the place (with respect to the equivalence relation). Consider the place p_{12}*. It belongs to the first component, and the* distance *from* p_1 *is* 4*. The two words of belonging to the language of the first automaton ending in the image of* p_{12} *(c) are suvu and tuvu. If we consider the place* p_{11}*, it belongs to the second component, and the annotation uwu is the equivalence class containing also uzu, which are the two words of length* 3 *ending in the image of* p_{11} *(e).*

6 Conclusions

In this paper we have presented the notion of spread net which is able to represent the non sequential behaviors of safe nets, in particular of *mc*-nets. A spread net is a net where each place has an annotation representing the amount of information that has been collected to *produce* that place, and the information depends on two elements. One element is the information inferred from the annotations of the places in the preset of the transitions in the preset of that place, and the second element is the transition itself.

Beside the notion of spread net we have formalized the algorithm for spreading a net, which is basically the same algorithm which is used to unfold a net. Here we have presented the usual one based

on the notion of configuration of a spread net, but the annotations of places may be used to define more easily which subset of the involved places is a part of a marking reachable in the spread net and henceforth corresponding to a marking of the unfolded net.

Here we have considered very simple domains, without making any real consideration on the kind of properties one would like to prove on spread nets. But the main advantage of the notion is the fact that it is indeed independent on the chosen domain, hence it can be used in quite different context.

In this paper we have not investigated an interesting issue, namely what is the brand of event structure related to spread net, like it is done in [15]. However we believe that configuration structures can be easily related with spread nets, hence part of the results presented there should be applicable also in our setting. Clearly the kind of event structure related to spread nets will be somehow parametric on the kind of annotations of the spread net.

Acknowledgments. This work is partially supported by Aut. Reg. of Sardinia projects "Sardcoin" and "Smart collaborative engineering". The authors wish to thank the ICEcreamers and the anonymous reviewers for their useful comments, suggestions and criticisms.

References

[1] Sandie Balaguer, Thomas Chatain & Stefan Haar (2013): *Building Occurrence Nets from Reveals Relations*. *Fundamamenta Informaticae* 123(3), pp. 245–272. Available at `http://dx.doi.org/10.3233/FI-2013-809`.

[2] Giovanni Casu & G. Michele Pinna (2017): *Merging Relations: A Way to Compact Petri Nets' Behaviors Uniformly*. In Frank Drewes, Carlos Martín-Vide & Bianca Truthe, editors: *LATA 2017 Conference Proceedings*, *Lecture Notes in Computer Science* 10168, pp. 325–337. Available at `https://doi.org/10.1007/978-3-319-53733-7_24`.

[3] Giovanni Casu & G. Michele Pinna (2017): *Petri nets and dynamic causality for service-oriented computations*. In Ahmed Seffah, Birgit Penzenstadler, Carina Alves & Xin Peng, editors: *SAC 2017 Conference Proceedings*, ACM, pp. 1326–1333. Available at `http://doi.acm.org/10.1145/3019612.3019806`.

[4] Pierpaolo Degano, José Meseguer & Ugo Montanari (1989): *Axiomatizing Net Computations and Processes*. In: *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, IEEE Computer Society, pp. 175–185. Available at `https://doi.org/10.1109/LICS.1989.39172`.

[5] Pierpaolo Degano, José Meseguer & Ugo Montanari (1996): *Axiomatizing the Algebra of Net Computations and Processes*. *AI* 33(7), pp. 641–667. Available at `https://doi.org/10.1007/BF03036469`.

[6] Joost Engelfriet (1991): *Branching Processes of Petri Nets*. *Acta Informatica* 28(6), pp. 575–591. Available at `https://doi.org/10.1007/BF01463946`.

[7] Eric Fabre (2007): *Trellis Processes : A Compact Representation for Runs of Concurrent Systems*. *Discrete Event Dynamic Systems* 17(3), pp. 267–306. Available at `https://doi.org/10.1007/s10626-006-0001-0`.

[8] Ursula Goltz & Wolfgang Reisig (1983): *The Non-sequential Behavior of Petri Nets*. *Information and Control* 57(2/3), pp. 125–147. Available at `https://doi.org/10.1016/S0019-9958(83)80040-0`.

[9] Stefan Haar, Christian Kern & Stefan Schwoon (2013): *Computing the reveals relation in occurrence nets*. *Theoretical Computer Science* 493, pp. 66–79. Available at `http://dx.doi.org/10.1016/j.tcs.2013.04.028`.

[10] Victor Khomenko, Alex Kondratyev, Maciej Koutny & Walter Vogler (2006): *Merged Processes: a new condensed representation of Petri net behaviour*. *Acta Informatica* 43(5), pp. 307–330. Available at `http://dx.doi.org/10.1007/s00236-006-0023-y`.

[11] Victor Khomenko, Maciej Koutny & Walter Vogler (2003): *Canonical prefixes of Petri net unfoldings*. *Acta Informatica* 40(2), pp. 95–118. Available at http://dx.doi.org/10.1007/s00236-003-0122-y.

[12] Kenneth L. McMillan (1993): *Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits*. In Gregor von Bochmann & David K. Probst, editors: *Computer Aided Verification, Fourth International Workshop, CAV '92, Montreal, Canada, June 29 - July 1, 1992, Proceedings*, *Lecture Notes in Computer Science* 663, Springer, pp. 164–177. Available at http://dx.doi.org/10.1007/3-540-56496-9_14.

[13] Mogens Nielsen, Gordon D. Plotkin & Glynn Winskel (1981): *Petri Nets, Event Structures and Domains, Part 1*. *Theoretical Computer Science* 13, pp. 85–108. Available at https://doi.org/10.1016/0304-3975(81)90112-2.

[14] Einar Smith & Wolfgang Reisig (1987): *The Semantics of a Net is a Net*. In Klaus Voss, Hartmann J. Genrich & Grzegorz Rozenberg, editors: *Concurrency and Nets*, Springer Verlag, pp. 461–479. Available at https://doi.org/10.1007/978-3-642-72822-8_29.

[15] Rob J. van Glabbeek & Gordon D. Plotkin (2009): *Configuration structures, event structures and Petri nets*. *Theoretical Computer Science* 410(41), pp. 4111–4159. Available at https://doi.org/10.1016/j.tcs.2009.06.014.

[16] Glynn Winskel (1987): *Event Structures*. In Wilfried Brauer, Wolfgang Reisig & Grzegorz Rozenberg, editors: *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, *Lecture Notes in Computer Science* 255, Springer Verlag, pp. 325–392. Available at https://doi.org/10.1007/3-540-17906-2_31.